Package ‘broom’
January 29, 2022

Type   Package
Title  Convert Statistical Objects into Tidy Tibbles
Version 0.7.12

Description  Summarizes key information about statistical objects in tidy tibbles. This makes it easy to report results, create plots and consistently work with large numbers of models at once. Broom provides three verbs that each provide different types of information about a model. tidy() summarizes information about model components such as coefficients of a regression. glance() reports information about an entire model, such as goodness of fit measures like AIC and BIC. augment() adds information about individual observations to a dataset, such as fitted values or influence measures.

License  MIT + file LICENSE


BugReports  https://github.com/tidymodels/broom/issues

Depends  R (>= 3.1)

Imports  backports, dplyr (>= 1.0.0), ellipsis, generics (>= 0.0.2), glue, methods, purrr, rlang, tibble (>= 3.0.0), tidyr (>= 1.0.0), ggplot2

Suggests  AER, akima, AUC, bbmle, betareg, biglm, binGroup, boot, btergm, car, caret, cluster, cmprsk, coda, covr, drc, e1071, emmeans, epiR, ergm (>= 3.10.4), fixest (>= 0.9.0), gam (>= 1.15), gee, geepack, glmnet, glmnetUtils, gmm, Hmisc, irlba, joineRML, Kendall, knitr, ks, Lahman, lavaan, leaps, lfe, lm.beta, lme4, lmodel2, lmtree (>= 0.9.38), lsmeans, maps, mapproots, margins, MASS, Matrix, mclust, mediation, metafor, mfx, mgcv, mlogit, modeldata, modeltests, muhaz, multcomp, network, nnet, orcutt (>= 2.2), ordinal, plm, poLCA, psych, quantreg, rgeos, rmarkdown, robust, robustbase, rsample, sandwich, sp, spdep (>= 1.1), spatialreg, speedglm, spelling, survey, survival, systemfit, testthat (>= 2.1.0), tseries, vars, zoo
'zoo-tidiers.R' 'zzz.R'

**NeedsCompilation**: no

**Author**: David Robinson [aut],
Alex Hayes [aut] (<https://orcid.org/0000-0002-4985-5160>),
Simon Couch [aut, cre] (<https://orcid.org/0000-0001-5676-5107>),
Indrajeet Patil [ctb] (<https://orcid.org/0000-0003-1995-6531>),
Derek Chiu [ctb],
Matthieu Gomez [ctb],
Boris Demeshev [ctb],
Dieter Menne [ctb],
Benjamin Nutter [ctb],
Luke Johnston [ctb],
Ben Bolker [ctb],
Francois Briatte [ctb],
Jeffrey Arnold [ctb],
Jonah Gabry [ctb],
Luciano Selzer [ctb],
Gavin Simpson [ctb],
Jens Preussner [ctb],
Jay Hesselberth [ctb],
Hadley Wickham [ctb],
Matthew Lincoln [ctb],
Alessandro Gasparini [ctb],
Lukasz Komsta [ctb],
Frederick Novometsky [ctb],
Wilson Freitas [ctb],
Michelle Evans [ctb],
Jason Cory Brunson [ctb],
Simon Jackson [ctb],
Ben Whalley [ctb],
Karissa Whiting [ctb],
Yves Rosseel [ctb],
Michael Kuehn [ctb],
Jorge Cimentada [ctb],
Erle Holgersen [ctb],
Karl Dunkle Werner [ctb] (<https://orcid.org/0000-0003-0523-7309>),
Ethan Christensen [ctb],
Steven Pav [ctb],
Paul PJ [ctb],
Ben Schneider [ctb],
Patrick Kennedy [ctb],
Lily Medina [ctb],
Brian Fannin [ctb],
Jason Muhlenkamp [ctb],
Matt Lehman [ctb],
Bill Denney [ctb] (<https://orcid.org/0000-0002-5759-428X>),
Nic Crane [ctb],
Andrew Bates [ctb],
Zoe Wilkinson Saldana [ctb],
Tyler Littlefield [ctb],
Charles T. Gray [ctb],
Shabhb E. Banks [ctb],
Serina Robinson [ctb],
Roger Bivand [ctb],
Riinu Ots [ctb],
Nicholas Williams [ctb],
Nina Jakobsen [ctb],
Michael Weylandt [ctb],
Lisa Lendway [ctb],
Karl Hailperin [ctb],
Josue Rodriguez [ctb],
Jenny Bryan [ctb],
Chris Jarvis [ctb],
Greg Macfarlane [ctb],
Brian Mannakee [ctb],
Drew Tyre [ctb],
Shreyas Singh [ctb],
Laurens Geffert [ctb],
Hong Ooi [ctb],
Henrik Bengtsson [ctb],
Eduard Szocs [ctb],
David Hugh-Jones [ctb],
Matthieu Stigler [ctb],
Hugo Tavares [ctb] (<https://orcid.org/0000-0001-9373-2726>),
R. Willem Vervoort [ctb],
Brenton M. Wiernik [ctb],
Josh Yamamoto [ctb],
Jasme Lee [ctb],
Taren Sanders [ctb] (<https://orcid.org/0000-0002-4504-6008>),
Ilaria Prosdocimi [ctb] (<https://orcid.org/0000-0001-8565-094X>),
Daniel D. Sjoberg [ctb] (<https://orcid.org/0000-0003-0862-2018>)

Maintainer  Simon Couch <simonpatrickcouch@gmail.com>

Repository  CRAN

Date/Publication  2022-01-28 23:30:05 UTC

R topics documented:

augment.betamfx ........................................... 10
augment.betareg ........................................... 13
augment.clm ................................................ 15
augment.coxph ............................................. 17
augment.decomposed.ts ................................... 20
augment.drc ................................................ 22
augment.factanal .......................................... 25
augment.felm .............................................. 26
<table>
<thead>
<tr>
<th>R topics documented:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>augment.fixest</td>
<td>28</td>
</tr>
<tr>
<td>augment.gam</td>
<td>31</td>
</tr>
<tr>
<td>augment.glm</td>
<td>33</td>
</tr>
<tr>
<td>augment.glmRob</td>
<td>35</td>
</tr>
<tr>
<td>augment.glmrob</td>
<td>36</td>
</tr>
<tr>
<td>augment.htest</td>
<td>38</td>
</tr>
<tr>
<td>augment.ivreg</td>
<td>40</td>
</tr>
<tr>
<td>augment.kmeans</td>
<td>42</td>
</tr>
<tr>
<td>augment.lm</td>
<td>44</td>
</tr>
<tr>
<td>augment.lmRob</td>
<td>47</td>
</tr>
<tr>
<td>augment.lmrob</td>
<td>49</td>
</tr>
<tr>
<td>augment.loess</td>
<td>51</td>
</tr>
<tr>
<td>augment.Mclust</td>
<td>53</td>
</tr>
<tr>
<td>augment.mfx</td>
<td>55</td>
</tr>
<tr>
<td>augment.mjoint</td>
<td>58</td>
</tr>
<tr>
<td>augment.mlogit</td>
<td>61</td>
</tr>
<tr>
<td>augment.nlrq</td>
<td>62</td>
</tr>
<tr>
<td>augment.nls</td>
<td>64</td>
</tr>
<tr>
<td>augment.pam</td>
<td>66</td>
</tr>
<tr>
<td>augment.plm</td>
<td>68</td>
</tr>
<tr>
<td>augment.polCA</td>
<td>70</td>
</tr>
<tr>
<td>augment.polr</td>
<td>72</td>
</tr>
<tr>
<td>augment.pcomp</td>
<td>74</td>
</tr>
<tr>
<td>augment.rlm</td>
<td>76</td>
</tr>
<tr>
<td>augment.rma</td>
<td>78</td>
</tr>
<tr>
<td>augment.rq</td>
<td>80</td>
</tr>
<tr>
<td>augment.rqs</td>
<td>82</td>
</tr>
<tr>
<td>augment.sarlm</td>
<td>83</td>
</tr>
<tr>
<td>augment.smooth.spline</td>
<td>85</td>
</tr>
<tr>
<td>augment.speedlm</td>
<td>87</td>
</tr>
<tr>
<td>augment.stl</td>
<td>88</td>
</tr>
<tr>
<td>augment survreg</td>
<td>90</td>
</tr>
<tr>
<td>augment_columns</td>
<td>92</td>
</tr>
<tr>
<td>bootstrap</td>
<td>93</td>
</tr>
<tr>
<td>confint tidy</td>
<td>94</td>
</tr>
<tr>
<td>data.frame_tidiers</td>
<td>95</td>
</tr>
<tr>
<td>durbinWatsonTest_tidiers</td>
<td>97</td>
</tr>
<tr>
<td>finish_glance</td>
<td>98</td>
</tr>
<tr>
<td>fix_data_frame</td>
<td>99</td>
</tr>
<tr>
<td>glance.aareg</td>
<td>99</td>
</tr>
<tr>
<td>glance.aov</td>
<td>101</td>
</tr>
<tr>
<td>glance.Arima</td>
<td>102</td>
</tr>
<tr>
<td>glance.betamfx</td>
<td>103</td>
</tr>
<tr>
<td>glance.betareg</td>
<td>105</td>
</tr>
<tr>
<td>glance.biglm</td>
<td>106</td>
</tr>
<tr>
<td>glance.binDesign</td>
<td>108</td>
</tr>
<tr>
<td>glance.cch</td>
<td>110</td>
</tr>
<tr>
<td>glance.clm</td>
<td>111</td>
</tr>
</tbody>
</table>
R topics documented:

glance.clmm .............................................................. 113

glance.coeftest .......................................................... 115

glance.coxph .............................................................. 117

glance.crr ................................................................. 119

glance.cv.glmnet ........................................................ 120

glance.drc ................................................................. 123

glance.ergm ............................................................... 124

glance.factanal .......................................................... 125

glance.felm ............................................................... 127

glance.fitdistr ........................................................... 129

glance.fixest ............................................................. 130

glance.Gam ............................................................... 132

glance.gam ............................................................... 134

glance.garch ............................................................. 135

glance.geeglm ........................................................... 136

glance.glm ............................................................... 138

glance.glmnet ........................................................... 139

glance.glmRob ........................................................... 141

glance.gmm ............................................................... 142

glance.ivreg ............................................................. 144

glance.kmeans ............................................................ 146

glance.lavaan ............................................................ 148

glance.lm ................................................................. 150

glance.lmodel2 ........................................................... 152

glance.lmRob ............................................................. 154

glance.lmrob ............................................................. 155

glance.margins .......................................................... 157

glance.Mclust ............................................................ 159

glance.mfx .............................................................. 160

glance.mjoint ............................................................ 163

glance.mlogit ............................................................. 165

glance.muhaz ............................................................. 166

glance.multinom ........................................................ 168

glance.negbin ........................................................... 169

glance.nlrq ............................................................... 171

glance.nls ............................................................... 172

glance.orcutt ........................................................... 173

glance.pam .............................................................. 175

glance.plm .............................................................. 176

glance.poLCA ............................................................. 178

glance.polr ............................................................... 180

glance.pyears ........................................................... 182

glance.ridgelm .......................................................... 183

glance.rlm .............................................................. 185

glance.rma .............................................................. 186

glance.rq ................................................................. 188

glance.sarlm ............................................................. 189

glance.smooth.spline .................................................. 191
R topics documented:

glance.speedglm ......................................................... 192
  glance.speedlm ..................................................... 194
  glance.summary.lm ................................................. 195
  glance.survdiff ................................................... 198
  glance.surexp ..................................................... 199
  glance.survfit ................................................... 201
  glance.survreg ................................................... 203
  glance.svyglm .................................................... 204
  glance.svyolr .................................................... 206
  glance.varest ................................................... 208
  glance_optim .................................................... 209
  leveneTest_tidiers ................................................. 210
  list_tidiers ..................................................... 211
  null_tidiers ..................................................... 212
  sparse_tidiers ................................................... 213
  sp_tidiers ....................................................... 214
  summary_tidiers .................................................. 215
  tidy.aareg ....................................................... 216
  tidy.acf .......................................................... 217
  tidy.anova ........................................................ 218
  tidy.aov .......................................................... 220
  tidy.aovlist ...................................................... 221
  tidy.Arima ........................................................ 222
  tidy.betamfx ...................................................... 223
  tidy.betareg ...................................................... 225
  tidy.biglm ........................................................ 227
  tidy.binDesign ................................................... 228
  tidy.binWidth ..................................................... 230
  tidy.boot .......................................................... 231
  tidy.btergm ....................................................... 233
  tidy.cch ........................................................... 234
  tidy.cld ........................................................... 235
  tidy.clm .......................................................... 237
  tidy.clmm ........................................................ 239
  tidy.coefest ...................................................... 241
  tidy.confint.glht ............................................... 242
  tidy.confusionMatrix ............................................. 244
  tidy.coxph ........................................................ 245
  tidy.crr ........................................................... 247
  tidy.cv.glmnet ................................................... 249
  tidy.density ...................................................... 251
  tidy.dist .......................................................... 252
  tidy.drc ........................................................... 253
  tidy.emmGrid ...................................................... 254
  tidy.epi.2by2 ..................................................... 256
  tidy.ergm .......................................................... 258
  tidy.factanal ..................................................... 260
  tidy.felm .......................................................... 261
R topics documented:

tidy.fitdistr .................................................. 263
tidy.fixest .................................................. 264
tidy.ftable .................................................. 266
tidy.Gam ..................................................... 267
tidy.gam ..................................................... 268
tidy.garch ................................................... 270
tidy.geeglm .................................................. 272
tidy.glht ..................................................... 273
tidy.glm ..................................................... 275
tidy.glmnet .................................................. 276
tidy.glmRob .................................................. 278
tidy.glmrob .................................................. 279
tidy.gmm ..................................................... 280
tidy.htest ................................................... 283
tidy.ivreg .................................................... 284
tidy.kappa ................................................... 286
tidy.kde ...................................................... 288
tidy.Kendall ................................................ 289
tidy.kmeans .................................................. 291
tidy.lavaan ................................................... 292
tidy.lm ....................................................... 294
tidy.lm.beta ................................................ 296
tidy.lmodel2 ............................................... 298
tidy.lmRob .................................................. 299
tidy.lmrob .................................................... 300
tidy.lsmobj .................................................. 302
tidy.manova .................................................. 304
tidy.map ...................................................... 305
tidy.margins .................................................. 306
tidy.Mclust .................................................. 308
tidy.mediate ................................................ 310
tidy.mfx ...................................................... 311
tidy.mjjoint ................................................ 314
tidy.mle2 ..................................................... 316
tidy.mlm ...................................................... 317
tidy.mlogit .................................................. 319
tidy.muhaz ................................................... 320
tidy.multinom ................................................ 321
tidy.negbin ................................................... 323
tidy.nlrq ..................................................... 324
tidy.nls ...................................................... 325
tidy.numeric .................................................. 327
tidy.orcutt .................................................. 328
tidy.pairwise.htest ........................................ 329
tidy.pam ...................................................... 330
tidy.plm ...................................................... 332
tidy.poLCA ................................................... 334
tidy.polr ...................................................... 336
Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.
Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

### Usage

```r
## S3 method for class 'betamfx'
augment(
  x,
  data = model.frame(x$fit),
  newdata = NULL,
  type.predict = c("response", "link", "precision", "variance", "quantile"),
  type.residuals = c("sweighted2", "deviance", "pearson", "response", "weighted", "sweighted"),
  ...
)
```

### Arguments

- **x**: A betamfx object.
- **data**: A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
- **newdata**: A base::data.frame or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict Character indicating type of prediction to use. Passed to the type argument of \texttt{betareg::predict.betareg()}. Defaults to "response".

type.residuals Character indicating type of residuals to use. Passed to the type argument of \texttt{betareg::residuals.betareg()}. Defaults to "sweighted2.

... Additional arguments. Not used. Needed to match generic signature only. \textbf{Cautionary note:} Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass \texttt{conf.lvel = 0.9}, all computation will proceed using \texttt{conf.lvel = 0.95}. Additionally, if you pass \texttt{newdata = my_tibble} to an \texttt{augment()} method that does not accept a \texttt{newdata} argument, it will use the default value for the data argument.

Details

This augment method wraps \texttt{augment.betareg()} for \texttt{mfx::betamfx()} objects.

Value

A \texttt{tibble::tibble()} with columns:

- \texttt{.cooksd} Cooks distance.
- \texttt{.fitted} Fitted or predicted value.
- \texttt{.resid} The difference between observed and fitted values.

See Also

\texttt{augment.betareg()}, \texttt{mfx::betamfx()}

Other mfx tidiers: \texttt{augment.mfx()}, \texttt{glance.betamfx()}, \texttt{glance.mfx()}, \texttt{tidy.betamfx()}, \texttt{tidy.mfx()}

Examples

```r
## Not run:
library(mfx)

## Simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

## Beta outcome
y = rbeta(n, shape1 = plogis(1 + 0.5 * x), shape2 = (abs(0.2*x)))
## Use Smithson and Verkuilen correction
y = (y*(n-1)+0.5)/n

d = data.frame(y,x)
mod_betamfx = betamfx(y ~ x | x, data = d)
tidy(mod_betamfx, conf.int = TRUE)

## Compare with the naive model coefficients of the equivalent betareg call (not run)
```
augment.betareg

# tidy(betamfx(y ~ x | x, data = d), conf.int = TRUE)

augment(mod_betamfx)
glance(mod_betamfx)

## End(Not run)

augment.betareg  Augment data with information from a(n) betareg object

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

**Usage**

## S3 method for class 'betareg'

augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict,
  type.residuals,
Arguments

x  A betareg object produced by a call to betareg::betareg().
data  A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. Do not pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.

newdata  A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict  Character indicating type of prediction to use. Passed to the type argument of the stats::predict() generic. Allowed arguments vary with model class, so be sure to read the predict.my_class documentation.
type.residuals  Character indicating type of residuals to use. Passed to the type argument of stats::residuals() generic. Allowed arguments vary with model class, so be sure to read the residuals.my_class documentation.

Details

For additional details on Cook’s distance, see stats::cooks.distance().

Value

A tibble::tibble() with columns:

.cooksd  Cooks distance.
.fitted  Fitted or predicted value.
.resid   The difference between observed and fitted values.

See Also

augment(), betareg::betareg()
```r
if (requireNamespace("betareg", quietly = TRUE)) {
  library(betareg)
  data("GasolineYield", package = "betareg")
  mod <- betareg(yield ~ batch + temp, data = GasolineYield)
  mod
  tidy(mod)
  tidy(mod, conf.int = TRUE)
  tidy(mod, conf.int = TRUE, conf.level = .99)
  augment(mod)
  glance(mod)
}
```

**augment.clm**

Augment data with information from a(n) clm object

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a **tibble::tibble** with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that **splines::ns(), stats::poly()** and
survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

## S3 method for class 'clm'

```r
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict = c("prob", "class"),
  ...
)
```

Arguments

- `x`: A clm object returned from ordinal::clm().
- `data`: A base::data.frame or tibble::tibble() containing the original data that was used to produce the object `x`. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. Do not pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
- `newdata`: A base::data.frame() or tibble::tibble() containing all the original predictors used to create `x`. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
- `type.predict`: Which type of prediction to compute, either "prob" or "class", passed to ordinal::predict.clm(). Defaults to "prob".
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

See Also
tidy, ordinal::clm(), ordinal::predict.clm()

Other ordinal tidiers: augment.polr(), glance.clmm(), glance.clm(), glance.polr(), glance.svyolr(), tidy.clmm(), tidy.clm(), tidy.polr(), tidy.svyolr()

Examples

```r
if (requireNamespace("ordinal", quietly = TRUE)) {

```
library(ordinal)

fit <- clm(rating ~ temp * contact, data = wine)

tidy(fit)
tidy(fit, conf.int = TRUE, conf.level = 0.9)
tidy(fit, conf.int = TRUE, conf.type = "Wald", exponentiate = TRUE)

glance(fit)
augment(fit, type.predict = "prob")
augment(fit, type.predict = "class")

fit2 <- clm(rating ~ temp, nominal = ~contact, data = wine)
tidy(fit2)
glance(fit2)

}
We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'coxph'
augment(
  x,
  data = NULL,
  newdata = NULL,
  type.predict = "lp",
  type.residuals = "martingale",
  ...
)
```

Arguments

- **x**: A `coxph` object returned from `survival::coxph()`.
- **data**: A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
- **newdata**: A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
- **type.predict**: Character indicating type of prediction to use. Passed to the `type` argument of the `stats::predict()` generic. Allowed arguments vary with model class, so be sure to read the `predict.my_class` documentation.
- **type.residuals**: Character indicating type of residuals to use. Passed to the `type` argument of the `stats::residuals()` generic. Allowed arguments vary with model class, so be sure to read the `residuals.my_class` documentation.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

When the modeling was performed with `na.action = "na.omit"` (as is the typical default), rows with NA in the initial data are omitted entirely from the augmented data frame. When the modeling was performed with `na.action = "na.exclude"`, one should provide the original data as a second argument, at which point the augmented data will contain those rows (typically with NAs
in place of the new columns). If the original data is not provided to `augment()` and `na.action = "na.exclude"`, a warning is raised and the incomplete rows are dropped.

**Value**

A `tibble::tibble()` with columns:

- `.fitted` Fitted or predicted value.
- `.resid` The difference between observed and fitted values.
- `.se.fit` Standard errors of fitted values.

**See Also**

- `stats::na.action`
- `augment()`, `survival::coxph()`

Other coxph tidiers: `glance.coxph()`, `tidy.coxph()`

Other survival tidiers: `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

**Examples**

```r
if (requireNamespace("survival", quietly = TRUE)) {
  library(survival)
  cfit <- coxph(Surv(time, status) ~ age + sex, lung)
  tidy(cfit)
  tidy(cfit, exponentiate = TRUE)
  lp <- augment(cfit, lung)
  risks <- augment(cfit, lung, type.predict = "risk")
  expected <- augment(cfit, lung, type.predict = "expected")
  glance(cfit)
  # also works on clogit models
  resp <- levels(logan$occupation)
  n <- nrow(logan)
  indx <- rep(1:n, length(resp))
  logan2 <- data.frame(
    logan[indx, ],
    id = indx,
    tocc = factor(rep(resp, each = n))
  )
  logan2$case <- (logan2$occupation == logan2$tocc)
```
cl <- clogit(case ~ tocc + tocc:education + strata(id), logan2)
tidy(cl)
glance(cl)

library(ggplot2)

ggplot(lp, aes(age, .fitted, color = sex)) + geom_point()

ggplot(risks, aes(age, .fitted, color = sex)) + geom_point()

ggplot(expected, aes(time, .fitted, color = sex)) + geom_point()


describe::describe(frame)

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::pol() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.
Usage

```r
## S3 method for class 'decomposed.ts'
augment(x, ...)
```

Arguments

- `x`: A `decomposed.ts` object returned from `stats::decompose()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble` with one row for each observation in the original times series:

- `.seasonal`: The seasonal component of the decomposition.
- `.trend`: The trend component of the decomposition.
- `.remainder`: The remainder, or "random" component of the decomposition.
- `.weight`: The final robust weights (`stl` only).
- `.seasadj`: The seasonally adjusted (or "deseasonalised") series.

See Also

- `augment()`, `stats::decompose()`
- Other decompose tidiers: `augment.stl()`

Examples

```r
# Time series of temperatures in Nottingham, 1920-1939:
nottem

# Perform seasonal decomposition on the data with both decompose
# and stl:
d1 <- stats::decompose(nottem)
d2 <- stats::stl(nottem, s.window = "periodic", robust = TRUE)

# Compare the original series to its decompositions.
cbind(
  broom::tidy(nottem), broom::augment(d1),
  broom::augment(d2)
)
```
# Visually compare seasonal decompositions in tidy data frames.

library(tibble)
library(dplyr)
library(tidyr)
library(ggplot2)

decomps <- tibble(
  # Turn the ts objects into data frames.
  series = list(as.data.frame(nottem), as.data.frame(nottem)),
  # Add the models in, one for each row.
  decomp = c("decompose", "stl"),
  model = list(d1, d2)
) %>%
  rowwise() %>%
  # Pull out the fitted data using broom::augment.
  mutate(augment = list(broom::augment(model))) %>%
  ungroup() %>%
  # Unnest the data frames into a tidy arrangement of
  # the series next to its seasonal decomposition, grouped
  # by the method (stl or decompose).
  group_by(decomp) %>%
  unnest(c(series, augment)) %>%
  ungroup() %>%
  select(decomp, index, x, adjusted = .seasadj)

ggplot(decomps) +
  geom_line(aes(x = index, y = x), colour = "black") +
  geom_line(aes(
    x = index, y = adjusted, colour = decomp,
    group = decomp
  ))

---

**augment.drc**

*Augment data with information from an drc object*

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no `.resid` column will be included in the output.
Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a `tibble::tibble` with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

**Usage**

```r
## S3 method for class 'drc'
augment(
  x, 
  data = NULL, 
  newdata = NULL, 
  se_fit = FALSE, 
  conf.int = FALSE, 
  conf.level = 0.95, 
  ...
)
```

**Arguments**

- **x**
  A `drc` object produced by a call to `drc::drm()`.

- **data**
  A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. Do not pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.

- **newdata**
  A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the data argument will be ignored.

- **se_fit**
  Logical indicating whether or not a `.se.fit` column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to `FALSE`.

- **conf.int**
  Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`. 
conf.level  The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

...  Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value
A tibble::tibble() with columns:
  .cooks  Cooks distance.
  .fitted  Fitted or predicted value.
  .lower  Lower bound on interval for fitted values.
  .resid  The difference between observed and fitted values.
  .se.fit  Standard errors of fitted values.
  .upper  Upper bound on interval for fitted values.

See Also
augment(), drc::drm()
Other drc tidiers: glance.drc(), tidy.drc()

Examples

if (requireNamespace("drc", quietly = TRUE)) {
  library(drc)
  mod <- drm(dead / total ~ conc, type, 
             weights = total, data = selenium, fct = LL.2(), type = "binomial"
  )
  tidy(mod)
  tidy(mod, conf.int = TRUE)
  glance(mod)
  augment(mod, selenium)
}
augment.factanal

Augment data with information from a(n) factanal object

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

## S3 method for class 'factanal'
augment(x, data, ...)

Arguments

- **x**: A factanal object created by stats::factanal().
- **data**: A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
augment.felm

Augment data with information from a(n) felm object

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model.
formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'felm'
augment(x, data = model.frame(x), ...)
```

Arguments

- `x`: A `felm` object returned from `lfe::felm()`.
- `data`: A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. Do not pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with columns:

- `.fitted`: Fitted or predicted value.
- `.resid`: The difference between observed and fitted values.

See Also

- `augment()`, `lfe::felm()`
- Other `felm` tidiers: `tidy.felm()`

Examples

```r
if (requireNamespace("lfe", quietly = TRUE)) {

library(lfe)

# Use built-in "airquality" dataset
head(airquality)
```
# No FEs; same as lm()
est0 <- felm(Ozone ~ Temp + Wind + Solar.R, airquality)
tidy(est0)
augment(est0)

# Add month fixed effects
est1 <- felm(Ozone ~ Temp + Wind + Solar.R | Month, airquality)
tidy(est1)
tidy(est1, fe = TRUE)
augment(est1)
glance(est1)

# The "se.type" argument can be used to switch out different standard errors
# types on the fly. In turn, this can be useful exploring the effect of
# different error structures on model inference.
tidy(est1, se.type = "iid")
tidy(est1, se.type = "robust")

# Add clustered SEs (also by month)
est2 <- felm(Ozone ~ Temp + Wind + Solar.R | Month | 0 | Month, airquality)
tidy(est2, conf.int = TRUE)
tidy(est2, conf.int = TRUE, se.type = "cluster")
tidy(est2, conf.int = TRUE, se.type = "robust")
tidy(est2, conf.int = TRUE, se.type = "iid")

}
For convenience, many augment methods provide default data arguments, so that `augment(fit)` will return the augmented training data. In these cases, `augment` tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a `tibble::tibble` with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

**Usage**

```r
## S3 method for class 'fixest'
augment(
  x,
  data = NULL,
  newdata = NULL,
  type.predict = c("link", "response"),
  type.residuals = c("response", "deviance", "pearson", "working"),
  ...
)
```

**Arguments**

- `x` A `fixest` object returned from any of the `fixest` estimators
- `data` A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
- `newdata` A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
- `type.predict` Passed to `predict.fixest` type argument. Defaults to "link" (like `predict.glm`).
- `type.residuals` Passed to `predict.fixest` type argument. Defaults to "response" (like `residuals.lm`, but unlike `residuals.glm`).
- `...` Additional arguments passed to `summary` and `confint`. Important arguments are `se` and `cluster`. Other arguments are `dof`, `exact_dof`, `forceCovariance`, and `keepBounded`. See `summary.fixest`.

**Value**

A `tibble::tibble()` with columns:

- `.fitted` Fitted or predicted value.
.resid The difference between observed and fitted values.

Note

Important note: fixest models do not include a copy of the input data, so you must provide it manually.

augment.fixest only works for fixest::feols(), fixest::feglm(), and fixest::femlm() models. It does not work with results from fixest::fenegbin(), fixest::feNmlm(), or fixest::fepois().

See Also

augment(), fixest::feglm(), fixest::femlm(), fixest::feols()

Other fixest tidiers: tidy.fixest()

Examples

if (requireNamespace("fixest", quietly = TRUE)) {

library(fixest)

gravity <- feols(log(Euros) ~ log(dist_km) | Origin + Destination + Product + Year, trade)

tidy(gravity)
glance(gravity)
augment(gravity, trade)

## To get robust or clustered SEs, users can either:
# 1) Or, specify the arguments directly in the tidy() call

tidy(gravity, conf.int = TRUE, cluster = c("Product", "Year"))
tidy(gravity, conf.int = TRUE, se = "threeway")

# 2) Feed tidy() a summary.fixest object that has already accepted these arguments

gridavity_summ <- summary(gravity, cluster = c("Product", "Year"))
tidy(gravity_summ, conf.int = TRUE)

# Approach (1) is preferred.

}


Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it must be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'gam'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict,  # Optional
  type.residuals,  # Optional
  ...
)
```

Arguments

- `x` A `gam` object returned from a call to `mgcv::gam()`.
data  A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.

newdata  A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.

type.predict  Character indicating type of prediction to use. Passed to the type argument of the `stats::predict()` generic. Allowed arguments vary with model class, so be sure to read the `predict.my_class` documentation.

type.residuals  Character indicating type of residuals to use. Passed to the type argument of `stats::residuals()` generic. Allowed arguments vary with model class, so be sure to read the `residuals.my_class` documentation.

...  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass newdata = my_tibble to an `augment()` method that does not accept a newdata argument, it will use the default value for the data argument.

Details

For additional details on Cook’s distance, see `stats::cooks.distance()`.

Value

A `tibble::tibble()` with columns:

- `.cooksd`  Cooks distance.
- `.fitted`  Fitted or predicted value.
- `.hat`  Diagonal of the hat matrix.
- `.resid`  The difference between observed and fitted values.
- `.se.fit`  Standard errors of fitted values.
- `.sigma`  Estimated residual standard deviation when corresponding observation is dropped from model.

See Also

`augment()`, `mgcv::gam()`
Examples

```r
if (requireNamespace("mgcv", quietly = TRUE)) {
  g <- mgcv::gam(mpg ~ s(hp) + am + qsec, data = mtcars)
  tidy(g)
  tidy(g, parametric = TRUE)
  glance(g)
  augment(g)
}
```

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a **tibble::tibble** with the same **number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.
Usage

```r
## S3 method for class 'glm'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict = c("link", "response", "terms"),
  type.residuals = c("deviance", "pearson"),
  se.fit = FALSE,
  ...
)
```

Arguments

- **x**: A `glm` object returned from `stats::glm()`.
- **data**: A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and Cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
- **newdata**: A `base::data.frame` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
- **type.predict**: Passed to `stats::predict.glm()` type argument. Defaults to "link".
- **type.residuals**: Passed to `stats::residuals.glm()` and to `stats::rstandard.glm()` type arguments. Defaults to "deviance".
- **se.fit**: Logical indicating whether or not a `.se.fit` column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to `FALSE`.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in `...` where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

If the weights for any of the observations in the model are 0, then columns ".infl" and ".hat" in the result will be 0 for those observations.

A `.resid` column is not calculated when data is specified via the `newdata` argument.
Value

A `tibble::tibble()` with columns:

- `.cooksd`  
  Cooks distance.
- `.fitted`  
  Fitted or predicted value.
- `.hat`  
  Diagonal of the hat matrix.
- `.resid`  
  The difference between observed and fitted values.
- `.se.fit`  
  Standard errors of fitted values.
- `.sigma`  
  Estimated residual standard deviation when corresponding observation is dropped from model.
- `.std.resid`  
  Standardised residuals.

See Also

- `stats::glm()`  
  Other lm tidiers: `augment.lm()`, `glance.glm()`, `glance.lm()`, `glance.summary.lm()`, `glance.svyglm()`, `tidy.glm()`, `tidy.lm.beta()`, `tidy.lm()`, `tidy.mlm()`, `tidy.summary.lm()`

---

`augment.glmRob`  
`Augment data with information from a(n) glmRob object`

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it must be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a `tibble::tibble` with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and
survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'glmRob'
augment(x, ...)
```

Arguments

- `x`: Unused.
- `...`: Unused.

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.
augment.glmrob

Usage

```r
## S3 method for class 'glmrob'
augment(
x,
data = model.frame(x),
newdata = NULL,
type.predict = c("link", "response"),
type.residuals = c("deviance", "pearson"),
se.fit = FALSE,
...)
```

Arguments

- **x**: A `glmrob` object returned from `robustbase::glmrob()`.
- **data**: A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
- **newdata**: A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
- **type.predict**: Character indicating type of prediction to use. Passed to the type argument of the `stats::predict()` generic. Allowed arguments vary with model class, so be sure to read the `predict.my_class` documentation.
- **type.residuals**: Character indicating type of residuals to use. Passed to the type argument of the `stats::residuals()` generic. Allowed arguments vary with model class, so be sure to read the `residuals.my_class` documentation.
- **se.fit**: Logical indicating whether or not a `.se.fit` column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to `FALSE`.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Details

For tidiers for robust models from the `MASS` package see `tidy.rlm()`.
Value

A `tibble::tibble()` with columns:

- `.fitted` Fitted or predicted value.
- `.resid` The difference between observed and fitted values.

See Also

- `robustbase::glmrob()`
- Other robustbase tidiers: `augment.lmrob()`, `glance.lmrob()`, `tidy.glmrob()`, `tidy.lmrob()`

---

### Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it must be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a `tibble::tibble` with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

### Usage

```r
## S3 method for class 'htest'
augment(x, ...)
```
Arguments

- **x**: An htest object, such as those created by `stats::cor.test()`, `stats::t.test()`, `stats::wilcox.test()`, `stats::chisq.test()`, etc.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

See `stats::chisq.test()` for more details on how residuals are computed.

Value

A `tibble::tibble()` with exactly one row and columns:

- **.observed**: Observed count.
- **.prop**: Proportion of the total.
- **.row.prop**: Row proportion (2 dimensions table only).
- **.col.prop**: Column proportion (2 dimensions table only).
- **.expected**: Expected count under the null hypothesis.
- **.resid**: Pearson residuals.
- **.std.resid**: Standardized residual.

See Also

`augment()`, `stats::chisq.test()`

Other htest tidiers: `tidy.htest()`, `tidy.pairwise.htest()`, `tidy.power.htest()`

Examples

```r
tt <- t.test(rnorm(10))
tidy(tt)
# Same output for all htests

# tt <- t.test(mpg ~ am, data = mtcars)
tidy(tt)

wt <- wilcox.test(mpg ~ am, data = mtcars, conf.int = TRUE, exact = FALSE)
tidy(wt)

ct <- cor.test(mtcars$wt, mtcars$mpg)
tidy(ct)
```
chit <- chisq.test(xtabs(Freq ~ Sex + Class, data = as.data.frame(Titanic)))
tidy(chit)
augment(chit)

---

### augment.ivreg

**Augment data with information from a(n) ivreg object**

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a `tibble::tibble` with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

**Usage**

```r
## S3 method for class 'ivreg'
augment(x, data = model.frame(x), newdata = NULL, ...)
```

**Arguments**

- `x` An ivreg object created by a call to `AER::ivreg()`.
- `data` A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data
to the data argument. Augment will report information such as influence and
cooks distance for data passed to the data argument. These measures are only
defined for the original training data.

newdata
A base::data.frame() or tibble::tibble() containing all the original pre-
dictors used to create x. Defaults to NULL, indicating that nothing has been
passed to newdata. If newdata is specified, the data argument will be ignored.

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be
ignored. If the misspelled argument has a default value, the default value will be
used. For example, if you pass conf.level = 0.9, all computation will proceed
using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an
augment() method that does not accept a newdata argument, it will use the
default value for the data argument.

Details
This tidier currently only supports ivreg-classed objects outputted by the AER package. The ivreg
package also outputs objects of class ivreg, and will be supported in a later release.

Value
A tibble::tibble() with columns:

.fitted Fitted or predicted value.
.resid The difference between observed and fitted values.

See Also
augment(), AER::ivreg()

Other ivreg tidiers: glance.ivreg(), tidy.ivreg()

Examples

if (requireNamespace("AER", quietly = TRUE)) {

library(AER)

data("CigarettesSW", package = "AER")

ivr <- ivreg(
  log(packs) ~ income | population,
  data = CigarettesSW,
  subset = year == "1995"
)

summary(ivr)

tidy(ivr)
tidy(ivr, conf.int = TRUE)
augment.kmeans

Augment data with information from an kmeans object

Description

Augment accepts a model object and a dataset and adds information about each observation in
the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in
the .resid column, and standard errors for the fitted values in a .se.fit column. New columns
always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user
passes data to the data argument, it must be exactly the data that was used to fit the model object.
Pass datasets to newdata to augment data that was not used during model fitting. This still requires
that at least all predictor variable columns used to fit the model are present. If the original outcome
variable used to fit the model is not included in newdata, then no .resid column will be included
in the output.

Augment will often behave differently depending on whether data or newdata is given. This is be-
cause there is often information associated with training observations (such as influences or related)
measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment( fit)
will return the augmented training data. In these cases, augment tries to reconstruct the original
data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the
passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do
not support matrix-columns. This means you should not specify a matrix of covariates in a model
formula during the original model fitting process, and that splines::ns(), stats::poly() and
survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly
passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but
make no guarantees about behavior when data is missing at this time.

Usage

## S3 method for class 'kmeans'
augment(x, data, ...)

tidy(ivr, conf.int = TRUE, instruments = TRUE)
augment(ivr)
augment(ivr, data = CigarettesSW)
augment(ivr, newdata = CigarettesSW)
glance(ivr)
Arguments

x  A kmeans object created by `stats::kmeans()`.
data  A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object x. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
...

Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

.cluster  Cluster assignment.

See Also

`augment()`, `stats::kmeans()`

Other kmeans tidiers: `glance.kmeans()`, `tidy.kmeans()`

Examples

```r
## Not run:
library(cluster)
library(dplyr)

library(modeldata)
data(hpc_data)

x <- hpc_data[, 2:5]
fit <- pam(x, k = 4)
tidy(fit)
glance(fit)
augment(fit, x)

## End(Not run)
```
Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it must be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related measures) that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

### Usage

```r
## S3 method for class 'lm'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  se.fit = FALSE,
  interval = c("none", "confidence", "prediction"),
  ...)
```

### Arguments

- `x` An `lm` object created by `stats::lm()`.
data  A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.

newdata  A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.

se_fit  Logical indicating whether or not a .se.fit column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to FALSE.

interval  Character indicating the type of confidence interval columns to be added to the augmented output. Passed on to predict() and defaults to "none".

...  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

When the modeling was performed with na.action = "na.omit" (as is the typical default), rows with NA in the initial data are omitted entirely from the augmented data frame. When the modeling was performed with na.action = "na.exclude", one should provide the original data as a second argument, at which point the augmented data will contain those rows (typically with NAs in place of the new columns). If the original data is not provided to augment() and na.action = "na.exclude", a warning is raised and the incomplete rows are dropped.

Some unusual lm objects, such as rlm from MASS, may omit .cooksd and .std.resid. gam from mgcv omits .sigma.

When newdata is supplied, only returns .fitted, .resid and .se.fit columns.

Value

A tibble::tibble() with columns:

- .cooksd  Cooks distance.
- .fitted  Fitted or predicted value.
- .hat     Diagonal of the hat matrix.
- .lower   Lower bound on interval for fitted values.
- .resid   The difference between observed and fitted values.
- .se.fit  Standard errors of fitted values.
- .sigma  Estimated residual standard deviation when corresponding observation is dropped from model.
augment.lm

.std.resid  Standardised residuals.
.upper     Upper bound on interval for fitted values.

See Also

stats::na.action
augment(), stats::predict.lm()

Other lm tidiers: augment.glm(), glance.glm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm.beta(), tidy.lm(), tidy.mlm(), tidy.summary.lm()

Examples

library(ggplot2)
library(dplyr)

mod <- lm(mpg ~ wt + qsec, data = mtcars)
tidy(mod)
glance(mod)

# coefficient plot
d <- tidy(mod, conf.int = TRUE)
ggplot(d, aes(estimate, term, xmin = conf.low, xmax = conf.high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0, lty = 4) +
  geom_errorbarh()

# Aside: There are tidy() and glance() methods for lm.summary objects too.
# This can be useful when you want to conserve memory by converting large lm
# objects into their leaner summary.lm equivalents.
s <- summary(mod)
tidy(s, conf.int = TRUE)
glance(s)
augment(mod)
augment(mod, mtcars, interval = "confidence")

# predict on new data
newdata <- head(mtcars) %>%
  mutate(wt = wt + 1)
augment(mod, newdata = newdata)

# ggplot2 example where we also construct 95% prediction interval
mod2 <- lm(mpg ~ wt, data = mtcars)  ## simpler bivariate model since we're plotting in 2D
au <- augment(mod2, newdata = newdata, interval = "prediction")
ggplot(au, aes(wt, mpg)) +
```r
ggplot(au, aes(.hat, .std.resid)) +
  geom_vline(size = 2, colour = "white", xintercept = 0) +
  geom_hline(size = 2, colour = "white", yintercept = 0) +
  geom_point() +
  geom_smooth(se = FALSE)
```

```r
# predict on new data without outcome variable. Output does not include .resid
newdata <- newdata %>%
  select(-mpg)
augment(mod, newdata = newdata)

au <- augment(mod, data = mtcars)

```r
ggplot(au, aes(.hat, .std.resid)) +
  geom_vline(size = 2, colour = "white", xintercept = 0) +
  geom_hline(size = 2, colour = "white", yintercept = 0) +
  geom_point() +
  geom_smooth(se = FALSE)
```

```r
# column-wise models
a <- matrix(rnorm(20), nrow = 10)
b <- a + rnorm(length(a))
result <- lm(b ~ a)
tidy(result)
```

---

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.
For convenience, many augment methods provide default data arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a `tibble::tibble` with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'lmRob'
augment(x, data = model.frame(x), newdata = NULL, ...)
```

Arguments

- `x`: A `lmRob` object returned from `robust::lmRob()`.
- `data`: A `base::data.frame` or `tibble::tibble` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
- `newdata`: A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

For tidiers for robust models from the `MASS` package see `tidy.rlm()`.

See Also

- `robust::lmRob()`
- Other robust tidiers: `glance.glmRob()`, `glance.lmRob()`, `tidy.glmRob()`, `tidy.lmRob()`
Examples

```r
if (requireNamespace("robust", quietly = TRUE)) {
  library(robust)
  m <- lmRob(mpg ~ wt, data = mtcars)

  tidy(m)
  augment(m)
  glance(m)
}
```

### augment.lmrob

Augment data with information from a(n) lmrob object

#### Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

#### Usage

```r
## S3 method for class 'lmrob'
augment(x, data = model.frame(x), newdata = NULL, se_fit = FALSE, ...)
```
Arguments

- **x**: A `lmrob` object returned from `robustbase::lmrob()`.
- **data**: A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
- **newdata**: A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
- **se_fit**: Logical indicating whether or not a `.se.fit` column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to `FALSE`.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

For tidiers for robust models from the **MASS** package see `tidy.rlm()`.

Value

A `tibble::tibble()` with columns:

- **.fitted**: Fitted or predicted value.
- **.resid**: The difference between observed and fitted values.

See Also

`robustbase::lmrob()`

Other robustbase tidiers: `augment.glmrob()`, `glance.lmrob()`, `tidy.glmrob()`, `tidy.lmrob()`

Examples

```r
library(robustbase)
# From the robustbase::lmrob examples:
data(coleman)
set.seed(0)

m <- robustbase::lmrob(Y ~ ., data = coleman)
tidy(m)
```
augment(loess)

augment(m)
glance(m)

# From the robustbase::glmrob examples:
data(carrots)
Rfit <- glmrob(cbind(success, total - success) ~ logdose + block,
   family = binomial, data = carrots, method = "Mqle",
   control = glmrobMqle.control(tcc = 1.2)
)
tidy(Rfit)
augment(Rfit)

describe(loess)
Tidy a(n) loess object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'loess'
augment(x, data = model.frame(x), newdata = NULL, se_fit = FALSE, ...)

Arguments

x A loess objects returned by stats::loess.
data A base::data.frame or tibble::tibble containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. Do not pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
se_fit Logical indicating whether or not a .se.fit column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to FALSE.
...

Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.
When the modeling was performed with `na.action = "na.omit"` (as is the typical default), rows with NA in the initial data are omitted entirely from the augmented data frame. When the modeling was performed with `na.action = "na.exclude"`, one should provide the original data as a second argument, at which point the augmented data will contain those rows (typically with NAs in place of the new columns). If the original data is not provided to `augment()` and `na.action = "na.exclude"`, a warning is raised and the incomplete rows are dropped.

Note that `loess` objects by default will not predict on data outside of a bounding hypercube defined by the training data unless the original `loess` object was fit with `control = loess.control(surface = "direct")`). See `stats::predict.loess()` for details.

A `tibble::tibble()` with columns:

- `.fitted` Fitted or predicted value.
- `.resid` The difference between observed and fitted values.
- `.se.fit` Standard errors of fitted values.

See Also

`stats::na.action`, `augment()`, `stats::loess()`, `stats::predict.loess()`

Examples

```r
lo <- loess(
  mpg ~ hp + wt,
  mtcars,
  control = loess.control(surface = "direct")
)

augment(lo)

# with all columns of original data
augment(lo, mtcars)

# with a new dataset
augment(lo, newdata = head(mtcars))
```
Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

### Usage

```r
augment(x, data = NULL, ...) # S3 method for class 'Mclust'
```

### Arguments

- **x**: An Mclust object return from `mclust::Mclust()`.
- **data**: A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:

.class Predicted class.
.uncertainty The uncertainty associated with the classification. Equal to one minus the model class probability.

See Also

augment(), mclust::Mclust()

Other mclust tidiers: tidy.Mclust()

Examples

if (requireNamespace("mclust", quietly = TRUE)) {

library(dplyr)
library(mclust)
set.seed(27)

centers <- tibble::tibble(
  cluster = factor(1:3),
  num_points = c(100, 150, 50), # number points in each cluster
  x1 = c(5, 0, -3), # x1 coordinate of cluster center
  x2 = c(-1, 1, -2) # x2 coordinate of cluster center
)

points <- centers %>%
  mutate(
    x1 = purrr::map2(num_points, x1, rnorm),
    x2 = purrr::map2(num_points, x2, rnorm)
  ) %>%
  dplyr::select(-num_points, -cluster) %>%
  tidyr::unnest(c(x1, x2))

m <- mclust::Mclust(points)

tidy(m)
augment(m, points)
glance(m)
Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'mfx'
augment(
  x,
  data = model.frame(x$fit),
  newdata = NULL,
  type.predict = c("link", "response", "terms"),
  type.residuals = c("deviance", "pearson"),
  se.fit = FALSE,
  ...
)
```
## Arguments

### x

A `logitmfx`, `negbinmfx`, `poissonmfx`, or `probitmfx` object. (Note that `betamfx` objects receive their own set of tidiers.)
data A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. Do not pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.

newdata A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.

type.predict Passed to stats::predict.glm() type argument. Defaults to "link".

type.residuals Passed to stats::residuals.glm() and to stats::rstandard.glm() type arguments. Defaults to "deviance".

se_fit Logical indicating whether or not a .se.fit column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to FALSE.

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ... where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

This generic augment method wraps augment.glm() for applicable objects from the mfx package.

Value

A tibble::tibble() with columns:

- .cooksrd: Cooks distance.
- .fitted: Fitted or predicted value.
- .hat: Diagonal of the hat matrix.
- .resid: The difference between observed and fitted values.
- .se.fit: Standard errors of fitted values.
- .sigma: Estimated residual standard deviation when corresponding observation is dropped from model.
- .std.resid: Standardised residuals.

See Also

augment.glm(), mfx::logitmfx(), mfx::negbinmfx(), mfx::poissonmfx(), mfx::probitmfx() Other mfx tidiers: augment.betamfx(), glance.betamfx(), glance.mfx(), tidy.betamfx(), tidy.mfx()
if (requireNamespace("mfx", quietly = TRUE)) {

## Not run:
library(mfx)

## Get the marginal effects from a logit regression
mod_logmfx <- logitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)
tidy(mod_logmfx, conf.int = TRUE)

## Compare with the naive model coefficients of the same logit call (not run)
# tidy(glm(am ~ cyl + hp + wt, family = binomial, data = mtcars), conf.int = TRUE)

augment(mod_logmfx)
glance(mod_logmfx)

## Another example, this time using probit regression
mod пробмfx <- probitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)
tidy(mod пробмfx, conf.int = TRUE)
augment(mod пробмfx)
glance(mod пробмfx)

## End(Not run)
}

---

augment.mjoint  

Augment data with information from a(n) mjoint object

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.
For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'mjoint'
augment(x, data = x$data, ...)
```

Arguments

- `x` An mjoint object returned from joineRML::mjoint().
- `data` A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cox's distance for data passed to the data argument. These measures are only defined for the original training data.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

See joineRML::fitted.mjoint() and joineRML::residuals.mjoint() for more information on the difference between population-level and individual-level fitted values and residuals.

If fitting a joint model with a single longitudinal process, make sure you are using a named list to define the formula for the fixed and random effects of the longitudinal submodel.

Value

A tibble::tibble() with one row for each original observation with addition columns:

- `.fitted_j_0` population-level fitted values for the j-th longitudinal process
- `.fitted_j_1` individuals-level fitted values for the j-th longitudinal process
- `.resid_j_0` population-level residuals for the j-th longitudinal process
.resid_j_1 individual-level residuals for the j-th longitudinal process

Examples

if (requireNamespace("joineRML", quietly = TRUE)) {

## Not run:
# Fit a joint model with bivariate longitudinal outcomes
library(joineRML)
data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) &
  !is.na(heart.valve$log.lvmi) &
  heart.valve$num <= 50,]
fit <- mjoint(
  formLongFixed = list(
    "grad" = log.grad ~ time + sex + hs,
    "lvmi" = log.lvmi ~ time + sex
  ),
  formLongRandom = list(
    "grad" = ~ 1 | num,
    "lvmi" = ~ time | num
  ),
  formSurv = Surv(fuyrs, status) ~ age,
  data = hvd,
inits = list("gamma" = c(0.11, 1.51, 0.80)),
timeVar = "time"
)

# Extract the survival fixed effects
tidy(fit)

# Extract the longitudinal fixed effects
tidy(fit, component = "longitudinal")

# Extract the survival fixed effects with confidence intervals
tidy(fit, ci = TRUE)

# Extract the survival fixed effects with confidence intervals based
# on bootstrapped standard errors
bSE <- bootSE(fit, nboot = 5, safe.boot = TRUE)
tidy(fit, boot_se = bSE, ci = TRUE)

# Augment original data with fitted longitudinal values and residuals
hvd2 <- augment(fit)

# Extract model statistics
glance(fit)

## End(Not run)
}
Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related measures) that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'mlogit'
augment(x, data = x$model, ...)
```

Arguments

- `x` an object returned from mlogit::mlogit().
- `data` Not currently used
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.
Details

At the moment this only works on the estimation dataset. Need to set it up to predict on another dataset.

Value

A `tibble::tibble()` with columns:

- `.fitted` Fitted or predicted value.
- `.probability` Class probability of modal class.
- `.resid` The difference between observed and fitted values.

See Also

`augment()`

Other `mlogit` tidiers: `glance.mlogit()`, `tidy.mlogit()`

Examples

```r
if (requireNamespace("mlogit", quietly = TRUE)) {

  ## Not run:
  library(mlogit)
  data("Fishing", package = "mlogit")
  Fish <- dfidx(Fishing, varying = 2:9, shape = "wide", choice = "mode")
  m <- mlogit(mode ~ price + catch | income, data = Fish)

  tidy(m)
  augment(m)
  glance(m)

  ## End(Not run)
}
```

---

 augment.nlrq Tidy a(n) nlrq object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
## Usage

```r
## S3 method for class 'nlrq'
augment(x, data = NULL, newdata = NULL, ...)
```

### Arguments

- **x**  
  A `nlrq` object returned from `quantreg::nlrq()`.

- **data**  
  A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.

- **newdata**  
  A `base::data.frame` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.

- **...**  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

### See Also

- `augment()`, `quantreg::nlrq()`

### Other quantreg tidiers:

- `augment.rqs()`, `augment.rq()`, `glance.nlrq()`, `glance.rq()`, `tidy.nlrq()`, `tidy.rqs()`, `tidy.rq()`

### Examples

```r
n <- nls(mpg ~ k * e^wt, data = mtcars, start = list(k = 1, e = 2))

tidy(n)
augment(n)
glance(n)

library(ggplot2)
ggplot(augment(n), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))

newdata <- head(mtcars)
newdata$wt <- newdata$wt + 1
augment(n, newdata = newdata)
```
Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

## S3 method for class 'nls'
augment(x, data = NULL, newdata = NULL, ...)

Arguments

x
An nls object returned from stats::nls().

data
A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. Do not pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata

A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create \( x \). Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

augment.nls does not currently support confidence intervals due to a lack of support in `stats::predict.nls()`.

Value

A `tibble::tibble()` with columns:

- `.fitted` Fitted or predicted value.
- `.resid` The difference between observed and fitted values.

See Also

tidy, `stats::nls()`, `stats::predict.nls()`

Other nls tidiers: `glance.nls()`, `tidy.nls()`

Examples

```r
n <- nls(mpg ~ k * e^wt, data = mtcars, start = list(k = 1, e = 2))
tidy(n)
augment(n)
glance(n)

library(ggplot2)
ggplot(augment(n), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))

newdata <- head(mtcars)
newdata$wt <- newdata$wt + 1
augment(n, newdata = newdata)
```
augment.pam

Augment data with information from a(n) pam object

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'pam'
augment(x, data = NULL, ...)
```

Arguments

- `x` An pam object returned from cluster::pam()
- `data` A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. Do not pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

### Value

A `tibble::tibble()` with columns:

- `.cluster` Cluster assignment.
- `.fitted` Fitted or predicted value.
- `.resid` The difference between observed and fitted values.

### See Also

`augment()`, `cluster::pam()`

Other pam tidiers: `glance.pam()`, `tidy.pam()`

### Examples

```r
## Not run:
library(dplyr)
library(ggplot2)
library(cluster)
library(modeldata)
data(hpc_data)

x <- hpc_data[, 2:5]
p <- pam(x, k = 4)

tidy(p)
glance(p)
augment(p, x)

augment(p, x) %>%
  ggplot(aes(compounds, input_fields)) +
  geom_point(aes(color = .cluster)) +
  geom_text(aes(label = cluster), data = tidy(p), size = 10)

## End(Not run)
```
**augment.plm**

*Augment data with information from a(n) plm object*

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it must be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, `augment` tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a `tibble::tibble` with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

**Usage**

```r
## S3 method for class 'plm'
augment(x, data = model.frame(x), ...)  
```

**Arguments**

- `x`  
  A `plm` objected returned by `plm::plm()`.

- `data`  
  A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
... Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in . . . , where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with columns:

- `.fitted` Fitted or predicted value.
- `.resid` The difference between observed and fitted values.

See Also

`augment()`, `plm::plm()`

Other plm tidiers: `glance.plm()`, `tidy.plm()`

Examples

```r
if (requireNamespace("plm", quietly = TRUE)) {

library(plm)

data("Produc", package = "plm")
zz <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
    data = Produc, index = c("state", "year")
)
summary(zz)

 tidy(zz)
 tidy(zz, conf.int = TRUE)
 tidy(zz, conf.int = TRUE, conf.level = 0.9)

 augment(zz)
 glance(zz)
}
```
Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'polca'
augment(x, data = NULL, ...)
```

Arguments

- `x` A polca object returned from R::polca().
- `data` A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Details**

If the `data` argument is given, those columns are included in the output (only rows for which predictions could be made). Otherwise, the `y` element of the `poLCA` object, which contains the manifest variables used to fit the model, are used, along with any covariates, if present, in `x`.

Note that while the probability of all the classes (not just the predicted modal class) can be found in the `posterior` element, these are not included in the augmented output.

**Value**

A `tibble::tibble()` with columns:

- `.class`: Predicted class.
- `.probability`: Class probability of modal class.

**See Also**

`augment()`, `poLCA::poLCA()`

Other `poLCA` tidiers: `glance.poLCA()`, `tidy.poLCA()`

**Examples**

```r
if (requireNamespace("poLCA", quietly = TRUE)) {

library(poLCA)
library(dplyr)

data(values)
f <- cbind(A, B, C, D) ~ 1
M1 <- poLCA(f, values, nclass = 2, verbose = FALSE)

M1
tidy(M1)
augment(M1)
glance(M1)

library(ggplot2)

ggplot(tidy(M1), aes(factor(class), estimate, fill = factor(outcome))) +
  geom_bar(stat = "identity", width = 1) +
  facet_wrap(~variable)
  ## Three-class model with a single covariate.
```
data(election)
f2a <- cbind(
    MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG,
    MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB
) ~ PARTY
nes2a <- poLCA(f2a, election, nclass = 3, nrep = 5, verbose = FALSE)

td <- tidy(nes2a)

# show

ggplot(td, aes(outcome, estimate, color = factor(class), group = class)) +
    geom_line() +
    facet_wrap(~variable, nrow = 2) +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))

au <- augment(nes2a)

au

count(au, .class)

# if the original data is provided, it leads to NAs in new columns
# for rows that weren't predicted
au2 <- augment(nes2a, data = election)
au2

dim(au2)

---

**augment.polr**  
*Augment data with information from a(n) polr object*

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.
For convenience, many augment methods provide default data arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a `tibble::tibble` with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'polr'
augment(
  x,
  data = model.frame(x),
  newdata = NULL,
  type.predict = c("class"),
  ...
)
```

Arguments

- `x` A `polr` object returned from `MASS::polr()`.
- `data` A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. Do not pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
- `newdata` A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
- `type.predict` Which type of prediction to compute, passed to `MASS::predict.polr()`. Only supports "class" at the moment.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a newdata argument, it will use the default value for the data argument.

See Also

tidy(), `MASS::polr()`
Other ordinal tidiers: augment.clm(), glance.clmm(), glance.clm(), glance.polr(), glance.svyolr(), tidy.clmm(), tidy.clm(), tidy.polr(), tidy.svyolr()

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {
  library(MASS)
  fit <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
  tidy(fit, exponentiate = TRUE, conf.int = TRUE)
  glance(fit)
  augment(fit, type.predict = "class")
  fit2 <- polr(factor(gear) ~ am + mpg + qsec, data = mtcars)
  tidy(fit, p.values = TRUE)
}
```

__augment.prcomp__

Augment data with information from a(n) prcomp object

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model
formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'prcomp'
augment(x, data = NULL, newdata, ...)
```

Arguments

- **x**: A `prcomp` object returned by `stats::prcomp()`.
- **data**: A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
- **newdata**: A `base::data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble` containing the original data along with additional columns containing each observation’s projection into PCA space.

See Also

`stats::prcomp()`, `svd_tidiers`

Other svd tidiers: `tidy.prcomp()`, `tidy_irlba()`, `tidy_svd()`
**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `fitted` column, residuals in the `resid` column, and standard errors for the fitted values in a `se.fit` column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

**Usage**

```r
## S3 method for class 'rlm'
augment(x, data = model.frame(x), newdata = NULL, se_fit = FALSE, ...)
```

**Arguments**

- `x` An `rlm` object returned by `MASS::rlm()`.
- `data` A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object x. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. Do not pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata  A base::data.frame() or tibble::tibble() containing all the original predictors used to create \( x \). Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.

se_fit  Logical indicating whether or not a .se.fit column should be added to the augmented output. For some models, this calculation can be somewhat time-consuming. Defaults to FALSE.

...  Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value  A tibble::tibble() with columns:

- .fitted  Fitted or predicted value.
- .hat  Diagonal of the hat matrix.
- .resid  The difference between observed and fitted values.
- .se.fit  Standard errors of fitted values.
- .sigma  Estimated residual standard deviation when corresponding observation is dropped from model.

See Also  MASS::rlm()

Other rlm tidiers: glance.rlm(), tidy.rlm()

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {

library(MASS)

r <- rlm(stack.loss ~ ., stackloss)

tidy(r)
augment(r)
.glance(r)

}
```
Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines:ns(), stats:poly() and survival:Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

### Usage

```r
## S3 method for class 'rma'
augment(x, interval = c("prediction", "confidence"), ...)
```

### Arguments

- **x**: An rma object such as those created by metafor::rma(), metafor::rma.uni(), metafor::rma.glmm(), metafor::rma.mh(), metafor::rma.mv(), or metafor::rma.peto().
- **interval**: For rma.mv models, should prediction intervals ("prediction", default) or confidence intervals ("confidence") intervals be returned? For rma.uni models, prediction intervals are always returned. For rma.mh and rma.peto models, confidence intervals are always returned.
Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:

- .fitted: Fitted or predicted value.
- .lower: Lower bound on interval for fitted values.
- .moderator: In meta-analysis, the moderators used to calculate the predicted values.
- .moderator.level: In meta-analysis, the level of the moderators used to calculate the predicted values.
- .resid: The difference between observed and fitted values.
- .se.fit: Standard errors of fitted values.
- .upper: Upper bound on interval for fitted values.
- .observed: The observed values for the individual studies

Examples

library(metafor)

df <- escalc(
  measure = "RR",
  ai = tpos,
  bi = tneg,
  ci = cpos,
  di = cneg,
  data = dat.bcg
)

meta_analysis <- rma(yi, vi, data = df, method = "EB")

augment(meta_analysis)
Augment data with information from a(n) rq object

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

**Usage**

```r
## S3 method for class 'rq'
augment(x, data = model.frame(x), newdata = NULL, ...)
```

**Arguments**

- **x**: An rq object returned from quantreg::rq().
- **data**: A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata  A base::data.frame() or tibble::tibble() containing all the original predictors used to create \( x \). Defaults to NULL, indicating that nothing has been passed to \( \text{newdata} \). If \( \text{newdata} \) is specified, the \( \text{data} \) argument will be ignored.

...  Arguments passed on to quantreg::predict.rq

object  object of class rq or rqs or rq.process produced by \( \text{rq} \)

interval  type of interval desired: default is 'none', when set to 'confidence' the function returns a matrix predictions with point predictions for each of the 'newdata' points as well as lower and upper confidence limits.

level  convergence probability for the 'confidence' intervals.

type  For predict.rq, the method for 'confidence' intervals, if desired. If 'percentile' then one of the bootstrap methods is used to generate percentile intervals for each prediction, if 'direct' then a version of the Portnoy and Zhou (1998) method is used, and otherwise an estimated covariance matrix for the parameter estimates is used. Further arguments to determine the choice of bootstrap method or covariance matrix estimate can be passed via the ... argument. For predict.rqs and predict.rq.process when stepfun = TRUE, type is "Qhat", "Fhat" or "fhat" depending on whether the user would like to have estimates of the conditional quantile, distribution or density functions respectively. As noted below the two former estimates can be monotonized with the function rearrange. When the "fhat" option is invoked, a list of conditional density functions is returned based on Silverman's adaptive kernel method as implemented in akj and approxfun.

na.action  function determining what should be done with missing values in 'newdata'. The default is to predict 'NA'.

Details

Depending on the arguments passed on to predict.rq via ..., a confidence interval is also calculated on the fitted values resulting in columns \( \text{.lower} \) and \( \text{.upper} \). Does not provide confidence intervals when data is specified via the \( \text{newdata} \) argument.

Value

A tibble::tibble() with columns:

\( .\text{fitted} \)  Fitted or predicted value.

\( .\text{resid} \)  The difference between observed and fitted values.

\( .\text{tau} \)  Quantile.

See Also

augment, quantreg::rq(), quantreg::predict.rq()

Other quantreg tidiers:  augment.nlrq(), augment.rqs(), glance.nlrq(), glance.rq(), tidy.nlrq(), tidy.rqs(), tidy.rq()
Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

### Usage

```r
# S3 method for class 'rqs'
augment(x, data = model.frame(x), newdata, ...)
```

### Arguments

- **x**
  - An rqs object returned from quantreg::rq().

- **data**
  - A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
Augment data with information from a(n) spatialreg object

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.
Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'sarlm'
augment(x, data = x$X, ...)  
```

Arguments

- `x`:
  An object returned from spatialreg::lagsarlm() or spatialreg::errorsarlm().

- `data`:
  Ignored, but included for internal consistency. See the details below.

- `...`:
  Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in . . . , where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

The predict method for sarlm objects assumes that the response is known. See ?predict.sarlm for more discussion. As a result, since the original data can be recovered from the fit object, this method currently does not take in data or newdata arguments.

Value

A tibble with columns:
augment.smooth.spline

.Tidy a(n) smooth.spline object

.fitted Fitted or predicted value.
.resid The difference between observed and fitted values.

See Also

augment()

Other spatialreg tidiers: glance.sarlm(), tidy.sarlm()

Examples

## Not run:
library(spatialreg)
data(oldcol, package = "spdep")
listw <- spdep::nb2listw(COL.nb, style = "W")

crime_sar <-
lagsarlm(CRIME ~ INC + HOVAL,
data = COL.OLD,
listw = listw,
method = "eigen")
tidy(crime_sar)
tidy(crime_sar, conf.int = TRUE)

glance(crime_sar)
augment(crime_sar)

crime_sem <- errorsarlm(CRIME ~ INC + HOVAL, data = COL.OLD, listw)
tidy(crime_sem)
tidy(crime_sem, conf.int = TRUE)

glance(crime_sem)
augment(crime_sem)

crime_sac <- sacsarlm(CRIME ~ INC + HOVAL, data = COL.OLD, listw)
tidy(crime_sac)
tidy(crime_sac, conf.int = TRUE)

glance(crime_sac)
augment(crime_sac)

## End(Not run)
Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'smooth.spline'
augment(x, data = x$data, ...)  
```

Arguments

- `x`  
  A `smooth.spline` object returned from `stats::smooth.spline()`.
- `data`  
  A `base::data.frame` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
- `...`  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with columns:

- `.fitted`  
  Fitted or predicted value.
- `.resid`  
  The difference between observed and fitted values.

See Also

- `augment()`, `stats::smooth.spline()`, `stats::predict.smooth.spline()`

Other smoothing spline tidiers: `glance.smooth.spline()`

Examples

```r
spl <- smooth.spline(mtcars$wt, mtcars$mpg, df = 4)
augment(spl, mtcars)
augment(spl) # calls original columns x and y
library(ggplot2)
ggplot(augment(spl, mtcars), aes(wt, mpg)) +
```
Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object. Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

```r
## S3 method for class 'speedlm'
augment(x, data = model.frame(x), newdata = NULL, ...)
```

Arguments

- **x**: A speedlm object returned from speedglm::speedlm().
- **data**: A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata

A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:

.fitted Fitted or predicted value.
.resid The difference between observed and fitted values.

See Also

speedglm::speedlm()

Other speedlm tidiers: glance.speedglm(), glance.speedlm(), tidy.speedglm(), tidy.speedlm()

Examples

if (requireNamespace("speedglm", quietly = TRUE)) {

mod <- speedglm::speedlm(mpg ~ wt + qsec, data = mtcars, fitted = TRUE)

tidy(mod)
glance(mod)
augment(mod)
}

Augment data with information from a(n) stl object

Description

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the .fitted column, residuals in the .resid column, and standard errors for the fitted values in a .se.fit column. New columns always begin with a . prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it must be exactly the data that was used to fit the model object.
Pass datasets to newdata to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in newdata, then no .resid column will be included in the output.

Augment will often behave differently depending on whether data or newdata is given. This is because there is often information associated with training observations (such as influences or related) measures that is not meaningfully defined for new observations.

For convenience, many augment methods provide default data arguments, so that augment(fit) will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a tibble::tibble with the same number of rows as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

We are in the process of defining behaviors for models fit with various na.action arguments, but make no guarantees about behavior when data is missing at this time.

Usage

## S3 method for class 'stl'
augment(x, data = NULL, weights = TRUE, ...)

Arguments

- **x**: An stl object returned from stats::stl().
- **data**: Ignored, included for consistency with the augment generic signature only.
- **weights**: Logical indicating whether or not to include the robust weights in the output.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble with one row for each observation in the original times series:

- **.seasonal**: The seasonal component of the decomposition.
- **.trend**: The trend component of the decomposition.
- **.remainder**: The remainder, or "random" component of the decomposition.
- **.weight**: The final robust weights, if requested.
- **.seasadj**: The seasonally adjusted (or "deseasonalised") series.
See Also

`augment()`, `stats::stl()`

Other decompose tidiers: `augment.decomposed.ts()`

---

**Description**

Augment accepts a model object and a dataset and adds information about each observation in the dataset. Most commonly, this includes predicted values in the `.fitted` column, residuals in the `.resid` column, and standard errors for the fitted values in a `.se.fit` column. New columns always begin with a `.` prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it must be exactly the data that was used to fit the model object. Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that at least all predictor variable columns used to fit the model are present. If the original outcome variable used to fit the model is not included in `newdata`, then no `.resid` column will be included in the output.

Augment will often behave differently depending on whether `data` or `newdata` is given. This is because there is often information associated with training observations (such as influences or related measures) that is not meaningfully defined for new observations.

For convenience, many augment methods provide default `data` arguments, so that `augment(fit)` will return the augmented training data. In these cases, augment tries to reconstruct the original data based on the model object with varying degrees of success.

The augmented dataset is always returned as a `tibble::tibble` with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a `tibble`. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model formula during the original model fitting process, and that `splines::ns()`, `stats::poly()` and `survival::Surv()` objects are not supported in input data. If you encounter errors, try explicitly passing a `tibble`, or fitting the original model on data in a `tibble`.

We are in the process of defining behaviors for models fit with various `na.action` arguments, but make no guarantees about behavior when data is missing at this time.

**Usage**

```r
## S3 method for class 'survreg'
augment(
  x,
  data = NULL,
  newdata = NULL,
  type.predict = "response",
  type.residuals = "response",
  ...
)
```
Arguments

x An survreg object returned from survival::survreg().
data A base::data.frame or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. Do not pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.

newdata A base::data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict Character indicating type of prediction to use. Passed to the type argument of the stats::predict() generic. Allowed arguments vary with model class, so be sure to read the predict.my_class documentation.
type.residuals Character indicating type of residuals to use. Passed to the type argument of stats::residuals() generic. Allowed arguments vary with model class, so be sure to read the residuals.my_class documentation.

Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:

.fitted Fitted or predicted value.
.resid The difference between observed and fitted values.
.se.fit Standard errors of fitted values.

See Also

augment(), survival::survreg()
Other survreg tidiers: glance.survreg(), tidy.survreg()
Other survival tidiers: augment.coxph(), glance.aareg(), glance.cch(), glance.coxph(),
glance.pyeas(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(),
tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyeas(), tidy.survdiff(), tidy.survexp(),
tidy.survfit(), tidy.survreg()

Examples

if (requireNamespace("survival", quietly = TRUE)) {
library(survival)

sr <- survreg(
  Surv(futime, fustat) ~ ecog.ps + rx, 
  ovarian, 
  dist = "exponential"
)

tidy(sr)
augment(sr, ovarian)
glance(sr)

# coefficient plot
td <- tidy(sr, conf.int = TRUE)
library(ggplot2)
ggplot(td, aes(estimate, term)) + 
geom_point() + 
geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) + 
geom_vline(xintercept = 0)

---

augment_columns: Add fitted values, residuals, and other common outputs to an augment call

**Description**

`augment_columns` is intended for use in the internals of augment methods only and is exported for developers extending the broom package. Please instead use `augment()` to appropriately make use of the functionality in `augment_columns()`.

**Usage**

```r
augment_columns(
  x, 
  data, 
  newdata = NULL, 
  type, 
  type.predict = type, 
  type.residuals = type, 
  se.fit = TRUE, 
  ... 
)
```
bootstrap

Arguments

- **x** a model
- **data** original data onto which columns should be added
- **newdata** new data to predict on, optional
- **type** Type of prediction and residuals to compute
- **type.predict** Type of prediction to compute; by default same as type
- **type.residuals** Type of residuals to compute; by default same as type
- **se.fit** Value to pass to predict's se.fit, or NULL for no value
- **...** extra arguments (not used)

Details

Note that, in the case that a residuals() or influence() generic is not implemented for the supplied model x, the function will fail quietly.

---

**bootstrap**

*Set up bootstrap replicates of a dplyr operation*

---

Description

The bootstrap() function is deprecated and will be removed from an upcoming release of broom. For tidy resampling, please use the rsample package instead. Functionality is no longer supported for this method.

Usage

`bootstrap(df, m, by_group = FALSE)`

Arguments

- **df** a data frame
- **m** number of bootstrap replicates to perform
- **by_group** If TRUE, then bootstrap within each group if df is a grouped tibble.

Details

This code originates from Hadley Wickham (with a few small corrections) here: https://github.com/tidyverse/dplyr/issues/269

See Also

Other deprecated: `confint_tidy()`, `data.frame_tidiers.finish_glance()`, `fix_data_frame()`, `summary_tidiers`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`
confint_tidy  

(Deprecated) Calculate confidence interval as a tidy data frame

Description

This function is now deprecated and will be removed from a future release of broom.

Usage

confint_tidy(x, conf.level = 0.95, func = stats::confint, ...)

Arguments

- **x**: a model object for which `confint()` can be calculated
- **conf.level**: confidence level
- **func**: A function to compute a confidence interval for `x`. Calling `func(x, level = conf.level, ...)` must return an object coercible to a tibble. This dataframe like object should have two columns corresponding to the lower and upper bounds on the confidence interval.
- **...**: extra arguments passed on to `confint`

Details

Return a confidence interval as a tidy data frame. This directly wraps the `confint()` function, but ensures it follows broom conventions: column names of `conf.low` and `conf.high`, and no row names.

confint_tidy

Value

A tibble with two columns: `conf.low` and `conf.high`.

See Also

Other deprecated: `bootstrap()`, `data.frame_tidiers`, `finish_glance()`, `fix_data_frame()`, `summary_tidiers`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`
data.frame_tidiers  Tidiers for data.frame objects

Description

Data frame tidiers are deprecated and will be removed from an upcoming release of broom.

Usage

```r
## S3 method for class 'data.frame'
tidy(x, ..., na.rm = TRUE, trim = 0.1)

## S3 method for class 'data.frame'
augment(x, data, ...)

## S3 method for class 'data.frame'
glance(x, ...)
```

Arguments

- `x`: A data.frame
- `...`: Additional arguments for other methods.
- `na.rm`: a logical value indicating whether NA values should be stripped before the computation proceeds.
- `trim`: the fraction (0 to 0.5) of observations to be trimmed from each end of x before the mean is computed. Passed to the `trim` argument of `mean`.
- `data`: data, not used

Details

These perform tidy summaries of data.frame objects. tidy produces summary statistics about each column, while glance simply reports the number of rows and columns. Note that augment.data.frame will throw an error.

Value

tidy.data.frame produces a data frame with one row per original column, containing summary statistics of each:

- `column`: name of original column
- `n`: Number of valid (non-NA) values
- `mean`: mean
- `sd`: standard deviation
- `median`: median
- `trimmed`: trimmed mean, with trim defaulting to 0.1
mad      median absolute deviation (from the median)
min      minimum value
max      maximum value
range    range
skew     skew
kurtosis kurtosis
se       standard error

glance returns a one-row data.frame with

nrow     number of rows
ncol     number of columns
complete.obs number of rows that have no missing values
na.fraction fraction of values across all rows and columns that are missing

Author(s)
David Robinson, Benjamin Nutter

Source
Skew and Kurtosis functions are adapted from implementations in the moments package:
Lukasz Komsta and Frederick Novomestky (2015). moments: Moments, cumulants, skewness,
kurtosis and related tests. R package version 0.14.
https://CRAN.R-project.org/package=moments

See Also
Other deprecated: bootstrap(), confint_tidy(), finish_glance(), fix_data_frame(), summary_tidiers,
tidy.density(), tidy.dist(), tidy.ftable(), tidy.numeric()

Examples

td <- tidy(mtcars)
td

glance(mtcars)

library(ggplot2)
# compare mean and standard deviation
ggplot(td, aes(mean, sd)) + geom_point() +
  geom_text(aes(label = column), hjust = 1, vjust = 1) +
  scale_x_log10() + scale_y_log10() + geom_abline()
durbinWatsonTest_tidiers

Tidy/glance a(n) durbinWatsonTest object

Description

For models that have only a single component, the tidy() and glance() methods are identical. Please see the documentation for both of those methods.

Usage

## S3 method for class 'durbinWatsonTest'
tidy(x, ...)

## S3 method for class 'durbinWatsonTest'
glance(x, ...)

Arguments

x
An object of class durbinWatsonTest created by a call to car::durbinWatsonTest().

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:

alternative Alternative hypothesis (character).
autocorrelation Autocorrelation.
p.value The two-sided p-value associated with the observed statistic.
statistic Test statistic for Durbin-Watson test.
method Always 'Durbin-Watson Test'.

See Also

tidy(), glance(), car::durbinWatsonTest()

Other car tidiers: leveneTest_tidiers
Examples

```r
if (requireNamespace("car", quietly = TRUE)) {
  dw <- car::durbinWatsonTest(lm(mpg ~ wt, data = mtcars))
  tidy(dw)
  glance(dw) # same output for all durbinWatsonTests
}
```

---

**finish_glance**

*(Deprecated) Add logLik, AIC, BIC, and other common measurements to a glance of a prediction*

**Description**

This function is now deprecated in favor of using custom logic and the appropriate `nobs()` method.

**Usage**

```r
finish_glance(ret, x)
```

**Arguments**

- `ret` a one-row data frame (a partially complete glance)
- `x` the prediction model

**Value**

A one-row data frame with additional columns added, such as

- `logLik` log likelihoods
- `AIC` Akaike Information Criterion
- `BIC` Bayesian Information Criterion
- `deviance` deviance
- `df.residual` residual degrees of freedom

**See Also**

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, fix_data_frame(), summary_tidiers, tidy.density(), tidy.dist(), tidy.ftable(), tidy.numeric()
**fix_data_frame**

Ensure an object is a data frame, with rownames moved into a column

---

**Description**

This function is deprecated as of broom 0.7.0 and will be removed from a future release. Please see `tibble::as_tibble`.

**Usage**

```r
fix_data_frame(x, newnames = NULL, newcol = "term")
```

**Arguments**

- `x`: a data.frame or matrix
- `newnames`: new column names, not including the rownames
- `newcol`: the name of the new rownames column

**Value**

a data.frame, with rownames moved into a column and new column names assigned

**See Also**

Other deprecated: `bootstrap()`, `confint_tidy()`, `data.frame_tidiers`, `finish_glance()`, `summary_tidiers`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`

---

**glance.aareg**

Glance at a(n) aareg object

---

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.
Usage

## S3 method for class 'aareg'
glance(x, ...)

Arguments

x
  An aareg object returned from survival::aareg().

...  Additional arguments. Not used. Needed to match generic signature only. Cau-
       tionary note: Misspelled arguments will be absorbed in ..., where they will be
       ignored. If the misspelled argument has a default value, the default value will
       be used. For example, if you pass conf.lvel = 0.9, all computation will proceed
       using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an
       augment() method that does not accept a newdata argument, it will use the
       default value for the data argument.

Value

A tibble::tibble() with exactly one row and columns:

df  Degrees of freedom used by the model.
nobs Number of observations used.
p.value P-value corresponding to the test statistic.
statistic Test statistic.

See Also

glance(), survival::aareg()

Other aareg tidiers: tidy.aareg()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.cch(), glance.coxph(),
glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(),
tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(),
tidy.survfit(), tidy.survreg()

Examples

if (requireNamespace("survival", quietly = TRUE)) {

  library(survival)

  afit <- aareg(
    Surv(time, status) ~ age + sex + ph.ecog,
    data = lung,
    dfbeta = TRUE
  )

  tidy(afit)
}
Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

### Usage

```r
## S3 method for class 'aov'

glance(x, ...)
```

### Arguments

- **x**: An `aov` object, such as those created by `stats::aov()`.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

### Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC**: Akaike's Information Criterion for the model.
- **BIC**: Bayesian Information Criterion for the model.
- **deviance**: Deviance of the model.
- **logLik**: The log-likelihood of the model. [`stats::logLik()`] may be a useful reference.
- **nobs**: Number of observations used.
**Note**

Note that tidy.aov() now contains the numerator and denominator degrees of freedom, which were included in the output of glance.aov() in some previous versions of the package.

**See Also**

glance()

Other anova tidiers: tidy.TukeyHSD(), tidy.anova(), tidy.aovlist(), tidy.aov(), tidy.manova()

**Examples**

```r
a <- aov(mpg ~ wt + qsec + disp, mtcars)
tidy(a)
```

---

**Description**

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

**Usage**

```r
## S3 method for class 'Arima'
glance(x, ...)
```

**Arguments**

- `x` An object of class Arima created by stats::arima().
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC**: Akaike’s Information Criterion for the model.
- **BIC**: Bayesian Information Criterion for the model.
- **logLik**: The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- **nobs**: Number of observations used.
- **sigma**: Estimated standard error of the residuals.

See Also

- `stats::arima()`
- Other Arima tidiers: `tidy.Arima()`

Examples

```r
fit <- arima(lh, order = c(1, 0, 0))
tidy(fit)
```

Usage

```r
## S3 method for class 'betamfx'
glance(x, ...)
```
Arguments

x  A betamfx object.
...  Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.1vel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

This glance method wraps glance.betareg() for mfx::betamfx() objects.

Value

A tibble::tibble() with exactly one row and columns:

- AIC  Akaike's Information Criterion for the model.
- BIC  Bayesian Information Criterion for the model.
- df.null  Degrees of freedom used by the null model.
- df.residual  Residual degrees of freedom.
- logLik  The log-likelihood of the model. [stats::logLik()] may be a useful reference.
- nobs  Number of observations used.
- pseudo.r.squared  Like the R squared statistic, but for situations when the R squared statistic isn’t defined.

See Also

glance.betareg(), mfx::betamfx()
Other mfx tidiers: augment.betamfx(), augment.mfx(), glance.mfx(), tidy.betamfx(), tidy.mfx()

Examples

```r
## Not run:
library(mfx)

## Simulate some data
set.seed(12345)
n = 1000
x = rnorm(n)

## Beta outcome
y = rbeta(n, shape1 = plogis(1 + 0.5 * x), shape2 = (abs(0.2*x)))
## Use Smithson and Verkuilen correction
y = (y*(n-1)+0.5)/n
```
d = data.frame(y,x)
mod_betamfx = betamfx(y ~ x | x, data = d)

 tidy(mod_betamfx, conf.int = TRUE)

## Compare with the naive model coefficients of the equivalent betareg call (not run)
# tidy(betamfx(y ~ x | x, data = d), conf.int = TRUE)

augment(mod_betamfx)
glance(mod_betamfx)

## End(Not run)

---

**glance.betareg**

*Glance at a(n) betareg object*

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

**Usage**

```r
## S3 method for class 'betareg'
glance(x, ...)
```

**Arguments**

- `x` A `betareg` object produced by a call to `betareg::betareg()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC**: Akaike's Information Criterion for the model.
- **BIC**: Bayesian Information Criterion for the model.
- **df.null**: Degrees of freedom used by the null model.
- **df.residual**: Residual degrees of freedom.
- **logLik**: The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- **nobs**: Number of observations used.
- **pseudo.r.squared**: Like the R squared statistic, but for situations when the R squared statistic isn’t defined.

See Also

- `glance()`, `betareg::betareg()`

Examples

```r
if (requireNamespace("betareg", quietly = TRUE)) {

library(betareg)
data("GasolineYield", package = "betareg")

mod <- betareg(yield ~ batch + temp, data = GasolineYield)

mod
tidy(mod)
tidy(mod, conf.int = TRUE)
tidy(mod, conf.int = TRUE, conf.level = .99)

augment(mod)

glance(mod)
}
```

---

`glance.biglm`  
*Glance at a(n) biglm object*
Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'biglm'

glance(x, ...)
```

Arguments

- `x`  A `biglm` object created by a call to `biglm::biglm()` or `biglm::bigglm()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC**  Akaike’s Information Criterion for the model.
- **deviance**  Deviance of the model.
- **df.residual**  Residual degrees of freedom.
- **nobs**  Number of observations used.
- **r.squared**  R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.

See Also

- `glance()`, `biglm::biglm()`, `biglm::bigglm()`
- Other biglm tidiers: `tidy.biglm()`
### Examples

```r
## Not run:
library(biglm)

bfit <- biglm(mpg ~ wt + disp, mtcars)
tidy(bfit)
tidy(bfit, conf.int = TRUE)
tidy(bfit, conf.int = TRUE, conf.level = .9)

glance(bfit)

# bigglm: logistic regression
bgfit <- bigglm(am ~ mpg, mtcars, family = binomial())

tidy(bgfit)
tidy(bgfit, exponentiate = TRUE)
tidy(bgfit, conf.int = TRUE)
tidy(bgfit, conf.int = TRUE, conf.level = .9)
tidy(bgfit, conf.int = TRUE, conf.level = .9, exponentiate = TRUE)

glance(bgfit)

## End(Not run)
```

---

**glance.binDesign**

**Glance at a(n) binDesign object**

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

**Usage**

```r
## S3 method for class 'binDesign'
glance(x, ...)
```
Arguments

- `x` A `binGroup::binDesign` object.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `power` Power achieved by the analysis.
- `n` Sample size used to achieve this power.
- `power.reached` Whether the desired power was reached.
- `maxit` Number of iterations performed.

See Also

- `glance()`, `binGroup::binDesign()`
- Other bingroup tidiers: `tidy.binDesign()`, `tidy.binWidth()`

Examples

```r
if (requireNamespace("binGroup", quietly = TRUE)) {
  library(binGroup)
  des <- binDesign(
    nmax = 300, delta = 0.06,
    p.hyp = 0.1, power = .8
  )

  glance(des)
  tidy(des)

  library(ggplot2)
  ggplot(tidy(des), aes(n, power)) +
    geom_line()
}
```
Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'cch'
glance(x, ...)
```

Arguments

- `x` An `cch` object returned from `survival::cch()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `iter` Iterations of algorithm/fitting procedure completed.
- `p.value` P-value corresponding to the test statistic.
- `rscore` Robust log-rank statistic
- `score` Score.
- `n` number of predictions
- `nevent` number of events
glance.clm

See Also

- `glance()`, `survival::cch()`
- Other cch tidiers: `glance.survfit()`, `tidy.cch()`
- Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {
  library(survival)
  # examples come from cch documentation
  subcoh <- nwtco$in.subcohort
  selccoh <- with(nwtco, rel == 1 | subcoh == 1)
  ccoh.data <- nwtco[selccoh,]
  ccoh.data$subcohort <- subcoh[selccoh]
  # central-lab histology
  ccoh.data$histol <- factor(ccoh.data$histol, labels = c("FH", "UH"))
  # tumour stage
  ccoh.data$stage <- factor(ccoh.data$stage, labels = c("I", "II", "III", "IV"))
  ccoh.data$age <- ccoh.data$age / 12 # Age in years
  fit.ccP <- cch(Surv(edrel, rel) ~ stage + histol + age, data = ccoh.data, subcoh = ~subcohort, id = ~seqno, cohort.size = 4028)
  tidy(fit.ccP)
  # coefficient plot
  library(ggplot2)
  ggplot(tidy(fit.ccP), aes(x = estimate, y = term)) + geom_point() + geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) + geom_vline(xintercept = 0)
}
```

---

**Glance at a(n) clm object**
Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'clm'

glance(x, ...)
```

Arguments

- `x`: A `clm` object returned from `ordinal::clm()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `AIC`: Akaike’s Information Criterion for the model.
- `BIC`: Bayesian Information Criterion for the model.
- `df.residual`: Residual degrees of freedom.
- `edf`: The effective degrees of freedom.
- `logLik`: The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- `nobs`: Number of observations used.

See Also

- `tidy.ordinal::clm()`

Other ordinal tidiers: `augment.clm()`, `augment.polr()`, `glance.clmm()`, `glance.polr()`, `glance.svyolr()`, `tidy.clmm()`, `tidy.clm()`, `tidy.polr()`, `tidy.svyolr()`
Examples

```r
if (requireNamespace("ordinal", quietly = TRUE)) {

library(ordinal)

fit <- clm(rating ~ temp * contact, data = wine)

tidy(fit)
tidy(fit, conf.int = TRUE, conf.level = 0.9)
tidy(fit, conf.int = TRUE, conf.type = "Wald", exponentiate = TRUE)

glance(fit)
augment(fit, type.predict = "prob")
augment(fit, type.predict = "class")

fit2 <- clm(rating ~ temp, nominal = ~contact, data = wine)
tidy(fit2)
glance(fit2)
}
```

---

**glance.clmm**  
*Glance at a(n) clmm object*

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

**Usage**

```r
## S3 method for class 'clmm'
glance(x, ...)
```
Arguments

- `x`: A `clmm` object returned from `ordinal::clmm()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `AIC`: Akaike’s Information Criterion for the model.
- `BIC`: Bayesian Information Criterion for the model.
- `edf`: The effective degrees of freedom.
- `logLik`: The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- `nobs`: Number of observations used.

See Also

- `tidy.ordinal::clmm()`
- Other ordinal tidiers: `augment.clm()`, `augment.polr()`, `glance.clm()`, `glance.polr()`, `glance.svyolr()`, `tidy.clmm()`, `tidy.clm()`, `tidy.polr()`, `tidy.svyolr()`

Examples

```r
if (requireNamespace("ordinal", quietly = TRUE)) {
  library(ordinal)
  fit <- clmm(rating ~ temp + contact + (1 | judge), data = wine)
  tidy(fit)
  tidy(fit, conf.int = TRUE, conf.level = 0.9)
  tidy(fit, conf.int = TRUE, exponentiate = TRUE)
  glance(fit)
  fit2 <- clmm(rating ~ temp + (1 | judge), nominal = ~contact, data = wine)
  tidy(fit2)
  glance(fit2)
}
```
Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'coeftest'
glance(x, ...)
```

Arguments

- `x`: A coeftest object returned from `lmtest::coeftest()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `adj.r.squared`: Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
- `AIC`: Akaike's Information Criterion for the model.
- `BIC`: Bayesian Information Criterion for the model.
- `deviance`: Deviance of the model.
- `df`: Degrees of freedom used by the model.
- `df.residual`: Residual degrees of freedom.
The log-likelihood of the model. \texttt{stats::logLik()} may be a useful reference.

- \texttt{nobs}
  Number of observations used.
- \texttt{p.value}
  P-value corresponding to the test statistic.
- \texttt{r.squared}
  R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
- \texttt{sigma}
  Estimated standard error of the residuals.
- \texttt{statistic}
  Test statistic.

\textbf{Note}

Because of the way that \texttt{lmtest::coeftest()} retains information about the underlying model object, the returned columns for \texttt{glance.coeftest()} will vary depending on the arguments. Specifically, four columns are returned regardless: "Loglik", "AIC", "BIC", and "nobs". Users can obtain additional columns (e.g. "r.squared", "df") by invoking the "save = TRUE" argument as part of \texttt{lmtest::coeftest()}. See examples.

As an aside, goodness-of-fit measures such as R-squared are unaffected by the presence of heteroskedasticity. For further discussion see, e.g. chapter 8.1 of Wooldridge (2016).

\textbf{References}


\textbf{See Also}

- \texttt{glance()}
- \texttt{lmtest::coeftest()}

\textbf{Examples}

```r
if (requireNamespace("lmtest", quietly = TRUE)) {
  library(lmtest)
  m <- lm(dist ~ speed, data = cars)
  coeftest(m)
  tidy(coeftest(m))
  tidy(coeftest(m, conf.int = TRUE))

  # A very common workflow is to combine \texttt{lmtest::coeftest} with alternate variance-covariance matrices via the sandwich package. The \texttt{lmtest} \texttt{tidiers} support this workflow too, enabling you to adjust the standard errors of your tidied models on the fly.
  library(sandwich)
  tidy(coeftest(m, vcov = vcovHC)) # "HC3" (default) robust SEs
  tidy(coeftest(m, vcov = vcovHC, type = "HC2")) # "HC2" robust SEs
  tidy(coeftest(m, vcov = NeweyWest)) # N-W HAC robust SEs
}
# The columns of the returned tibble for glance.coefxtest() will vary
# depending on whether the coeftest object retains the underlying model.
# Users can control this with the "save = TRUE" argument of coeftest().
glance(coeftest(m))
glance(coeftest(m, save = TRUE)) # More columns

---

glance.coxph

Glance a(n) coxph object

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'coxph'
glance(x, ...)
```

Arguments

- `x` A coxph object returned from `survival::coxph()`.

- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC**: Akaike’s Information Criterion for the model.
- **BIC**: Bayesian Information Criterion for the model.
- **logLik**: The log-likelihood of the model. [stats::logLik()] may be a useful reference.
- **n**: The total number of observations.
- **nevent**: Number of events.
- **nobs**: Number of observations used.

See `survival::coxph.object` for additional column descriptions.

See Also

`glance()`, `survival::coxph()`

Other `coxph` tidiers: `augment.coxph()`, `tidy.coxph()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {

  library(survival)

  cfit <- coxph(Surv(time, status) ~ age + sex, lung)

  tidy(cfit)
  tidy(cfit, exponentiate = TRUE)

  lp <- augment(cfit, lung)
  risks <- augment(cfit, lung, type.predict = "risk")
  expected <- augment(cfit, lung, type.predict = "expected")

  glance(cfit)

  # also works on clogit models
  resp <- levels(logan$occupation)
  n <- nrow(logan)
  indx <- rep(1:n, length(resp))
  logan2 <- data.frame(
    logan[indx, ],
    id = indx,
    tocc = factor(rep(resp, each = n))
  )
}
```
log2$case <- (log2$occupation == log2$tocc)

c1 <- clogit(case ~ tocc + tocc:education + strata(id), log2)
tidy(c1)
glance(c1)

library(ggplot2)

ggplot(lp, aes(age, .fitted, color = sex)) +
  geom_point()
ggplot(risks, aes(age, .fitted, color = sex)) +
  geom_point()
ggplot(expected, aes(time, .fitted, color = sex)) +
  geom_point()

---

glance.crr  

**Glance at a(n) crr object**

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

**Usage**

```r
## S3 method for class 'crr'

glance(x, ...)
```

**Arguments**

- `x` A *crr* object returned from `cmprsk::crr()`. 
Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in . . ., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with exactly one row and columns:

- converged: Logical indicating if the model fitting procedure was succesful and converged.
- df: Degrees of freedom used by the model.
- logLik: The log-likelihood of the model. [stats::logLik()] may be a useful reference.
- nobs: Number of observations used.
- statistic: Test statistic.

See Also

- glance(), cmprsk::crr()

Other cmprsk tidiers: tidy.crr()

Examples

if (requireNamespace("cmprsk", quietly = TRUE)) {

library(cmprsk)
lrf_time <- rexp(100) # time to loco-regional failure (lrf)
lrf_event <- sample(0:2, 100, replace = TRUE)
trt <- sample(0:1, 100, replace = TRUE)
strt <- sample(1:2, 100, replace = TRUE)
x <- crr(lrf_time, lrf_event, cbind(trt, strt))
tidy(x, conf.int = TRUE)
glance(x)
}

---

**glance.cv.glmnet**  
Glance at a(n) cv.glmnet object
Description

Glance accepts a model object and returns a \texttt{tibble::tibble()} with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as \texttt{NA}.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an \texttt{NA} of the appropriate type.

Usage

\begin{verbatim}
## S3 method for class 'cv.glmnet'
glance(x, ...)
\end{verbatim}

Arguments

\begin{itemize}
  \item \textbf{x} \hspace{1cm} A \texttt{cv.glmnet} object returned from \texttt{glmnet::cv.glmnet()}.
  \item \textbf{...} \hspace{1cm} Additional arguments. Not used. Needed to match generic signature only. \textbf{Cautionary note:} Misspelled arguments will be absorbed in \texttt{...}, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass \texttt{conf.1vel = 0.9}, all computation will proceed using \texttt{conf.1vel = 0.95}. Additionally, if you pass \texttt{newdata = my_tibble} to an \texttt{augment()} method that does not accept a \texttt{newdata} argument, it will use the default value for the \texttt{data} argument.
\end{itemize}

Value

A \texttt{tibble::tibble()} with exactly one row and columns:

\begin{itemize}
  \item \texttt{lambda.1se} \hspace{1cm} The value of the penalization parameter lambda that results in the sparsest model while remaining within one standard error of the minimum loss.
  \item \texttt{lambda.min} \hspace{1cm} The value of the penalization parameter lambda that achieved minimum loss as estimated by cross validation.
  \item \texttt{nobs} \hspace{1cm} Number of observations used.
\end{itemize}

See Also

\texttt{glance()}, \texttt{glmnet::cv.glmnet()}

Other glmnet tidiers: \texttt{glance.glmnet()}, \texttt{tidy.cv.glmnet()}, \texttt{tidy.glmnet()}
Examples

```r
if (requireNamespace("glmnet", quietly = TRUE)) {

  library(glmnet)
  set.seed(27)

  nobs <- 100
  nvar <- 50
  real <- 5

  x <- matrix(rnorm(nobs * nvar), nobs, nvar)
  beta <- c(rnorm(real, 0, 1), rep(0, nvar - real))
  y <- c(t(beta) %*% t(x)) + rnorm(nvar, sd = 3)

  cvfit1 <- cv.glmnet(x, y)
  tidy(cvfit1)
  glance(cvfit1)

  library(ggplot2)
  tidied_cv <- tidy(cvfit1)
  glance_cv <- glance(cvfit1)

  # plot of MSE as a function of lambda
  g <- ggplot(tidied_cv, aes(lambda, estimate)) +
    geom_line() +
    scale_x_log10()
  g

  # plot of MSE as a function of lambda with confidence ribbon
  g <- g + geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)
  g

  # plot of MSE as a function of lambda with confidence ribbon and choices
  # of minimum lambda marked
  g <- g +
    geom_vline(xintercept = glance_cv$lambda.min) +
    geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
  g

  # plot of number of zeros for each choice of lambda
  ggplot(tidied_cv, aes(lambda, nzero)) +
    geom_line() +
    scale_x_log10()

  # coefficient plot with min lambda shown
  tidied <- tidy(cvfit1$glmnet.fit)
  ggplot(tidied, aes(lambda, estimate, group = term)) +
    scale_x_log10() +
    geom_line()
```

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

## Usage

```r
## S3 method for class 'drc'

## S3 method for class 'drc'

glance(x, ...)
```

## Arguments

- `x` A drc object produced by a call to `drc::drm()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a newdata argument, it will use the default value for the data argument.

## Value

A `tibble::tibble()` with exactly one row and columns:

- `AIC` Akaike’s Information Criterion for the model.
- `BIC` Bayesian Information Criterion for the model.
- `df.residual` Residual degrees of freedom.
- `logLik` The log-likelihood of the model. [stats::logLik()] may be a useful reference.
- `AICc` AIC corrected for small samples

```r
geom_vline(xintercept = glance_cv$lambda.min) +
geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
```
See Also

glance(), drc::drm()

Other drc tidiers: augment.drc(), tidy.drc()

Examples

if (requireNamespace("drc", quietly = TRUE)) {

library(drc)

mod <- drm(dead / total ~ conc, type,
            weights = total, data = selenium, fct = LL.2(), type = "binomial"
)

tidy(mod)
tidy(mod, conf.int = TRUE)

glance(mod)

augment(mod, selenium)
}

---

glance.ergm

Glance at a(n) ergm object

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

## S3 method for class 'ergm'
glance(x, deviance = FALSE, mcmc = FALSE, ...)

---
Arguments

- **x**: An `ergm` object returned from a call to `ergm::ergm()`.
- **deviance**: Logical indicating whether or not to report null and residual deviance for the model, as well as degrees of freedom. Defaults to `FALSE`.
- **mcmc**: Logical indicating whether or not to report MCMC interval, burn-in and sample size used to estimate the model. Defaults to `FALSE`.
- **...**: Additional arguments to pass to `ergm::summary()`. **Cautionary note**: Mis-specified arguments may be silently ignored.

Value

`glance.ergm` returns a one-row tibble with the columns

- **independence**: Whether the model assumed dyadic independence
- **iterations**: The number of MCMLE iterations performed before convergence
- **logLik**: If applicable, the log-likelihood associated with the model
- **AIC**: The Akaike Information Criterion
- **BIC**: The Bayesian Information Criterion

If `deviance = TRUE`, and if the model supports it, the tibble will also contain the columns

- **null.deviance**: The null deviance of the model
- **df.null**: The degrees of freedom of the null deviance
- **residual.deviance**: The residual deviance of the model
- **df.residual**: The degrees of freedom of the residual deviance

See Also

- `glance()`, `ergm::ergm()`, `ergm::summary.ergm()`
- Other `ergm` tidiers: `tidy.ergm()`

---

**glance.factanal**

*Glance at a(n) factanal object*

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.
Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'factanal'
glance(x, ...)
```

Arguments

- `x` A `factanal` object created by `stats::factanal()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `converged` Logical indicating if the model fitting procedure was successful and converged.
- `df` Degrees of freedom used by the model.
- `method` Which method was used.
- `n` The total number of observations.
- `n.factors` The number of fitted factors.
- `nobs` Number of observations used.
- `p.value` P-value corresponding to the test statistic.
- `statistic` Test statistic.
- `total.variance` Total cumulative proportion of variance accounted for by all factors.

See Also

- `glance()`, `stats::factanal()`

Other factanal tidiers: `augment.factanal()`, `tidy.factanal()`
glance.felm

Examples

set.seed(123)

# data
m1 <- dplyr::tibble(
  v1 = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 4, 5, 6),
  v2 = c(1, 2, 1, 1, 1, 1, 2, 1, 2, 1, 3, 4, 3, 3, 4, 6, 5),
  v3 = c(3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5, 4, 6),
  v4 = c(3, 3, 4, 3, 3, 1, 2, 1, 1, 1, 2, 1, 1, 5, 6, 4),
  v5 = c(1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 1, 1, 1, 1, 6, 4, 5),
  v6 = c(1, 1, 1, 2, 1, 3, 3, 3, 4, 3, 1, 1, 1, 2, 1, 6, 5, 4)
)

# new data
m2 <- purrr::map_dfr(m1, rev)

# factor analysis objects
fit1 <- stats::factanal(m1, factors = 3, scores = "Bartlett")
fit2 <- stats::factanal(m1, factors = 3, scores = "regression")

# tidying the object
tidy(fit1)
tidy(fit2)

# augmented dataframe
augment(fit1)
augment(fit2)

# augmented dataframe (with new data)
augment(fit1, data = m2)
augment(fit2, data = m2)

---

**glance.felm**

*Glance at a(n) felm object*

**Description**

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.
Usage

```r
## S3 method for class 'felm'

glance(x, ...)
```

Arguments

- `x`: A `felm` object returned from `lfe::felm()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `adj.r.squared`: Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
- `df`: Degrees of freedom used by the model.
- `df.residual`: Residual degrees of freedom.
- `nobs`: Number of observations used.
- `p.value`: P-value corresponding to the test statistic.
- `r.squared`: R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
- `sigma`: Estimated standard error of the residuals.
- `statistic`: Test statistic.

Examples

```r
if (requireNamespace("lfe", quietly = TRUE)) {

  library(lfe)

  # Use built-in "airquality" dataset
  head(airquality)

  # No FEs; same as lm()
  est0 <- felm(Ozone ~ Temp + Wind + Solar.R, airquality)
  tidy(est0)
  augment(est0)

  # Add month fixed effects
  est1 <- felm(Ozone ~ Temp + Wind + Solar.R | Month, airquality)
  tidy(est1)
```
```r
tidy(est1, fe = TRUE)
augment(est1)
glance(est1)

# The "se.type" argument can be used to switch out different standard errors
# types on the fly. In turn, this can be useful exploring the effect of
# different error structures on model inference.
tidy(est1, se.type = "iid")
tidy(est1, se.type = "robust")

# Add clustered SEs (also by month)
est2 <- felm(Ozone ~ Temp + Wind + Solar.R | Month | 0 | Month, airquality)
tidy(est2, conf.int = TRUE)
tidy(est2, conf.int = TRUE, se.type = "cluster")
tidy(est2, conf.int = TRUE, se.type = "robust")
tidy(est2, conf.int = TRUE, se.type = "iid")
```

---

**glance.fitdistr**  
Glance at a(n) fitdistr object

### Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

### Usage

```r
## S3 method for class 'fitdistr'
glance(x, ...)
```

### Arguments

- `x`  
  A `fitdistr` object returned by `MASS::fitdistr()`.
Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in . . . , where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Value**

A `tibble::tibble()` with exactly one row and columns:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AIC</strong></td>
<td>Akaike’s Information Criterion for the model.</td>
</tr>
<tr>
<td><strong>BIC</strong></td>
<td>Bayesian Information Criterion for the model.</td>
</tr>
<tr>
<td><strong>logLik</strong></td>
<td>The log-likelihood of the model. [stats::logLik()] may be a useful reference.</td>
</tr>
<tr>
<td><strong>nobs</strong></td>
<td>Number of observations used.</td>
</tr>
</tbody>
</table>

**See Also**

`tidy()`, `MASS::fitdistr()`

Other fitdistr tidiers: `tidy.fitdistr()`

**Examples**

```r
if (requireNamespace("MASS", quietly = TRUE)) {
  set.seed(2015)
  x <- rnorm(100, 5, 2)
  library(MASS)
  fit <- fitdistr(x, dnorm, list(mean = 3, sd = 1))
  tidy(fit)
  glance(fit)
}
```
Description

Glance accepts a model object and returns a \texttt{tibble::tibble()} with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as \texttt{NA}.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an \texttt{NA} of the appropriate type.

Usage

\begin{verbatim}
## S3 method for class 'fixest'
.glance(x, ...)
\end{verbatim}

Arguments

- \texttt{x} A fixest object returned from any of the fixest estimators
- \texttt{...} Additional arguments passed to \texttt{summary} and \texttt{confint}. Important arguments are \texttt{se} and \texttt{cluster}. Other arguments are \texttt{dof}, \texttt{exact_dof}, \texttt{forceCovariance}, and \texttt{keepBounded}. See \texttt{summary.fixest}.

Value

A \texttt{tibble::tibble()} with exactly one row and columns:

- \texttt{adj.r.squared} Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
- \texttt{AIC} Akaike’s Information Criterion for the model.
- \texttt{BIC} Bayesian Information Criterion for the model.
- \texttt{logLik} The log-likelihood of the model. [\texttt{stats::logLik()}] may be a useful reference.
- \texttt{nobs} Number of observations used.
- \texttt{pseudo.r.squared} Like the R squared statistic, but for situations when the R squared statistic isn’t defined.
- \texttt{r.squared} R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
- \texttt{sigma} Estimated standard error of the residuals.
- \texttt{within.r.squared} R squared within fixed-effect groups.
Note

All columns listed below will be returned, but some will be NA, depending on the type of model estimated. `sigma`, `r.squared`, `adj.r.squared`, and `within.r.squared` will be NA for any model other than `feols`. `pseudo.r.squared` will be NA for `feols`.

Examples

```r
if (requireNamespace("fixest", quietly = TRUE)) {

library(fixest)

gravity <- feols(log(Euros) ~ log(dist_km) | Origin + Destination + Product + Year, trade)

tidy(gravity)
glance(gravity)
augment(gravity, trade)

## To get robust or clustered SEs, users can either:
# 1) Or, specify the arguments directly in the tidy() call

tidy(gravity, conf.int = TRUE, cluster = c("Product", "Year"))

tidy(gravity, conf.int = TRUE, se = "threeway")

# 2) Feed tidy() a summary.fixest object that has already accepted these arguments

gravity_summ <- summary(gravity, cluster = c("Product", "Year"))
tidy(gravity_summ, conf.int = TRUE)

# Approach (1) is preferred.

}
```

---

**glance.Gam**  
*Glance at a(n) Gam object*

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.
Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'Gam'
glance(x, ...)
```

Arguments

- `x` A Gam object returned from a call to `gam::gam()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

Glance at gam objects created by calls to `mgcv::gam()` with `glance.gam()`.

Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC**: Akaike's Information Criterion for the model.
- **BIC**: Bayesian Information Criterion for the model.
- **deviance**: Deviance of the model.
- **df**: Degrees of freedom used by the model.
- **df.residual**: Residual degrees of freedom.
- **logLik**: The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- **nobs**: Number of observations used.

See Also

`glance()`, `gam::gam()`

Other gam tidiers: `tidy.Gam()`
Glance at a(n) gam object

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'gam'

_glance(x, ...)
```

Arguments

- `x` A gam object returned from a call to mgcv::gam().
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with exactly one row and columns:

- **AIC** Akaike's Information Criterion for the model.
- **BIC** Bayesian Information Criterion for the model.
- **deviance** Deviance of the model.
- **df** Degrees of freedom used by the model.
- **df.residual** Residual degrees of freedom.
- **logLik** The log-likelihood of the model. [stats::logLik()] may be a useful reference.
- **nobs** Number of observations used.
glance.garch

See Also

glance().mgcv::gam()

Other mgcv tidiers: tidy.gam()

Examples

if (requireNamespace("mgcv", quietly = TRUE)) {
  g <- mgcv::gam(mpg ~ s(hp) + am + qsec, data = mtcars)
  tidy(g)
  tidy(g, parametric = TRUE)
  glance(g)
  augment(g)
}

glance.garch Tidy a(n) garch object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'garch'

```r
glance(x, test = c("box-ljung-test", "jarque-bera-test"), ...)
```

Arguments

- `x`: A garch object returned by `tsseries::garch()`.
- `test`: Character specification of which hypothesis test to use. The garch function reports 2 hypothesis tests: Jarque-Bera to residuals and Box-Ljung to squared residuals.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC**: Akaike's Information Criterion for the model.
- **BIC**: Bayesian Information Criterion for the model.
- **logLik**: The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- **method**: Which method was used.
- **nobs**: Number of observations used.
- **p.value**: P-value corresponding to the test statistic.
- **statistic**: Test statistic.
- **parameter**: Parameter field in the htest, typically degrees of freedom.

See Also

- `glance()`.
- `tseries::garch()`.

Other garch tidiers: `tidy.garch()`

---

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

**Usage**

```r
## S3 method for class 'geeglm'

# glance(geeglm)
```
Arguments

- **x**: A `geeglm` object returned from a call to `geepack::geeglm()`.

- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- **alpha**: Estimated correlation parameter for `geepack::geeglm`.
- **df.residual**: Residual degrees of freedom.
- **gamma**: Estimated scale parameter for `geepack::geeglm`.
- **max.cluster.size**: Max number of elements in clusters.
- **n.clusters**: Number of clusters.

See Also

- `glance()`, `geepack::geeglm()`

Examples

```r
if (requireNamespace("geepack", quietly = TRUE)) {

library(geepack)
data(state)

ds <- data.frame(state.region, state.x77)

geefit <- geeglm(Income ~ Frost + Murder,
                  id = state.region,
                  data = ds, family = gaussian,
                  corstr = "exchangeable")

 tidy(geefit)
tidy(geefit, conf.int = TRUE)
}
```
Glance at a(n) glm object

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'glm'
glance(x, ...)
```

Arguments

- `x` A glm object returned from stats::glm().
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with exactly one row and columns:

- AIC Akaike’s Information Criterion for the model.
- BIC Bayesian Information Criterion for the model.
- deviance Deviance of the model.
- df.null Degrees of freedom used by the null model.
- df.residual Residual degrees of freedom.
- logLik The log-likelihood of the model. [stats::logLik()] may be a useful reference.
- nobs Number of observations used.
- null.deviance Deviance of the null model.
See Also

stats::glm()

Other lm tidiers: augment.glm(), augment.lm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm.beta(), tidy.lm(), tidy.mlm(), tidy.summary.lm()

Examples

g <- glm(am ~ mpg, mtcars, family = "binomial")
glance(g)

---

glance.glmlenet  Glance at a(n) glmnet object

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

## S3 method for class 'glmnet'
glance(x, ...)

Arguments

x  A glmnet object returned from glmnet::glmnet().

...  Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with exactly one row and columns:

- `nobs` Number of observations used.
- `npasses` Total passes over the data across all lambda values.
- `nulldev` Null deviance.

See Also

`glance()`, `glmnet::glmnet()`

Other glmnet tidiers: `glance.cv.glmnet()`, `tidy.cv.glmnet()`, `tidy.glmnet()`

Examples

```r
if (requireNamespace("glmnet", quietly = TRUE)) {

  library(glmnet)

  set.seed(2014)
  x <- matrix(rnorm(100 * 20), 100, 20)
  y <- rnorm(100)
  fit1 <- glmnet(x, y)

  tidy(fit1)
  glance(fit1)

  library(dplyr)
  library(ggplot2)

  tidied <- tidy(fit1) %>% filter(term != "(Intercept)")

  ggplot(tidied, aes(step, estimate, group = term)) +
  geom_line()
  ggplot(tidied, aes(lambda, estimate, group = term)) +
  geom_line() +
  scale_x_log10()
  ggplot(tidied, aes(lambda, dev.ratio)) +
  geom_line()

  # works for other types of regressions as well, such as logistic
  g2 <- sample(1:2, 100, replace = TRUE)
  fit2 <- glmnet(x, g2, family = "binomial")
  tidy(fit2)
}
```
glance.glmRob

Glance at a(n) glmRob object

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

## S3 method for class 'glmRob'
glance(x, ...)

Arguments

x A glmRob object returned from robust::glmRob().

Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with exactly one row and columns:

deviance Deviance of the model.
df.residual Residual degrees of freedom.
nobs Number of observations used.
null.deviance Deviance of the null model.
sigma Estimated standard error of the residuals.
See Also

`robust::glmRob()`
Other robust tidiers: `augment.lmRob()`, `glance.lmRob()`, `tidy.glmRob()`, `tidy.lmRob()`

Examples

```r
if (requireNamespace("robust", quietly = TRUE)) {
  library(robust)
  gm <- glmRob(am ~ wt, data = mtcars, family = "binomial")
  tidy(gm)
  glance(gm)
}
```

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'gmm'
glance(x, ...)
```

Arguments

- `x`: A `gmm` object returned from `gmm::gmm()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
**Value**

A `tibble::tibble()` with exactly one row and columns:

- **df**: Degrees of freedom used by the model.
- **df.residual**: Residual degrees of freedom.
- **nobs**: Number of observations used.
- **p.value**: P-value corresponding to the test statistic.
- **statistic**: Test statistic.

**See Also**

- `glance()`, `gmm::gmm()`
- Other gmm tidiers: `tidy.gmm()`

**Examples**

```r
if (requireNamespace("gmm", quietly = TRUE)) {
  library(gmm)

  # examples come from the "gmm" package
  ## CAPM test with GMM
  data(Finance)
  r <- Finance[1:300, 1:10]
  rm <- Finance[1:300, "rm"]
  rf <- Finance[1:300, "rf"]

  z <- as.matrix(r - rf)
  t <- nrow(z)
  zm <- rm - rf
  h <- matrix(zm, t, 1)
  res <- gmm(z ~ zm, x = h)

  # tidy result
  tidy(res)
  tidy(res, conf.int = TRUE)
  tidy(res, conf.int = TRUE, conf.level = .99)

  # coefficient plot
  library(ggplot2)
  library(dplyr)

  tidy(res, conf.int = TRUE) %>%
    mutate(variable = reorder(term, estimate)) %>%
    ggplot(aes(estimate, variable)) +
    geom_point() +
    geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
    geom_vline(xintercept = 0, color = "red", lty = 2)
}
```
# from a function instead of a matrix
g <- function(theta, x) {
  gmat <- cbind(e, e * c(x[, 1]))
  return(gmat)
}
x <- as.matrix(cbind(rm, r))
res_black <- gmm(g, x = x, t0 = rep(0, 11))
tidy(res_black)
tidy(res_black, conf.int = TRUE)

## APT test with Fama-French factors and GMM
f1 <- zm
f2 <- Finance[1:300, "hml"] - rf
f3 <- Finance[1:300, "smb"] - rf
h <- cbind(f1, f2, f3)
res2 <- gmm(z ~ f1 + f2 + f3, x = h)
td2 <- tidy(res2, conf.int = TRUE)
td2

# coefficient plot
td2 %>%
  mutate(variable = reorder(term, estimate)) %>%
  ggplot(aes(estimate, variable)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0, color = "red", lty = 2)
}

---

glance.ivreg

Glance at a(n) ivreg object

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.
Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

**Usage**

```r
## S3 method for class 'ivreg'
glance(x, diagnostics = FALSE, ...)
```

**Arguments**

- `x`: An `ivreg` object created by a call to `AER::ivreg()`.
- `diagnostics`: Logical indicating whether or not to return the Wu-Hausman and Sargan diagnostic information.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Details**

This tidier currently only supports `ivreg`-classed objects outputted by the AER package. The ivreg package also outputs objects of class `ivreg`, and will be supported in a later release.

**Value**

A `tibble::tibble()` with exactly one row and columns:

- `adj.r.squared`: Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
- `df`: Degrees of freedom used by the model.
- `df.residual`: Residual degrees of freedom.
- `nobs`: Number of observations used.
- `r.squared`: R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
- `sigma`: Estimated standard error of the residuals.
- `statistic`: Wald test statistic.
- `p.value`: P-value for the Wald test.

**Note**

Beginning 0.7.0, `glance.ivreg` returns statistics for the Wu-Hausman test for endogeneity and the Sargan test of overidentifying restrictions. Sargan test values are returned as `NA` if the number of instruments is not greater than the number of endogenous regressors.
See Also

`glance()`, `AER::ivreg()`

Other ivreg tidiers: `augment.ivreg()`, `tidy.ivreg()`

Examples

```r
if (requireNamespace("AER", quietly = TRUE)) {
  library(AER)
  data("CigarettesSW", package = "AER")

  ivr <- ivreg(
    log(packs) ~ income | population,
    data = CigarettesSW,
    subset = year == "1995"
  )

  summary(ivr)

  tidy(ivr)
  tidy(ivr, conf.int = TRUE)
  tidy(ivr, conf.int = TRUE, instruments = TRUE)

  augment(ivr)
  augment(ivr, data = CigarettesSW)
  augment(ivr, newdata = CigarettesSW)

  glance(ivr)
}
```

---

**glance.kmeans**

*Glance at a(n) kmeans object*

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`. 

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

### Usage

```r
## S3 method for class 'kmeans'
glance(x, ...)
```

### Arguments

- `x` A `kmeans` object created by `stats::kmeans()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

### Value

A `tibble::tibble()` with exactly one row and columns:

- `betweenss` The total between-cluster sum of squares.
- `iter` Iterations of algorithm/fitting procedure completed.
- `tot.withinss` The total within-cluster sum of squares.
- `totss` The total sum of squares.

### See Also

- `glance()`, `stats::kmeans()`

Other kmeans tidiers: `augment.kmeans()`, `tidy.kmeans()`

### Examples

```r
## Not run:
library(cluster)
library(dplyr)

library(modeldata)
data(hpc_data)

x <- hpc_data[, 2:5]
fit <- pam(x, k = 4)
tidy(fit)
```
glance(fit)  
augment(fit, x)  
## End(Not run)

glance.lavaan  
Glance at a(n) lavaan object

Description
Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.
Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.
Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.
Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage
```r
## S3 method for class 'lavaan'
glance(x, ...)
```

Arguments
- **x**  
  A lavaan object, such as those returned from `lavaan::cfa()`, and `lavaan::sem()`.
- **...**  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value
A one-row `tibble::tibble` with columns:

- **chisq**  
  Model chi squared
- **npar**  
  Number of parameters in the model
- **rmsea**  
  Root mean square error of approximation
### glance.lavaan

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rmsea.conf.high</td>
<td>95 percent upper bound on RMSEA</td>
</tr>
<tr>
<td>srmr</td>
<td>Standardised root mean residual</td>
</tr>
<tr>
<td>agfi</td>
<td>Adjusted goodness of fit</td>
</tr>
<tr>
<td>cfi</td>
<td>Comparative fit index</td>
</tr>
<tr>
<td>tli</td>
<td>Tucker Lewis index</td>
</tr>
<tr>
<td>AIC</td>
<td>Akaike information criterion</td>
</tr>
<tr>
<td>BIC</td>
<td>Bayesian information criterion</td>
</tr>
<tr>
<td>ngroups</td>
<td>Number of groups in model</td>
</tr>
<tr>
<td>nobs</td>
<td>Number of observations included</td>
</tr>
<tr>
<td>norig</td>
<td>Number of observation in the original dataset</td>
</tr>
<tr>
<td>nexcluded</td>
<td>Number of excluded observations</td>
</tr>
<tr>
<td>converged</td>
<td>Logical - Did the model converge</td>
</tr>
<tr>
<td>estimator</td>
<td>Estimator used</td>
</tr>
<tr>
<td>missing_method</td>
<td>Method for eliminating missing data</td>
</tr>
</tbody>
</table>


**See Also**

- `glance()`, `lavaan::cfa()`, `lavaan::sem()`, `lavaan::fitmeasures()`
- Other lavaan tidiers: `tidy.lavaan()`

**Examples**

```r
## Not run:
library(lavaan)

cfa.fit <- cfa(
  "F =~ x1 + x2 + x3 + x4 + x5",
  data = HolzingerSwineford1939, group = "school"
)

# Not run: glance(cfa.fit)

## End(Not run)
```
### Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

### Usage

```r
## S3 method for class 'lm'

glance(x, ...)
```

### Arguments

- `x` An `lm` object created by `stats::lm()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

### Value

A `tibble::tibble()` with exactly one row and columns:

- `adj.r.squared` Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
- `AIC` Akaike’s Information Criterion for the model.
- `BIC` Bayesian Information Criterion for the model.
- `deviance` Deviance of the model.
- `df.residual` Residual degrees of freedom.
- `logLik` The log-likelihood of the model. [`stats::logLik()`] may be a useful reference.
nobs  Number of observations used.
p.value  P-value corresponding to the test statistic.
r.squared  R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
sigma  Estimated standard error of the residuals.
statistic  Test statistic.
df  The degrees for freedom from the numerator of the overall F-statistic. This is new in broom 0.7.0. Previously, this reported the rank of the design matrix, which is one more than the numerator degrees of freedom of the overall F-statistic.

See Also

glance(), glance.summary.lm()

Other lm tidiers: augment.glm(), augment.lm(), glance.glm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm.beta(), tidy.lm(), tidy.mlm(), tidy.summary.lm()

Examples

library(ggplot2)
library(dplyr)

mod <- lm(mpg ~ wt + qsec, data = mtcars)
tidy(mod)
glance(mod)

# coefficient plot
d <- tidy(mod, conf.int = TRUE)

ggplot(d, aes(estimate, term, xmin = conf.low, xmax = conf.high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0, lty = 4) +
  geom_errorbarh()

# Aside: There are tidy() and glance() methods for lm.summary objects too.
# This can be useful when you want to conserve memory by converting large lm
# objects into their leaner summary.lm equivalents.
s <- summary(mod)
tidy(s, conf.int = TRUE)
glance(s)

augment(mod)
augment(mod, mtcars, interval = "confidence")

# predict on new data
newdata <- mtcars %>%
  head(6) %>%
  mutate(wt = wt + 1)
augment(mod, newdata = newdata)

# ggplot2 example where we also construct 95% prediction interval
mod2 <- lm(mpg ~ wt, data = mtcars)  ## simpler bivariate model since we’re plotting in 2D
au <- augment(mod2, newdata = newdata, interval = "prediction")

ggplot(au, aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted)) +
  geom_ribbon(aes(ymin = .lower, ymax = .upper), col = NA, alpha = 0.3)

# predict on new data without outcome variable. Output does not include .resid
newdata <- newdata %>%
  select(-mpg)
augment(mod, newdata = newdata)

au <- augment(mod, data = mtcars)

ggplot(au, aes(.hat, .std.resid)) +
  geom_vline(size = 2, colour = "white", xintercept = 0) +
  geom_hline(size = 2, colour = "white", yintercept = 0) +
  geom_point() +
  geom_smooth(se = FALSE)

plot(mod, which = 6)

ggplot(au, aes(.hat, .cooksd)) +
  geom_vline(xintercept = 0, colour = NA) +
  geom_abline(slope = seq(0, 3, by = 0.5), colour = "white") +
  geom_smooth(se = FALSE) +
  geom_point()

# column-wise models
a <- matrix(rnorm(20), nrow = 10)
b <- a + rnorm(length(a))
result <- lm(b ~ a)
tidy(result)

glance.lmodel2

Glance accepts a model object and returns a \texttt{tibble::tibble()} with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information. Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.
Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

**Usage**

```r
# S3 method for class 'lmodel2'
glance(x, ...)
```

**Arguments**

- `x` A lmodel2 object returned by `lmodel2::lmodel2()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

**Value**

A `tibble::tibble()` with exactly one row and columns:

- `nobs` Number of observations used.
- `p.value` P-value corresponding to the test statistic.
- `r.squared` R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
- `theta` Angle between OLS lines `lm(y ~ x)` and `lm(x ~ y)`
- `H` H statistic for computing confidence interval of major axis slope

**See Also**

`glance()`, `lmodel2::lmodel2()`

Other lmodel2 tidiers: `tidy.lmodel2()`

**Examples**

```r
if (requireNamespace("lmodel2", quietly = TRUE)) {

library(lmodel2)

data(mod2ex2)

Ex2.res <- lmodel2(Prey ~ Predators, data = mod2ex2, "relative", "relative", 99)
}
```
Ex2.res
tidy(Ex2.res)
glance(Ex2.res)

# this allows coefficient plots with ggplot2
library(ggplot2)
ggplot(tidy(Ex2.res), aes(estimate, term, color = method)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high))

---

**glance.lmRob**  
Glance at an lmRob object

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

**Usage**

```r
## S3 method for class 'lmRob'
glance(x, ...)
```

**Arguments**

- `x`  
  A lmRob object returned from `robust::lmRob()`.

- `...`  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
glance.lmrob

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.
Usage

```r
## S3 method for class 'lmrob'
glance(x, ...)
```

Arguments

- `x`: A `lmrob` object returned from `robustbase::lmrob()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Details

For tidiers for robust models from the `MASS` package see `tidy.rlm()`.

Value

A `tibble::tibble()` with exactly one row and columns:

- `df.residual`: Residual degrees of freedom.
- `r.squared`: R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
- `sigma`: Estimated standard error of the residuals.

See Also

`robustbase::lmrob()`

Other robustbase tidiers: `augment.glmrob()`, `augment.lmrob()`, `tidy.glmrob()`, `tidy.lmrob()`

Examples

```r
library(robustbase)
# From the robustbase::lmrob examples:
data(coleman)
set.seed(0)

m <- robustbase::lmrob(Y ~ ., data = coleman)
tidy(m)
augment(m)
glance(m)

# From the robustbase::glmrob examples:
data(carrots)
Rfit <- glmrob(cbind(success, total - success) ~ logdose + block,

```
family = binomial, data = carrots, method = "Mqle",
control = glmrobMqle.control(tcc = 1.2)
)
tidy(Rfit)
augment(Rfit)

---

### Glance at a(n) margins object

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

**Usage**

```r
## S3 method for class 'margins'
glance(x, ...)
```

**Arguments**

- `x`: A margins object returned from `margins::margins()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Value**

A `tibble::tibble()` with exactly one row and columns:

- `adj.r.squared`: Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
- `df`: Degrees of freedom used by the model.
df.residual   Residual degrees of freedom.
nobs          Number of observations used.
p.value       P-value corresponding to the test statistic.
r.squared     R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
sigma         Estimated standard error of the residuals.
statistic     Test statistic.

Examples

if (requireNamespace("margins", quietly = TRUE)) {

library(margins)

## Example 1: Logit model ##
mod_log <- glm(am ~ cyl + hp + wt, data = mtcars, family = binomial)

# Get tidied "naive" model coefficients
tidy(mod_log)

# Convert to marginal effects with margins::margins()
marg_log <- margins(mod_log)

# Get tidied marginal effects
tidy(marg_log)
tidy(marg_log, conf.int = TRUE)
glance(marg_log) ## Requires running the underlying model again. Quick for this example.
## Not run: augment(marg_log) ## Not supported.
augment(mod_log) ## But can get the same info by running on the underlying model.

## Example 2: Threeway interaction terms ##
mod_ie <- lm(mpg ~ wt * cyl * disp, data = mtcars)

# Get tidied "naive" model coefficients
tidy(mod_ie)

# Convert to marginal effects with margins::margins()
marg_ie0 <- margins(mod_ie)

# Get tidied marginal effects
tidy(marg_ie0)
glance(marg_ie0)

# Marginal effects evaluated at specific values of a variable (here: cyl)
marg_ie1 <- margins(mod_ie, at = list(cyl = c(4, 6, 8)))
tidy(marg_ie1)

# Marginal effects of one interaction variable (here: wt), modulated at
# specific values of the two other interaction variables (here: cyl and drat)
```r
marg_ie2 <- margins(mod_ie,
                   variables = "wt", ## Main var
                   at = list(cyl = c(4,6,8), drat = c(3, 3.5, 4))) ## Modulating vars

 tidy(marg_ie2)
```

---

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

**Usage**

```r
## S3 method for class 'Mclust'
glance(x, ...)
```

**Arguments**

- `x` An `Mclust` object return from `mclust::Mclust()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.1vel = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

**Value**

A `tibble::tibble()` with exactly one row and columns:

- **BIC** Bayesian Information Criterion for the model.
- **df** Degrees of freedom used by the model.
The log-likelihood of the model. [stats::logLik()] may be a useful reference.

Number of observations used.

A string denoting the model type with optimal BIC

Number mixture components in optimal model

If the other model contains a noise component, the value of the hypervolume parameter. Otherwise ‘NA’.

Examples

```r
if (requireNamespace("mclust", quietly = TRUE)) {

library(dplyr)
library(mclust)
set.seed(27)

centers <- tibble::tibble(
  cluster = factor(1:3),
  num_points = c(100, 150, 50), # number points in each cluster
  x1 = c(5, 0, -3), # x1 coordinate of cluster center
  x2 = c(-1, 1, -2) # x2 coordinate of cluster center
)

points <- centers %>%
  mutate(
    x1 = purrr::map2(num_points, x1, rnorm),
    x2 = purrr::map2(num_points, x2, rnorm)
  ) %>%
  dplyr::select(-num_points, -cluster) %>%
  tidyr::unnest(c(x1, x2))

m <- mclust::Mclust(points)

tidy(m)
augment(m, points)

```

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.
Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

**Usage**

```r
## S3 method for class 'mfx'
glance(x, ...)

## S3 method for class 'logitmfx'
glance(x, ...)

## S3 method for class 'negbinmfx'
glance(x, ...)

## S3 method for class 'poissonmfx'
glance(x, ...)

## S3 method for class 'probitmfx'
glance(x, ...)
```

**Arguments**

- `x`  
  A `logitmfx`, `negbinmfx`, `poissonmfx`, or `probitmfx` object. (Note that `betamfx` objects receive their own set of tidiers.)

- `...`  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Details**

This generic glance method wraps `glance.glm()` for applicable objects from the `mfx` package.

**Value**

A `tibble::tibble()` with exactly one row and columns:

- **AIC**  
  Akaike’s Information Criterion for the model.

- **BIC**  
  Bayesian Information Criterion for the model.
deviance  Deviance of the model.
df.null   Degrees of freedom used by the null model.
df.residual  Residual degrees of freedom.
logLik   The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs   Number of observations used.
null.deviance  Deviance of the null model.

See Also
glance.glm(), mfx::logitmfx(), mfx::negbinmfx(), mfx::poissonmfx(), mfx::probitmfx()

Other mfx tidiers: augment.betamfx(), augment.mfx(), glance.betamfx(), tidy.betamfx(), tidy.mfx()

Examples

if (requireNamespace("mfx", quietly = TRUE)) {

  ## Not run:
  library(mfx)

  ## Get the marginal effects from a logit regression
  mod_logmfx <- logitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)
  tidy(mod_logmfx, conf.int = TRUE)

  ## Compare with the naive model coefficients of the same logit call (not run)
  # tidy(glm(am ~ cyl + hp + wt, family = binomial, data = mtcars), conf.int = TRUE)

  augment(mod_logmfx)
  glance(mod_logmfx)

  ## Another example, this time using probit regression
  mod_probmfx <- probitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)
  tidy(mod_probmfx, conf.int = TRUE)
  augment(mod_probmfx)
  glance(mod_probmfx)

  ## End(Not run)
}
Glance accepts a model object and returns a \texttt{tibble::tibble()} with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as \texttt{NA}.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an \texttt{NA} of the appropriate type.

Usage

```r
## S3 method for class 'mjoint'

\texttt{glance(x, \ldots)}
```

Arguments

- \texttt{x} An \texttt{mjoint} object returned from \texttt{joineRML::mjoint()}.
- \texttt{\ldots} Additional arguments. Not used. Needed to match generic signature only. \textbf{Cautionary note:} Misspelled arguments will be absorbed in \ldots, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass \texttt{conf.\_level = 0.9}, all computation will proceed using \texttt{conf.\_level = 0.95}. Additionally, if you pass \texttt{newdata = my\_tibble} to an \texttt{augment()} method that does not accept a \texttt{newdata} argument, it will use the default value for the \texttt{data} argument.

Value

A \texttt{tibble::tibble()} with exactly one row and columns:

- \texttt{AIC} Akaike's Information Criterion for the model.
- \texttt{BIC} Bayesian Information Criterion for the model.
- \texttt{logLik} The log-likelihood of the model. \texttt{[stats::logLik()]} may be a useful reference.
- \texttt{sigma2\_j} The square root of the estimated residual variance for the j-th longitudinal process.
See Also

glance(), joineRML::mjoint()

Other mjoint tidiers: tidy.mjoint()

Examples

if (requireNamespace("joineRML", quietly = TRUE)) {

## Not run:
# Fit a joint model with bivariate longitudinal outcomes
library(joineRML)
data(heart.valve)
hvd <- heart.valve[!is.na(heart.valve$log.grad) &
  !is.na(heart.valve$log.lvmi) &
  heart.valve$num <= 50, ]
fit <- mjoint(
  formLongFixed = list(
    "grad" = log.grad ~ time + sex + hs,
    "lvmi" = log.lvmi ~ time + sex
  ),
  formLongRandom = list(
    "grad" = ~ 1 | num,
    "lvmi" = ~ time | num
  ),
  formSurv = Surv(fuyrs, status) ~ age,
  data = hvd,
inits = list("gamma" = c(0.11, 1.51, 0.80)),
timeVar = "time"
)

# Extract the survival fixed effects
tidy(fit)

# Extract the longitudinal fixed effects
tidy(fit, component = "longitudinal")

# Extract the survival fixed effects with confidence intervals
tidy(fit, ci = TRUE)

# Extract the survival fixed effects with confidence intervals based
# on bootstrapped standard errors
bSE <- bootSE(fit, nboot = 5, safe.boot = TRUE)
tidy(fit, boot_se = bSE, ci = TRUE)

# Augment original data with fitted longitudinal values and residuals
hvd2 <- augment(fit)

# Extract model statistics
glance(fit)

## End(Not run)
Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

## Usage

```r
## S3 method for class 'mlogit'

glance(x, ...)
```

### Arguments

- `x` an object returned from mlogit::mlogit().
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

### Value

A tibble::tibble() with exactly one row and columns:

- `AIC` Akaike’s Information Criterion for the model.
- `BIC` Bayesian Information Criterion for the model.
- `logLik` The log-likelihood of the model. [stats::logLik()] may be a useful reference.
- `nobs` Number of observations used.
- `rho2` McFadden’s rho squared with respect to a market shares (constants-only) model.
- `rho20` McFadden’s rho squared with respect to an equal shares (no information) model.
See Also

`glance()`, `mlogit::mlogit()`

Other mlogit tidiers: `augment.mlogit()`, `tidy.mlogit()`

Examples

```r
if (requireNamespace("mlogit", quietly = TRUE)) {
  ## Not run:
  library(mlogit)
  data("Fishing", package = "mlogit")
  Fish <- dfidx(Fishing, varying = 2:9, shape = "wide", choice = "mode")
  m <- mlogit(mode ~ price + catch | income, data = Fish)

  tidy(m)
  augment(m)
  glance(m)

  ## End(Not run)
}
```

---

**glance.muhaz**

Glance at a(n) muhaz object

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'muhaz'
glance(x, ...)
```
Arguments

`x`  
A muhaz object returned by `muhaz::muhaz()`.

`...`  
Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `max.hazard`  
  Maximal estimated hazard.

- `max.time`  
  The maximum observed event or censoring time.

- `min.hazard`  
  Minimal estimated hazard.

- `min.time`  
  The minimum observed event or censoring time.

- `nobs`  
  Number of observations used.

See Also

`glance()`, `muhaz::muhaz()`

Other muhaz tidiers: `tidy.muhaz()`

Examples

```r
if (requireNamespace("muhaz", quietly = TRUE)) {
  if (requireNamespace("survival", quietly = TRUE)) {

    library(muhaz)
    library(survival)

    x <- muhaz(ovarian$futime, ovarian$fustat)
    tidy(x)
    glance(x)
  }
}
```
Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'multinom'
glance(x, ...)
```

Arguments

- `x` A multinom object returned from nnet::multinom().
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with exactly one row and columns:

- `AIC` Akaike's Information Criterion for the model.
- `deviance` Deviance of the model.
- `edf` The effective degrees of freedom.
- `nobs` Number of observations used.

See Also

- glance(), nnet::multinom()
- Other multinom tidiers: tidy.multinom()
Examples

```r
if (requireNamespace("nnet", quietly = TRUE)) {

  library(nnet)
  library(MASS)

  example(birthwt)
  bwt.mu <- multinom(low ~ ., bwt)
  tidy(bwt.mu)
  glance(bwt.mu)

  ##* This model is a truly terrible model
  ##* but it should show you what the output looks
  ##* like in a multinomial logistic regression

  fit.gear <- multinom(gear ~ mpg + factor(am), data = mtcars)
  tidy(fit.gear)
  glance(fit.gear)
}
```

---

### glance.negbin

**Glance at a(n) negbin object**

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

**Usage**

```r
## S3 method for class 'negbin'
glance(x, ...)
```
Arguments

x  A negbin object returned by \texttt{MASS::glm.nb()}.

... Additional arguments. Not used. Needed to match generic signature only. \textbf{Cautionary note:} Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass \texttt{conf.lvel = 0.9}, all computation will proceed using \texttt{conf.level = 0.95}. Additionally, if you pass \texttt{newdata = my_tibble} to an \texttt{augment()} method that does not accept a \texttt{newdata} argument, it will use the default value for the data argument.

Value

A \texttt{tibble::tibble()} with exactly one row and columns:

- \texttt{AIC}  Akaike’s Information Criterion for the model.
- \texttt{BIC}  Bayesian Information Criterion for the model.
- \texttt{deviance}  Deviance of the model.
- \texttt{df.null}  Degrees of freedom used by the null model.
- \texttt{df.residual}  Residual degrees of freedom.
- \texttt{logLik}  The log-likelihood of the model. \texttt{[stats::logLik()]} may be a useful reference.
- \texttt{nobs}  Number of observations used.
- \texttt{null.deviance}  Deviance of the null model.

See Also

\texttt{glance()}, \texttt{MASS::glm.nb()}

Other glm.nb tidiers: \texttt{tidy.negbin()}

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {
  library(MASS)

r <- glm.nb(Days ~ Sex/(Age + Eth*Lrn), data = quine)

tidy(r)
glance(r)
}
```
Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'nrlq'
glance(x, ...)
```

Arguments

- `x` A `nrlq` object returned from `quantreg::nrlq()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lev = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `AIC` Akaike’s Information Criterion for the model.
- `BIC` Bayesian Information Criterion for the model.
- `df.residual` Residual degrees of freedom.
- `logLik` The log-likelihood of the model. [stats::logLik()] may be a useful reference.
- `tau` Quantile.
glance.nls

Glance at a(n) nls object

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'nls'
glance(x, ...)
```

Arguments

- `x`  
  An `nls` object returned from `stats::nls()`.
- `...`  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC**  
  Akaike's Information Criterion for the model.
- **BIC**  
  Bayesian Information Criterion for the model.
- **deviance**  
  Deviance of the model.
df.residual  Residual degrees of freedom.
finTol      The achieved convergence tolerance.
isConv      Whether the fit successfully converged.
logLik      The log-likelihood of the model. [stats::logLik()] may be a useful reference.
nobs        Number of observations used.
sigma       Estimated standard error of the residuals.

See Also
tidy, stats::nls()
Other nls tidiers: augment.nls(), tidy.nls()

Examples

n <- nls(mpg ~ k * e^wt, data = mtcars, start = list(k = 1, e = 2))
tidy(n)
augment(n)
glance(n)

library(ggplot2)
ggplot(augment(n), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))

newdata <- head(mtcars)
newdata$wt <- newdata$wt + 1
augment(n, newdata = newdata)
Usage

```
## S3 method for class 'orcutt'
glance(x, ...)
```

Arguments

- **x**: An orcutt object returned from `orcutt::cochrane.orcutt()`.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- **adj.r.squared**: Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
- **dw.original**: Durbin-Watson statistic of original fit.
- **dw.transformed**: Durbin-Watson statistic of transformed fit.
- **nobs**: Number of observations used.
- **number.interaction**: Number of interactions.
- **p.value.original**: P-value of original Durbin-Watson statistic.
- **p.value.transformed**: P-value of autocorrelation after transformation.
- **r.squared**: R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
- **rho**: Spearman’s rho autocorrelation

See Also

- `glance()`, `orcutt::cochrane.orcutt()`
- Other orcutt tidiers: `tidy.orcutt()`

Examples

```r
if (requireNamespace("orcutt", quietly = TRUE)) {

library(orcutt)

reg <- lm(mpg ~ wt + qsec + disp, mtcars)
```

tidy(reg)
co <- cochrane.orcutt(reg)
tidy(co)
glance(co)

---

**glance.pam**

*Glance at a(n) pam object*

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as **NA**.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an **NA** of the appropriate type.

**Usage**

```r
## S3 method for class 'pam'
glance(x, ...)
```

**Arguments**

- `x`: An pam object returned from `cluster::pam()`
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with exactly one row and columns:

- `avg.silhouette.width`
  
  The average silhouette width for the dataset.

See Also

- `glance()`, `cluster::pam()`

Other pam tidiers: `augment.pam()`, `tidy.pam()`

Examples

```r
## Not run:
library(dplyr)
library(ggplot2)
library(cluster)
library(modeldata)
data(hpc_data)

x <- hpc_data[, 2:5]
p <- pam(x, k = 4)
tidy(p)
glance(p)
augment(p, x)

augment(p, x) %>%
  ggplot(aes(compounds, input_fields)) +
  geom_point(aes(color = .cluster)) +
  geom_text(aes(label = cluster), data = tidy(p), size = 10)

## End(Not run)
```

---

### Glance at a(n) plm object

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.
Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'plm'

glance(x, ...)
```

Arguments

- `x` A `plm` objected returned by `plm::plm()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `adj.r.squared` Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
- `deviance` Deviance of the model.
- `df.residual` Residual degrees of freedom.
- `nobs` Number of observations used.
- `p.value` P-value corresponding to the test statistic.
- `r.squared` R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
- `statistic` F-statistic

See Also

- `glance()`, `plm::plm()`
- Other `plm` tidiers: `augment.plm()`, `tidy.plm()`

Examples

```r
if (requireNamespace("plm", quietly = TRUE)) {
  library(plm)
  data("Produc", package = "plm")
  zz <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,}
```r
data = Produc, index = c("state", "year")

summary(zz)

tidy(zz)
tidy(zz, conf.int = TRUE)
tidy(zz, conf.int = TRUE, conf.level = 0.9)

augment(zz)
glance(zz)

```
Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC**
  - Akaike's Information Criterion for the model.
- **BIC**
  - Bayesian Information Criterion for the model.
- **chi.squared**
  - The Pearson Chi-Square goodness of fit statistic for multiway tables.
- **df**
  - Degrees of freedom used by the model.
- **df.residual**
  - Residual degrees of freedom.
- **logLik**
  - The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- **nobs**
  - Number of observations used.
- **g.squared**
  - The likelihood ratio/deviance statistic

See Also

- `glance()`, `poLCA::poLCA()`
- Other poLCA tidiers: `augment.poLCA()`, `tidy.poLCA()`

Examples

```r
if (requireNamespace("poLCA", quietly = TRUE)) {

  library(poLCA)
  library(dplyr)

  data(values)
  f <- cbind(A, B, C, D) ~ 1
  M1 <- poLCA(f, values, nclass = 2, verbose = FALSE)

  M1
tidy(M1)
augment(M1)
  glance(M1)

  library(ggplot2)

  ggplot(tidy(M1), aes(factor(class), estimate, fill = factor(outcome))) +
    geom_bar(stat = "identity", width = 1) +
    facet_wrap(~variable)

  ## Three-class model with a single covariate.

  data(election)
  f2a <- cbind(
    MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG,
    MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB
  ) ~ PARTY
  nes2a <- poLCA(f2a, election, nclass = 3, nrep = 5, verbose = FALSE)

td <- tidy(nes2a)
```
td

# show

ggplot(td, aes(outcome, estimate, color = factor(class), group = class)) +
  geom_line() +
  facet_wrap(~variable, nrow = 2) +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))

au <- augment(nes2a)
au
count(au, .class)

# if the original data is provided, it leads to NAs in new columns
# for rows that weren't predicted
au2 <- augment(nes2a, data = election)
au2
dim(au2)

---

## S3 method for class 'polr'
glance(x, ...)

### Arguments

- **x**
  - A polr object returned from MASS::polr().
Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC** Akaike’s Information Criterion for the model.
- **BIC** Bayesian Information Criterion for the model.
- **deviance** Deviance of the model.
- **df.residual** Residual degrees of freedom.
- **edf** The effective degrees of freedom.
- **logLik** The log-likelihood of the model. [stats::logLik()] may be a useful reference.
- **nobs** Number of observations used.

See Also

- `tidy.MMSS::polr()`
- Other ordinal tidiers: `augment.clm()`, `augment.polr()`, `glance.clmm()`, `glance.clm()`, `glance.svyolr()`, `tidy.clmm()`, `tidy.clm()`, `tidy.polr()`, `tidy.svyolr()`

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {
  library(MASS)
  fit <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
  tidy(fit, exponentiate = TRUE, conf.int = TRUE)
  glance(fit)
  augment(fit, type.predict = "class")

  fit2 <- polr(factor(gear) ~ am + mpg + qsec, data = mtcars)
  tidy(fit, p.values = TRUE)
}
```
Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'pyears'
glance(x, ...)
```

Arguments

- `x`: A `pyears` object returned from `survival::pyears()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `nobs`: Number of observations used.
- `total`: total number of person-years tabulated
- `offtable`: total number of person-years off table
See Also

- `glance()`, `survival::pyears()`
- Other `pyears` tidiers: `tidy.pyears()`
- Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {

library(survival)

temp.yr <- tcut(mgus$dxyr, 55:92, labels = as.character(55:91))
temp.age <- tcut(mgus$age, 34:101, labels = as.character(34:100))
ptime <- ifelse(is.na(mgus$pctime), mgus$futime, mgus$pctime)
pstat <- ifelse(is.na(mgus$pctime), 0, 1)
pfit <- pyears(Surv(ptime / 365.25, pstat) ~ temp.yr + temp.age + sex, mgus,
data.frame = TRUE)
tidy(pfit)
glance(pfit)

# if data.frame argument is not given, different information is present in
# output
pfit2 <- pyears(Surv(ptime / 365.25, pstat) ~ temp.yr + temp.age + sex, mgus)
tidy(pfit2)
glance(pfit2)
}
```

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

---

**glance.ridgelm**  
**Glance at a(n) ridgelm object**

- **Description**
  - Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.
  - Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.
  - Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`. 
Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'ridgelm'
glance(x, ...)
```

Arguments

- `x`: A `ridgelm` object returned from `MASS::lm.ridge()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

This is similar to the output of `select.ridgelm`, but it is returned rather than printed.

Value

A `tibble::tibble()` with exactly one row and columns:

- `kHKB`: modified HKB estimate of the ridge constant
- `kLW`: modified L-W estimate of the ridge constant
- `lambdaGCV`: choice of lambda that minimizes GCV

See Also

- `glance()`, `MASS::select.ridgelm()`, `MASS::lm.ridge()`
- Other `ridgelm` tidiers: `tidy.ridgelm()`

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {

  names(longley)[1] <- "y"
  fit1 <- MASS::lm.ridge(y ~ ., longley)
  tidy(fit1)

  fit2 <- MASS::lm.ridge(y ~ ., longley, lambda = seq(0.001, .05, .001))
  td2 <- tidy(fit2)
  g2 <- glance(fit2)
```
# coefficient plot
library(ggplot2)
ggplot(td2, aes(lambda, estimate, color = term)) +
  geom_line()

# GCV plot
ggplot(td2, aes(lambda, GCV)) +
  geom_line()

# add line for the GCV minimizing estimate
ggplot(td2, aes(lambda, GCV)) +
  geom_line() +
  geom_vline(xintercept = g2$lambdaGCV, col = "red", lty = 2)

---

glance.rlm

Glance at a(n) rlm object

Description

Glance accepts a model object and returns a \texttt{tibble::tibble()} with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as \texttt{NA}.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an \texttt{NA} of the appropriate type.

Usage

```r
## S3 method for class 'rlm'
glance(x, ...)
```

Arguments

- `x` An \texttt{rlm} object returned by \texttt{MASS::rlm()}.  
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lev = 0.9`, all computation will proceed
using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Value**

A `tibble::tibble()` with exactly one row and columns:

- **AIC**: Akaike's Information Criterion for the model.
- **BIC**: Bayesian Information Criterion for the model.
- **converged**: Logical indicating if the model fitting procedure was successful and converged.
- **deviance**: Deviance of the model.
- **logLik**: The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- **nobs**: Number of observations used.
- **sigma**: Estimated standard error of the residuals.

**See Also**

`glance()`, `MASS::rlm()`

Other `rlm` tidiers: `augment.rlm()`, `tidy.rlm()`

**Examples**

```r
if (requireNamespace("MASS", quietly = TRUE)) {
  library(MASS)
  r <- rlm(stack.loss ~ ., stackloss)
  tidy(r)
  augment(r)
  glance(r)
}
```

---

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.
Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'rma'
glance(x, ...)
```

Arguments

- `x` An rma object such as those created by `metafor::rma()`, `metafor::rma.uni()`, `metafor::rma.glmm()`, `metafor::rma.mh()`, `metafor::rma.mv()`, or `metafor::rma.peto()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `cochran.qe` In meta-analysis, test statistic for the Cochran’s $Q_e$ test of residual heterogeneity.
- `cochran.qm` In meta-analysis, test statistic for the Cochran’s $Q_m$ omnibus test of coefficients.
- `df.residual` Residual degrees of freedom.
- `h.squared` Value of the H-Squared statistic.
- `i.squared` Value of the I-Squared statistic.
- `measure` The measure used in the meta-analysis.
- `method` Which method was used.
- `nobs` Number of observations used.
- `p.value.cochran.qe` In meta-analysis, p-value for the Cochran’s $Q_e$ test of residual heterogeneity.
- `p.value.cochran.qm` In meta-analysis, p-value for the Cochran’s $Q_m$ omnibus test of coefficients.
- `tau.squared` In meta-analysis, estimated amount of residual heterogeneity.
- `tau.squared.se` In meta-analysis, standard error of residual heterogeneity.
Examples

```r
library(metafor)

df <- escalc(
  measure = "RR",
  ai = tpos,
  bi = tneg,
  ci = cpos,
  di = cneg,
  data = dat.bcg
)

meta_analysis <- rma(yi, vi, data = df, method = "EB")

glance(meta_analysis)
```

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'rq'
glance(x, ...)
```

Arguments

- `x` An `rq` object returned from `quantreg::rq()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed
using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Details**

Only models with a single `tau` value may be passed. For multiple values, please use a `purrr::map()` workflow instead, e.g.

```r
taus %>%
  map(function(tau_val) rq(y ~ x, tau = tau_val)) %>%
  map_dfr(glance)
```

**Value**

A `tibble::tibble()` with exactly one row and columns:

- **AIC**: Akaike’s Information Criterion for the model.
- **BIC**: Bayesian Information Criterion for the model.
- **df.residual**: Residual degrees of freedom.
- **logLik**: The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- **tau**: Quantile.

**See Also**

`glance()`, `quantreg::rq()`

Other `quantreg` tidiers: `augment.nlrq()`, `augment.rqs()`, `augment.rq()`, `glance.nlrq()`, `tidy.nlrq()`, `tidy.rqs()`, `tidy.rq()`

---

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.
Usage

## S3 method for class 'sarlm'

`glance(x, ...)

Arguments

x An object returned from `spatialreg::lagsarlm()` or `spatialreg::errorsarlm()`.

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC** Akaike's Information Criterion for the model.
- **BIC** Bayesian Information Criterion for the model.
- **deviance** Deviance of the model.
- **logLik** The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- **nobs** Number of observations used.

See Also

`glance()`, `spatialreg::lagsarlm()`, `spatialreg::errorsarlm()`, `spatialreg::sacsarlm()`

Other spatialreg tidiers: `augment.sarlm()`, `tidy.sarlm()`

Examples

## Not run:
library(spatialreg)
data(oldcol, package = "spdep")
listw <- spdep::nb2listw(COL.nb, style = "W")

crime_sar <-
  lagsarlm(CRIME ~ INC + HOVAL,
          data = COL.OLD,
          listw = listw,
          method = "eigen")

tidy(crime_sar)
tidy(crime_sar, conf.int = TRUE)

  glance(crime_sar)
  augment(crime_sar)
crime_sem <- errorsarlm(CRIME ~ INC + HOVAL, data = COL.OLD, listw)
tidy(crime_sem)
tidy(crime_sem, conf.int = TRUE)

crime_sac <- sacsarlm(CRIME ~ INC + HOVAL, data = COL.OLD, listw)
tidy(crime_sac)
tidy(crime_sac, conf.int = TRUE)

Description
Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage
## S3 method for class 'smooth.spline'

Arguments
x 
A smooth.spline object returned from stats::smooth.spline().

... 
Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf lvl = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value
A tibble::tibble() with exactly one row and columns:

crit 
Minimized criterion
cv.crit  Cross-validation score  
df  Degrees of freedom used by the model.  
lambda  Choice of lambda corresponding to 'spar'.  
nobs  Number of observations used.  
pen.crit  Penalized criterion.  
spar  Smoothing parameter.  

See Also

`augment()`, `stats::smooth.spline()`  
Other smoothing spline tidiers: `augment.smooth.spline()`  

Examples

```r
spl <- smooth.spline(mtcars$wt, mtcars$mpg, df = 4)
augment(spl, mtcars)
augment(spl) # calls original columns x and y

library(ggplot2)
ggplot(augment(spl, mtcars), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))
```

---

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

**Usage**

```r
## S3 method for class 'speedglm'

### S3 method for class 'speedglm'

```
Arguments

- **x**: A `speedglm` object returned from `speedglm::speedglm()`.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC**: Akaike's Information Criterion for the model.
- **BIC**: Bayesian Information Criterion for the model.
- **deviance**: Deviance of the model.
- **df.null**: Degrees of freedom used by the null model.
- **df.residual**: Residual degrees of freedom.
- **logLik**: The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- **nobs**: Number of observations used.
- **null.deviance**: Deviance of the null model.

See Also

- `speedglm::speedlm()`
- Other `speedlm` tidiers: `augment.speedlm()`, `glance.speedlm()`, `tidy.speedglm()`, `tidy.speedlm()`

Examples

```r
if (requireNamespace("speedglm", quietly = TRUE)) {

  library(speedglm)

  clotting <- data.frame(
    u = c(5, 10, 15, 20, 30, 40, 60, 80, 100),
    lot1 = c(118, 58, 42, 35, 27, 25, 21, 19, 18)
  )

  fit <- speedglm(lot1 ~ log(u), data = clotting, family = Gamma(log))

  tidy(fit)
  glance(fit)
}
```
Glance at a(n) speedlm object

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'speedlm'
glance(x, ...)
```

Arguments

- `x`: A speedlm object returned from speedglm::speedlm().
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with exactly one row and columns:

- `adj.r.squared`: Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
- `AIC`: Akaike's Information Criterion for the model.
- `BIC`: Bayesian Information Criterion for the model.
- `deviance`: Deviance of the model.
- `df`: Degrees of freedom used by the model.
- `df.residual`: Residual degrees of freedom.
logLik
nobs
p.value
r.squared
statistic

The log-likelihood of the model. [stats::logLik()] may be a useful reference.
Number of observations used.
P-value corresponding to the test statistic.
R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
F-statistic.

See Also
speedglm::speedlm()
Other speedlm tidiers: augment.speedlm(), glance.speedglm(), tidy.speedglm(), tidy.speedlm()

Examples

if (requireNamespace("speedglm", quietly = TRUE)) {
  mod <- speedglm::speedlm(mpg ~ wt + qsec, data = mtcars, fitted = TRUE)
  tidy(mod)
  glance(mod)
  augment(mod)
}

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.
Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.
Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.
Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

## S3 method for class 'summary.lm'
glance(x, ...)
Arguments

`x`  
An `lm` object created by `stats::lm()`.

`...`  
Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

The `glance.summary.lm()` method is a potentially useful alternative to `glance.lm()`. For instance, if users have already converted large `lm` objects into their leaner `summary.lm` equivalents to conserve memory. Note, however, that this method does not return all of the columns of the non-summary method (e.g. AIC and BIC will be missing.)

Value

A `tibble::tibble()` with exactly one row and columns:

- `adj.r.squared`: Adjusted R squared statistic, which is like the R squared statistic except taking degrees of freedom into account.
- `df.residual`: Residual degrees of freedom.
- `nobs`: Number of observations used.
- `p.value`: P-value corresponding to the test statistic.
- `r.squared`: R squared statistic, or the percent of variation explained by the model. Also known as the coefficient of determination.
- `sigma`: Estimated standard error of the residuals.
- `statistic`: Test statistic.
- `df`: The degrees for freedom from the numerator of the overall F-statistic. This is new in broom 0.7.0. Previously, this reported the rank of the design matrix, which is one more than the numerator degrees of freedom of the overall F-statistic.

See Also

`glance()`, `glance.summary.lm()`

Other lm tidiers: `augment.glm()`, `augment.lm()`, `glance.glm()`, `glance.lm()`, `glance.svyglm()`, `tidy.glm()`, `tidy.lm.beta()`, `tidy.lm()`, `tidy.mlm()`, `tidy.summary.lm()`

Examples

```r
library(ggplot2)
library(dplyr)
```
```r
mod <- lm(mpg ~ wt + qsec, data = mtcars)
tidy(mod)
glance(mod)

# coefficient plot
d <- tidy(mod, conf.int = TRUE)
ggplot(d, aes(estimate, term, xmin = conf.low, xmax = conf.high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0, lty = 4) +
  geom_errorbarh()

# Aside: There are tidy() and glance() methods for lm.summary objects too.
# This can be useful when you want to conserve memory by converting large lm
# objects into their leaner summary.lm equivalents.
s <- summary(mod)
tidy(s, conf.int = TRUE)
glance(s)
augment(mod)
augment(mod, mtcars, interval = "confidence")

# predict on new data
newdata <- mtcars %>%
  head(6) %>%
  mutate(wt = wt + 1)
augment(mod, newdata = newdata)

# ggplot2 example where we also construct 95% prediction interval
mod2 <- lm(mpg ~ wt, data = mtcars) ## simpler bivariate model since we're plotting in 2D
au <- augment(mod2, newdata = newdata, interval = "prediction")
ggplot(au, aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted)) +
  geom_ribbon(aes(ymin = .lower, ymax = .upper), col = NA, alpha = 0.3)

# predict on new data without outcome variable. Output does not include .resid
newdata <- newdata %>%
  select(-mpg)
augment(mod, newdata = newdata)
au <- augment(mod, data = mtcars)
ggplot(au, aes(.hat, .std.resid)) +
  geom_vline(size = 2, colour = "white", xintercept = 0) +
  geom_hline(size = 2, colour = "white", yintercept = 0) +
  geom_point() +
  geom_smooth(se = FALSE)

plot(mod, which = 6)
```
```r
ggplot(au, aes(.hat, .cooksd)) + geom_vline(xintercept = 0, colour = NA) + geom_abline(slope = seq(0, 3, by = 0.5), colour = "white") + geom_smooth(se = FALSE) + geom_point()

# column-wise models
a <- matrix(rnorm(20), nrow = 10)
b <- a + rnorm(length(a))
result <- lm(b ~ a)
tidy(result)
```

glance.survdiff

Glance at a(n) survdiff object

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

Usage

```r
## S3 method for class 'survdiff'
glance(x, ...)
```

Arguments

- **x**: An `survdiff` object returned from `survival::survdiff()`.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with exactly one row and columns:

- `df` Degrees of freedom used by the model.
- `p.value` P-value corresponding to the test statistic.
- `statistic` Test statistic.

See Also

glance(), `survival::survdiff()`

Other `survdiff` tidiers: `tidy.survdiff()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {
  library(survival)
  s <- survdiff(
    Surv(time, status) ~ pat.karno + strata(inst),
    data = lung
  )
  tidy(s)
  glance(s)
}
```

Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.
Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'survexp'

_glance(x, ...)
```

Arguments

- `x`: An `survexp` object returned from `survival::survexp()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `n.max`: Maximum number of subjects at risk.
- `n.start`: Initial number of subjects at risk.
- `timepoints`: Number of timepoints.

See Also

- `glance()`, `survival::survexp()`
- Other `survexp` tidiers: `tidy.survexp()`
- Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {

  library(survival)
  sexpfit <- survexp(
    futime ~ 1,
    rmap = list(
      sex = "male",
      year = accept.dt,
      age = (accept.dt - birth.dt)
  )
```

glance.survfit

Glance at a(n) survfit object

Description

Glance accepts a model object and returns a tibble::tibble() with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'survfit'
glance(x, ...)
```

Arguments

- `x` An survfit object returned from survival::survfit().
- `...` Additional arguments passed to summary.survfit(). Important arguments include rmean.

Value

A tibble::tibble() with exactly one row and columns:

- `events` Number of events.
- `n.max` Maximum number of subjects at risk.
- `n.start` Initial number of subjects at risk.
nobs Number of observations used.
records Number of observations
rmean Restricted mean (see [survival::print.survfit()]).
rmean.std.error Restricted mean standard error.
conf.low lower end of confidence interval on median
conf.high upper end of confidence interval on median
median median survival

See Also
glance(), survival::survfit()

Other cch tidiers: glance.cch(), tidy.cch()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(),
  glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survreg(),
  tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(),
  tidy.survfit(), tidy.survreg()

Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {

library(survival)
cfit <- coxph(Surv(time, status) ~ age + sex, lung)
sfit <- survfit(cfit)
tidy(sfit)
glance(sfit)

library(ggplot2)
ggplot(tidy(sfit), aes(time, estimate)) +
  geom_line() +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)

# multi-state
fitCI <- survfit(Surv(stop, status * as.numeric(event), type = "mstate") ~ 1,
  data = mgus1, subset = (start == 0)
)

td_multi <- tidy(fitCI)
td_multi
ggplot(td_multi, aes(time, estimate, group = state)) +
  geom_line(aes(color = state)) +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)
}
```
**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.

**Usage**

```r
## S3 method for class 'survreg'
glance(x, ...)
```

**Arguments**

- `x` An `survreg` object returned from `survival::survreg()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Value**

A `tibble::tibble()` with exactly one row and columns:

- **AIC** Akaike's Information Criterion for the model.
- **BIC** Bayesian Information Criterion for the model.
- **df** Degrees of freedom used by the model.
- **df.residual** Residual degrees of freedom.
- **iter** Iterations of algorithm/fitting procedure completed.
- **logLik** The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- **nobs** Number of observations used.
- **p.value** P-value corresponding to the test statistic.
- **statistic** Chi-squared statistic.
See Also

```
glance(), survival::survreg()
```

Other survreg tidiers: `augment.survreg()`, `tidy.survreg()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {

library(survival)

sr <- survreg(
  Surv(futime, fustat) ~ ecog.ps + rx, 
  ovarian, 
  dist = "exponential"
)

tidy(sr)
augment(sr, ovarian)
glance(sr)

# coefficient plot
td <- tidy(sr, conf.int = TRUE)
library(ggplot2)
ggplot(td, aes(estimate, term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) +
  geom_vline(xintercept = 0)
}
```

---

**glance.svyglm**

*Glance at a(n) svyglm object*

**Description**

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.
Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'svyglm'
glance(x, maximal = x, ...)
```

Arguments

- `x` A svyglm object returned from `survey::svyglm()`.
- `maximal` A svyglm object corresponding to the maximal model against which to compute the BIC. See Lumley and Scott (2015) for details. Defaults to `x`, which is equivalent to not using a maximal model.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- **AIC** Akaike’s Information Criterion for the model.
- **BIC** Bayesian Information Criterion for the model.
- **deviance** Deviance of the model.
- **df.null** Degrees of freedom used by the null model.
- **df.residual** Residual degrees of freedom.
- **null.deviance** Deviance of the null model.

References


See Also

`survey::svyglm()`, `stats::glm()`, `survey::anova.svyglm`  
Other lm tidiers: `augment.glm()`, `augment.lm()`, `glance.glm()`, `glance.lm()`, `glance.summary.lm()`, `tidy.glm()`, `tidy.lm.beta()`, `tidy.lm()`, `tidy.mlm()`, `tidy.summary.lm()`
Examples

```r
if (requireNamespace("survey", quietly = TRUE)) {

  library(survey)
  set.seed(123)
  data(api)

  # survey design
  dstrat <-
    svydesign(
      id = ~1,
      strata = ~stype,
      weights = ~pw,
      data = apistrat,
      fpc = ~fpc
    )

  # model
  m <- survey::svyglm(
    formula = sch.wide ~ ell + meals + mobility,
    design = dstrat,
    family = quasibinomial()
  )

  glance(m)
}
```

---

### glance.svyolr

**Glance at an** svyolr **object**

#### Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as `NA`.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an `NA` of the appropriate type.
Usage

## S3 method for class 'svyolr'

```r
glance(x, ...)
```

Arguments

- `x` A `svyolr` object returned from `survey::svyolr()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `df.residual` Residual degrees of freedom.
- `edf` The effective degrees of freedom.
- `nobs` Number of observations used.

See Also

`tidy, survey::svyolr()`

Other ordinal tidiers: `augment.clm()`, `augment.polr()`, `glance.clmm()`, `glance.clm()`, `glance.polr()`, `tidy.clmm()`, `tidy.clm()`, `tidy.polr()`, `tidy.svyolr()`

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {

library(MASS)

fit <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)

tidy(fit, exponentiate = TRUE, conf.int = TRUE)
glance(fit)
}
```
Description

Glance accepts a model object and returns a `tibble::tibble()` with exactly one row of model summaries. The summaries are typically goodness of fit measures, p-values for hypothesis tests on residuals, or model convergence information.

Glance never returns information from the original call to the modeling function. This includes the name of the modeling function or any arguments passed to the modeling function.

Glance does not calculate summary measures. Rather, it farms out these computations to appropriate methods and gathers the results together. Sometimes a goodness of fit measure will be undefined. In these cases the measure will be reported as NA.

Glance returns the same number of columns regardless of whether the model matrix is rank-deficient or not. If so, entries in columns that no longer have a well-defined value are filled in with an NA of the appropriate type.

Usage

```r
## S3 method for class 'varest'

glance(x, ...)
```

Arguments

- `x` A varest object produced by a call to `vars::VAR()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with exactly one row and columns:

- `lag.order` Lag order.
- `logLik` The log-likelihood of the model. `[stats::logLik()]` may be a useful reference.
- `n` The total number of observations.
- `nobs` Number of observations used.

See Also

`glance()`, `vars::VAR()`
Examples

```r
if (requireNamespace("vars", quietly = TRUE)) {

library(vars)
data("Canada", package = "vars")

mod <- VAR(Canada, p = 1, type = "both")
tidy(mod)
glance(mod)
}
```

---

### Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, `stats::optim()`, `svd()` and `akima::interp()` produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves are implemented as functions of the form `tidy_<function>` or `glance_<function>` and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.

### Usage

```r
glance_optim(x, ...)
```

### Arguments

- **x**
  - A list returned from `stats::optim()`.
- **...**
  - Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.
Value

A `tibble::tibble()` with exactly one row and columns:

- `convergence` Convergence code.
- `function.count` Number of calls to ‘fn‘.
- `gradient.count` Number of calls to ‘gr‘.
- `value` Minimized or maximized output value.

See Also

- `glance()`, `stats::optim()`
- Other list tidiers: `list_tidiers`, `tidy_irlba()`, `tidy_optim()`, `tidy_svd()`, `tidy_xyz()`

Examples

```r
o <- optim(c(1, 1, 1), f)
```

Description

For models that have only a single component, the `tidy()` and `glance()` methods are identical. Please see the documentation for both of those methods.

Usage

```r
## S3 method for class 'leveneTest'
tidy(x, ...)
```

Arguments

- `x` An object of class anova created by a call to `car::leveneTest()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvl = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
list_tidiers

Value

A `tibble::tibble()` with columns:

- **df**: Degrees of freedom used by this term in the model.
- **p.value**: The two-sided p-value associated with the observed statistic.
- **statistic**: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- **df.residual**: Residual degrees of freedom.

See Also

- `tidy()`, `glance()`, `car::leveneTest()`
- Other car tidiers: `durbinWatsonTest_tidiers`

Examples

```r
if (requireNamespace("car", quietly = TRUE)) {

  library(car)
  data(Moore)
  lt <- with(Moore, leveneTest(conformity, fcategory))
  tidy(lt)
  glance(lt) # same output for all leveneTest
}
```

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, `stats::optim()`, `base::svd()` and `akima::interp()` produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

Usage

```r
## S3 method for class 'list'
tidy(x, ...)

## S3 method for class 'list'
glance(x, ...)
```

Arguments

- **x**: A list, potentially representing an object that can be tidied.
- **...**: Additionally, arguments passed to the tidying function.
Details

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves are implemented as functions of the form tidy_<function> or glance_<function> and are not exported (but they are documented!). If no appropriate tidying method is found, throws an error.

See Also

Other list tidiers: glance_optim(), tidy_irlba(), tidy_optim(), tidy_svd(), tidy_xyz()

null_tidiers  

Tidiers for NULL inputs

Description

tidy(NULL), glance(NULL) and augment(NULL) all return an empty tibble::tibble. This empty tibble can be treated a tibble with zero rows, making it convenient to combine with other tibbles using functions like purrr::map_df() on lists of potentially NULL objects.

Usage

## S3 method for class `NULL`
tidy(x, ...)

## S3 method for class `NULL`
glance(x, ...)

## S3 method for class `NULL`
augment(x, ...)

Arguments

x  
The value NULL.

...  
Additional arguments (not used).

Value

An empty tibble::tibble.

See Also

tibble::tibble
Tidy a sparseMatrix object from the Matrix package

Description

Tidy a sparseMatrix object from the Matrix package into a three-column data frame, row, column, and value (with zeros missing). If there are row names or column names, use those, otherwise use indices.

Usage

```
## S3 method for class 'dgTMatrix'
tidy(x, ...)

## S3 method for class 'dgCMatrix'
tidy(x, ...)

## S3 method for class 'sparseMatrix'
tidy(x, ...)
```

Arguments

- **x**: A Matrix object
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- **row**: Row ID of the original observation.
- **value**: The value/estimate of the component. Results from data reshaping.
- **column**: Column name in the original matrix.
Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Note that the sf package now defines tidy spatial objects and is the recommended approach to spatial data. sp tidiers are likely to be deprecated in the near future in favor of sf::st_as_sf(). Development of sp tidiers has halted in broom.

Usage

```r
## S3 method for class 'SpatialPolygonsDataFrame'
tidy(x, region = NULL, ...)

## S3 method for class 'SpatialPolygons'
tidy(x, ...)

## S3 method for class 'Polygons'
tidy(x, ...)

## S3 method for class 'Polygon'
tidy(x, ...)

## S3 method for class 'SpatialLinesDataFrame'
tidy(x, ...)

## S3 method for class 'Lines'
tidy(x, ...)

## S3 method for class 'Line'
tidy(x, ...)
```

Arguments

- **x**: A SpatialPolygonsDataFrame, SpatialPolygons, Polygons, Polygon, SpatialLinesDataFrame, Lines or Line object.
- **region**: name of variable used to split up regions
- **...**: not used by this method
### Description

Tidiers for `summaryDefault` objects have been deprecated as of broom 0.7.0 in favor of `skimr::skim()`.

### Usage

```r
## S3 method for class 'summaryDefault'
tidy(x, ...)

## S3 method for class 'summaryDefault'
glance(x, ...)
```

### Arguments

- `x` A `summaryDefault` object, created by calling `summary()` on a vector.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

### Value

A one-row `tibble::tibble` with columns:

- `minimum` Minimum value in original vector.
- `q1` First quartile of original vector.
- `median` Median of original vector.
- `mean` Mean of original vector.
- `q3` Third quartile of original vector.
- `maximum` Maximum value in original vector.
- `na` Number of `NA` values in original vector. Column present only when original vector had at least one `NA` entry.

### See Also

Other deprecated: `bootstrap()`, `confint_tidy()`, `data.frame_tidiers`, `finish_glance()`, `fix_data_frame()`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`, `tidy.numeric()`
Examples

```r
v <- rnorm(1000)
s <- summary(v)
s
 tidy(s)
glance(s)

v2 <- c(v,NA)
tidy(summary(v2))
```

tidy.aareg

**Tidy an aareg object**

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'aareg'
tidy(x, ...)
```

Arguments

- `x`: An aareg object returned from `survival::aareg()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `. . .`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Details

- `robust.se` is only present when `x` was created with `dfbeta = TRUE`. 
Value

A `tibble::tibble()` with columns:

- **estimate**: The estimated value of the regression term.
- **p.value**: The two-sided p-value associated with the observed statistic.
- **robust.se**: robust version of standard error estimate.
- **statistic**: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- **std.error**: The standard error of the regression term.
- **term**: The name of the regression term.
- **z**: z score.

See Also

- `tidy()`, `survival::aareg()`
- Other aareg tidiers: `glance.aareg()`
- Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.surveexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.surveexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {

library(survival)

afit <- aareg(
  Surv(time, status) ~ age + sex + ph.ecog,
  data = lung,
  dfbeta = TRUE
)

tidy(afit)
}
```

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
Usage

```r
## S3 method for class 'acf'
tidy(x, ...)
```

Arguments

- `x`: An `acf` object created by `stats::acf()`, `stats::pacf()` or `stats::ccf()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvl = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble` with columns:

- `acf`: Autocorrelation.
- `lag`: Lag values.

See Also

- `tidy()`, `stats::acf()`, `stats::pacf()`, `stats::ccf()`
- Other time series tidiers: `tidy.spec()`, `tidy.ts()`, `tidy.zoo()`

Examples

```r
tidy(acf(lh, plot = FALSE))
tidy(ccf(mdeaths, fdeaths, plot = FALSE))
tidy(pacf(lh, plot = FALSE))
```

Tidy a(n) anova object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'anova'
tidy(x, ...)
```
Arguments

- **x**
  - An anova object, such as those created by `stats::anova()`, `car::Anova()`, or `car::leveneTest()`.

- **...**
  - Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in **...** where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

The term column of an ANOVA table can come with leading or trailing whitespace, which this tidying method trims.

For documentation on the tidier for `car::leveneTest()` output, see `tidy.leveneTest()`

Value

A `tibble::tibble()` with columns:

- **df**
  - Degrees of freedom used by this term in the model.

- **meansq**
  - Mean sum of squares. Equal to total sum of squares divided by degrees of freedom.

- **p.value**
  - The two-sided p-value associated with the observed statistic.

- **statistic**
  - The value of a T-statistic to use in a hypothesis that the regression term is non-zero.

- **sumsq**
  - Sum of squares explained by this term.

- **term**
  - The name of the regression term.

See Also

`tidy()`, `stats::anova()`, `car::Anova()`, `car::leveneTest()`

Other anova tidiers: `glance.aov()`, `tidy.TukeyHSD()`, `tidy.aovlist()`, `tidy.aov()`, `tidy.manova()`

Examples

```r
a <- lm(mpg ~ wt + qsec + disp, mtcars)
b <- lm(mpg ~ wt + qsec, mtcars)
tidy(anova(a, b))
```
Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'aov'
tidy(x, ...)
```

Arguments

- `x` An `aov` object, such as those created by `stats::aov()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

The term column of an ANOVA table can come with leading or trailing whitespace, which this tidying method trims.

For documentation on the tidier for `car::leveneTest()` output, see `tidy.leveneTest()`

See Also

`tidy(), stats::aov()`

Other anova tidiers: `glance.aov(), tidy.TukeyHSD(), tidy.anova(), tidy.aovlist(), tidy.manova()`

Examples

```r
a <- aov(mpg ~ wt + qsec + disp, mtcars)
tidy(a)
```
tidy.aovlist

Tidy a(n) aovlist object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'aovlist'
tidy(x, ...)
```

Arguments

- `x`: An aovlist objects, such as those created by `stats::aov()`.  
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

The `term` column of an ANOVA table can come with leading or trailing whitespace, which this tidying method trims.

For documentation on the tidier for `car::leveneTest()` output, see `tidy.leveneTest()`

Value

A `tibble::tibble()` with columns:

- `df`: Degrees of freedom used by this term in the model.
- `meansq`: Mean sum of squares. Equal to total sum of squares divided by degrees of freedom.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `stratum`: The error stratum.
- `sumsq`: Sum of squares explained by this term.
- `term`: The name of the regression term.
See Also
tidy(), stats::aov()

Other anova tidiers: glance.aov(), tidy.TukeyHSD(), tidy.anova(), tidy.aov(), tidy.manova()

Examples

```r
a <- aov(mpg ~ wt + qsec + Error(disp / am), mtcars)
tidy(a)
```

### tidy.Arima

**Tidy a(n) Arima object**

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'Arima'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

**Arguments**

- `x` An object of class Arima created by stats::arima().
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- `conf.level` The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.
Value

A **tibble::tibble()** with columns:

- **conf.high**: Upper bound on the confidence interval for the estimate.
- **conf.low**: Lower bound on the confidence interval for the estimate.
- **estimate**: The estimated value of the regression term.
- **std.error**: The standard error of the regression term.
- **term**: The name of the regression term.

See Also

**stats::arima()**

Other Arima tidiers: **glance.Arima()**

Examples

```r
fit <- arima(lh, order = c(1, 0, 0))
tidy(fit)
glance(fit)
```

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'betamfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- **x**: A `betamfx` object.
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- **conf.level**: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in . . ., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

The mfx package provides methods for calculating marginal effects for various generalized linear models (GLMs). Unlike standard linear models, estimated model coefficients in a GLM cannot be directly interpreted as marginal effects (i.e., the change in the response variable predicted after a one unit change in one of the regressors). This is because the estimated coefficients are multiplicative, dependent on both the link function that was used for the estimation and any other variables that were included in the model. When calculating marginal effects, users must typically choose whether they want to use i) the average observation in the data, or ii) the average of the sample marginal effects. See vignette("mfixarticle") from the mfx package for more details.

Value

A tibble::tibble() with columns:

- conf.high: Upper bound on the confidence interval for the estimate.
- conf.low: Lower bound on the confidence interval for the estimate.
- estimate: The estimated value of the regression term.
- p.value: The two-sided p-value associated with the observed statistic.
- statistic: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- std.error: The standard error of the regression term.
- term: The name of the regression term.
- atmean: TRUE if the marginal effects were originally calculated as the partial effects for the average observation. If FALSE, then these were instead calculated as average partial effects.

See Also

tidy.betareg(), mfx::betamfx()

Other mfx tidiers: augment.betamfx(), augment.mfx(), glance.betamfx(), glance.mfx(), tidy.mfx()

Examples

```
## Not run:
library(mfx)

## Simulate some data
set.seed(12345)
```
n = 1000
x = rnorm(n)

## Beta outcome
y = rbeta(n, shape1 = plogis(1 + 0.5 * x), shape2 = (abs(0.2*x)))
## Use Smithson and Verkuilen correction
y = (y*(n-1)+0.5)/n
d = data.frame(y,x)
mod_betamfx = betamfx(y ~ x | x, data = d)
tidy(mod_betamfx, conf.int = TRUE)

## Compare with the naive model coefficients of the equivalent betareg call (not run)
# tidy(betamfx(y ~ x | x, data = d), conf.int = TRUE)

augment(mod_betamfx)

## End(Not run)

---

**tidy.betareg**  
*Tidy a(n) betareg object*

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'betareg'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

**Arguments**

- **x**  
  A betareg object produced by a call to `betareg::betareg()`.
- **conf.int**  
  Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- **conf.level**  
  The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **...**  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed
using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a newdata argument, it will use the default value for the data argument.

**Details**

The tibble has one row for each term in the regression. The component column indicates whether a particular term was used to model either the "mean" or "precision". Here the precision is the inverse of the variance, often referred to as phi. At least one term will have been used to model the precision phi.

**Value**

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.
- `component`: Whether a particular term was used to model the mean or the precision in the regression. See details.

**See Also**

`tidy()`, `betareg::betareg()`

**Examples**

```r
if (requireNamespace("betareg", quietly = TRUE)) {

library(betareg)
data("GasolineYield", package = "betareg")

mod <- betareg(yield ~ batch + temp, data = GasolineYield)

mod
tidy(mod)
tidy(mod, conf.int = TRUE)
tidy(mod, conf.int = TRUE, conf.level = .99)

augment(mod)

glance(mod)
}
```
tidy.biglm

Tidy a(n) biglm object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'biglm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

- `x` A biglm object created by a call to `biglm::biglm()` or `biglm::bigglm()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `exponentiate` Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `conf.high` Upper bound on the confidence interval for the estimate.
- `conf.low` Lower bound on the confidence interval for the estimate.
- `estimate` The estimated value of the regression term.
- `p.value` The two-sided p-value associated with the observed statistic.
### tidy.binDesign

Tidy a(n) binDesign object

#### Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

#### Examples

```r
## Not run:
library(biglm)

bfit <- biglm(mpg ~ wt + disp, mtcars)
tidy(bfit)
tidy(bfit, conf.int = TRUE)
tidy(bfit, conf.int = TRUE, conf.level = .9)

# bigglm: logistic regression
bgfit <- bigglm(bin ~ mpg, mtcars, family = binomial())
tidy(bgfit)
tidy(bgfit, exponentiate = TRUE)
tidy(bgfit, conf.int = TRUE)
tidy(bgfit, conf.int = TRUE, conf.level = .9)
tidy(bgfit, conf.int = TRUE, conf.level = .9, exponentiate = TRUE)

# bigglm: logistic regression
bgfit <- bigglm(bin ~ mpg, mtcars, family = binomial())
tidy(bgfit)

# End(Not run)
```

<table>
<thead>
<tr>
<th>statistic</th>
<th>The value of a T-statistic to use in a hypothesis that the regression term is non-zero.</th>
</tr>
</thead>
<tbody>
<tr>
<td>std.error</td>
<td>The standard error of the regression term.</td>
</tr>
<tr>
<td>term</td>
<td>The name of the regression term.</td>
</tr>
</tbody>
</table>

#### See Also

`tidy()`, `biglm::biglm()`, `biglm::bigglm()`

Other biglm tidiers: `glance.biglm()`
tidy.binDesign

Usage

```r
## S3 method for class 'binDesign'
tidy(x, ...)
```

Arguments

- `x`: A `binGroup::binDesign()` object.
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with columns:

- `n`: Number of trials in given iteration.
- `power`: Power achieved for given value of `n`.

See Also

- `tidy()`, `binGroup::binDesign()`
- Other bingroup tidiers: `glance.binDesign()`, `tidy.binWidth()`

Examples

```r
if (requireNamespace("binGroup", quietly = TRUE)) {

library(binGroup)
des <- binDesign(
  nmax = 300, delta = 0.06,
  p.hyp = 0.1, power = .8
)

glance(des)
tidy(des)

# the ggplot2 equivalent of plot(des)
library(ggplot2)
ggplot(tidy(des), aes(n, power)) +
  geom_line()
}
```
Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'binWidth'
 tidy(x, ...)
```

Arguments

- `x`: A `binGroup::binWidth()` object.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with columns:

- `alternative`: Alternative hypothesis (character).
- `ci.width`: Expected width of confidence interval.
- `p`: True proportion.
- `n`: Total sample size

See Also

- `tidy()`, `binGroup::binWidth()`
- Other bingroup tidiers: `glance.binDesign()`, `tidy.binDesign()`

Examples

```r
if (requireNamespace("binGroup", quietly = TRUE)) {
  library(binGroup)
  library(dplyr)
}```
library(ggplot2)

bw <- binWidth(100, .1)
bw
tidy(bw)

---

**tidy.boot**  
*Tidy a(n) boot object*

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'boot'
tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  conf.method = c("perc", "bca", "basic", "norm"),
  ...
)
```

**Arguments**

- **x**  
  A *boot::boot()* object.

- **conf.int**  
  Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.

- **conf.level**  
  The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

- **conf.method**  
  Passed to the *type* argument of *boot::boot.ci()*. Defaults to "perc". The allowed types are "perc", "basic", "bca", and "norm". Does not support "stud" or "all".

- **...**  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ... where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an *augment()* method that does not accept a `newdata` argument, it will use the default value for the data argument.
Details

If weights were provided to the boot function, an estimate column is included showing the weighted bootstrap estimate, and the standard error is of that estimate.

If there are no original statistics in the "boot" object, such as with a call to tsboot with orig.t = FALSE, the original and statistic columns are omitted, and only estimate and std.error columns shown.

Value

A tibble::tibble() with columns:

- bias: Bias of the statistic.
- std.error: The standard error of the regression term.
- term: The name of the regression term.
- statistic: Original value of the statistic.

See Also

tidy(), boot::boot(), boot::tsboot(), boot::boot.ci(), rsample::bootstraps()

Examples

```r
if (requireNamespace("boot", quietly = TRUE)) {

library(boot)

clotting <- data.frame(
  u = c(5, 10, 15, 20, 30, 40, 60, 80, 100),
  lot1 = c(118, 58, 42, 35, 27, 25, 21, 19, 18),
  lot2 = c(69, 35, 26, 21, 18, 16, 13, 12, 12)
)

g1 <- glm(lot2 ~ log(u), data = clotting, family = Gamma)

bootfun <- function(d, i) {
  coef(update(g1, data = d[i, ]))
}

bootres <- boot(clotting, bootfun, R = 999)
tidy(g1, conf.int = TRUE)
tidy(bootres, conf.int = TRUE)
}
```
Tidy a(n) btergm object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

This method tidies the coefficients of a bootstrapped temporal exponential random graph model estimated with the xergm. It simply returns the coefficients and their confidence intervals.

Usage

```r
## S3 method for class 'btergm'
tidy(x, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

- `x`: A `btergm::btergm()` object.
- `conf.level`: Confidence level for confidence intervals. Defaults to 0.95.
- `exponentiate`: Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `term`: The name of the regression term.

See Also

tidy(), btergm::btergm()
Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'cch'
tidy(x, conf.level = 0.95, ...)
```

Arguments

- `x` An `cch` object returned from `survival::cch()`.
- `conf.level` confidence level for CI
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `conf.high` Upper bound on the confidence interval for the estimate.
- `conf.low` Lower bound on the confidence interval for the estimate.
- `estimate` The estimated value of the regression term.
- `p.value` The two-sided p-value associated with the observed statistic.
- `statistic` The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error` The standard error of the regression term.
- `term` The name of the regression term.
See Also

`tidy()`, `survival::cch()`

Other cch tidiers: `glance.cch()`, `glance.survfit()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survecp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survecp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {

library(survival)

# examples come from cch documentation
subcoh <- nwtco$in.subcohort
selccoh <- with(nwtco, rel == 1 | subcoh == 1)
ccoh.data <- nwtco[selccoh,]
ccoh.data$subcohort <- subcoh[selccoh]
## central-lab histology
ccoh.data$histol <- factor(ccoh.data$histol, labels = c("FH", "UH"))
## tumour stage
ccoh.data$stage <- factor(ccoh.data$stage, labels = c("I", "II", "III", "IV"))
ccoh.data$age <- ccoh.data$age / 12 # Age in years

fit.ccP <- cch(Surv(edrel, rel) ~ stage + histol + age,
data = ccoh.data,
subcoh = ~subcohort, id = ~seqno, cohort.size = 4028)

tidy(fit.ccP)

# coefficient plot
library(ggplot2)
ggplot(tidy(fit.ccP), aes(x = estimate, y = term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) +
  geom_vline(xintercept = 0)
}
```

tidy.cld  
_Tidy a(n) cld object_
Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'cld'
 tidy(x, ...)
```

Arguments

- `x` A `cld` object created by calling `multcomp::cld()` on a `glht`, `confint.glht()` or `summary.glht()` object.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `contrast` Levels being compared.
- `letters` Compact letter display denoting all pair-wise comparisons.

See Also

`tidy()`, `multcomp::cld()`, `multcomp::summary.glht()`, `multcomp::confint.glht()`, `multcomp::glht()`

Other `multcomp` tidiers: `tidy.confint.glht()`, `tidy.glht()`, `tidy.summary.glht()`

Examples

```r
if (requireNamespace("multcomp", quietly = TRUE)) {

library(multcomp)
library(ggplot2)

amod <- aov(breaks ~ wool + tension, data = warpbreaks)
wht <- glht(amod, linfct = mcp(tension = "Tukey"))

tidy(wht)
ggplot(wht, aes(lhs, estimate)) +
  geom_point()
```
tidy.clm

Tidy a(n) clm object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'clm'
tidy(
x, conf.int = FALSE, conf.level = 0.95, conf.type = c("profile", "Wald"), exponentiate = FALSE,
...
)
```

Arguments

- `x` A clm object returned from `ordinal::clm()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
conf.type  Whether to use "profile" or "Wald" confidence intervals, passed to the type
argument of ordinal::confint.clm(). Defaults to "profile".
exponentiate Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
...
Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

In broom 0.7.0 the coefficient_type column was renamed to coef.type, and the contents were changed as well.

Note that intercept type coefficients correspond to alpha parameters, location type coefficients correspond to beta parameters, and scale type coefficients correspond to zeta parameters.

Value

A tibble::tibble() with columns:

- conf.high Upper bound on the confidence interval for the estimate.
- conf.low Lower bound on the confidence interval for the estimate.
- estimate The estimated value of the regression term.
- p.value The two-sided p-value associated with the observed statistic.
- statistic The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- std.error The standard error of the regression term.
- term The name of the regression term.

See Also

tidy.ordinal::clm(), ordinal::confint.clm()

Other ordinal tidiers: augment.clm(), augment.polr(), glance.clmm(), glance.clm(), glance.polr(), glance.svyolr(), tidy.clmm(), tidy.polr(), tidy.svyolr()

Examples

```r
if (requireNamespace("ordinal", quietly = TRUE)) {

library(ordinal)

fit <- clm(rating ~ temp * contact, data = wine)
```
tidy(fit)
tidy(fit, conf.int = TRUE, conf.level = 0.9)
tidy(fit, conf.int = TRUE, conf.type = "Wald", exponentiate = TRUE)

glance(fit)
augment(fit, type.predict = "prob")
augment(fit, type.predict = "class")

fit2 <- clm(rating ~ temp, nominal = ~contact, data = wine)
tidy(fit2)
glance(fit2)

---

**tidy.clmm**

Tidy a(n) clmm object

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'clmm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

**Arguments**

- **x**
  - A clmm object returned from `ordinal::clmm()`.
- **conf.int**
  - Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- **conf.level**
  - The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **exponentiate**
  - Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
- **...**
  - Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.

Note

In broom 0.7.0 the `coefficient_type` column was renamed to `coef.type`, and the contents were changed as well.

Note that intercept type coefficients correspond to `alpha` parameters, location type coefficients correspond to `beta` parameters, and scale type coefficients correspond to `zeta` parameters.

See Also

`tidy.ordinal::clmm()`, `ordinal::confint.clm()`

Other ordinal tidiers: `augment.clm()`, `augment.polr()`, `glance.clmm()`, `glance.clm()`, `glance.polr()`, `glance.svyolr()`, `tidy.clm()`, `tidy.polr()`, `tidy.svyolr()`

Examples

```r
if (requireNamespace("ordinal", quietly = TRUE)) {

library(ordinal)

fit <- clmm(rating ~ temp + contact + (1 | judge), data = wine)

tidy(fit)
tidy(fit, conf.int = TRUE, conf.level = 0.9)
tidy(fit, conf.int = TRUE, exponentiate = TRUE)

glance(fit)

fit2 <- clmm(rating ~ temp + (1 | judge), nominal = ~contact, data = wine)
tidy(fit2)

glance(fit2)
}
```
Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'coeftest'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x`: A coeftest object returned from `lmtest::coeftest()`.
- `conf.int`: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- `conf.level`: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.
See Also

`tidy()`, `lmtest::coeftest()`

Examples

```r
if (requireNamespace("lmtest", quietly = TRUE)) {

library(lmtest)

m <- lm(dist ~ speed, data = cars)

coeftest(m)
tidy(coeftest(m))
tidy(coeftest(m, conf.int = TRUE))

# A very common workflow is to combine lmtest::coeftest with alternate
# variance-covariance matrices via the sandwich package. The lmtest
# tidiers support this workflow too, enabling you to adjust the standard
# errors of your tidied models on the fly.
library(sandwich)
tidy(coeftest(m, vcov = vcovHC))  # "HC3" (default) robust SEs
tidy(coeftest(m, vcov = vcovHC, type = "HC2"))  # "HC2" robust SEs

tidy(coeftest(m, vcov = NeweyWest))  # N-W HAC robust SEs

# The columns of the returned tibble for glance.coeftest() will vary
# depending on whether the coeftest object retains the underlying model.
# Users can control this with the "save = TRUE" argument of coeftest().


glance(coeftest(m))
glance(coeftest(m, save = TRUE))  # More columns
}
```

--

tidy.confint.glht  
*Tidy a(n) confint.glht object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'confint.glht'
tidy(x, ...)
```
Arguments

- `x` A `confint.glht` object created by calling `multcomp::confint.glht()` on a `glht` object created with `multcomp::glht()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with columns:

- `conf.high` Upper bound on the confidence interval for the estimate.
- `conf.low` Lower bound on the confidence interval for the estimate.
- `contrast` Levels being compared.
- `estimate` The estimated value of the regression term.

See Also

- `tidy()`, `multcomp::confint.glht()`, `multcomp::glht()`
- Other multcomp tidiers: `tidy.cld()`, `tidy.glht()`, `tidy.summary.glht()`

Examples

```r
if (requireNamespace("multcomp", quietly = TRUE)) {

library(multcomp)
library(ggplot2)

amod <- aov(breaks ~ wool + tension, data = warpbreaks)
wh <- glht(amod, linfct = mcp(tension = "Tukey"))

 tidy(wh)
 ggplot(wh, aes(lhs, estimate)) +
 geom_point()

 CI <- confint(wh)
 tidy(CI)
 ggplot(CI, aes(lhs, estimate, ymin = lwr, ymax = upr)) +
 geom_pointrange()

 tidy(summary(wh))
 ggplot(mapping = aes(lhs, estimate)) +
 geom_linerange(aes(ymin = lwr, ymax = upr), data = CI) +
 geom_point(aes(size = p), data = summary(wh)) +

```
tidy.confusionMatrix

Tidy a(n) confusionMatrix object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'confusionMatrix'
tidy(x, by_class = TRUE, ...)
```

Arguments

- `x`: An object of class `confusionMatrix` created by a call to `caret::confusionMatrix()`.
- `by_class`: Logical indicating whether or not to show performance measures broken down by class. Defaults to `TRUE`. When `by_class = FALSE` only returns a `tibble` with accuracy, kappa, and McNemar statistics.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `class`: The class under consideration.
- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `term`: The name of the regression term.
- `p.value`: P-value for accuracy and kappa statistics.
See Also

tidy(), caret::confusionMatrix()

Examples

```r
if (requireNamespace("caret", quietly = TRUE)) {

  library(caret)

  set.seed(27)

  two_class_sample1 <- as.factor(sample(letters[1:2], 100, TRUE))
  two_class_sample2 <- as.factor(sample(letters[1:2], 100, TRUE))

  two_class_cm <- caret::confusionMatrix(
    two_class_sample1,
    two_class_sample2
  )

  tidy(two_class_cm)
  tidy(two_class_cm, by_class = FALSE)

  # multiclass example

  six_class_sample1 <- as.factor(sample(letters[1:6], 100, TRUE))
  six_class_sample2 <- as.factor(sample(letters[1:6], 100, TRUE))

  six_class_cm <- caret::confusionMatrix(
    six_class_sample1,
    six_class_sample2
  )

  tidy(six_class_cm)
  tidy(six_class_cm, by_class = FALSE)
}
```

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
Usage

```r
## S3 method for class 'coxph'
tidy(x, exponentiate = FALSE, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- **x**: A `coxph` object returned from `survival::coxph()`.
- **exponentiate**: Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- **conf.level**: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with columns:

- **estimate**: The estimated value of the regression term.
- **p.value**: The two-sided p-value associated with the observed statistic.
- **statistic**: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- **std.error**: The standard error of the regression term.

See Also

`tidy()`, `survival::coxph()`

Other coxph tidiers: `augment.coxph()`, `glance.coxph()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`.

```
tidy.aareg(), tidy.cch(), tidy.pyears(), tidy.survdiff(), tidy.survexp(),
tidy.survfit(), tidy.survreg()
```
Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {

library(survival)

cfit <- coxph(Surv(time, status) ~ age + sex, lung)

tidy(cfit)
tidy(cfit, exponentiate = TRUE)

lp <- augment(cfit, lung)
risks <- augment(cfit, lung, type.predict = "risk")
expected <- augment(cfit, lung, type.predict = "expected")

glance(cfit)

# also works on clogit models
resp <- levels(logan$occupation)
n <- nrow(logan)
indx <- rep(1:n, length(resp))
logan2 <- data.frame(
    logan[indx, ],
    id = indx,
    tocc = factor(rep(resp, each = n))
)

logan2$case <- (logan2$occupation == logan2$tocc)

cl <- clogit(case ~ tocc + tocc:education + strata(id), logan2)
tidy(cl)
glance(cl)

library(ggplot2)

ggplot(lp, aes(age, .fitted, color = sex)) +
    geom_point()

ggplot(risks, aes(age, .fitted, color = sex)) +
    geom_point()

ggplot(expected, aes(time, .fitted, color = sex)) +
    geom_point()
}
```
Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'crr'
tidy(x, exponentiate = FALSE, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x`: A `crr` object returned from `cmprsk::crr()`.
- `exponentiate`: Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
- `conf.int`: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level`: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than `0` and less than `1`. Defaults to `0.95`, which corresponds to a 95 percent confidence interval.
- `...`: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.

See Also

`tidy()`, `cmprsk::crr()`

Other cmprsk tidiers: `glance.crr()`
Examples

```r
if (requireNamespace("cmprsk", quietly = TRUE)) {

library(cmprsk)
lrf_time <- rexp(100) # time to loco-regional failure (lrf)
lrf_event <- sample(0:2, 100, replace = TRUE)
trt <- sample(0:1, 100, replace = TRUE)
strt <- sample(1:2, 100, replace = TRUE)
x <- crr(lrf_time, lrf_event, cbind(trt, strt))
tidy(x, conf.int = TRUE)
glance(x)
}
```

### tidy.cv.glmnet

**Tidy a(n) cv.glmnet object**

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'cv.glmnet'
tidy(x, ...)
```

**Arguments**

- `x` A `cv.glmnet` object returned from `glmnet::cv.glmnet()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

**Value**

A `tibble::tibble()` with columns:

- `lambda` Value of penalty parameter lambda.
- `nzero` Number of non-zero coefficients for the given lambda.
std.error The standard error of the regression term.
conf.low lower bound on confidence interval for cross-validation estimated loss.
conf.high upper bound on confidence interval for cross-validation estimated loss.
estimate Median loss across all cross-validation folds for a given lambda

See Also

tidy().glmnet::cv.glmnet()
Other glmnet tidiers: glance.cv.glmnet(), glance.glmnet(), tidy.glmnet()

Examples

if (requireNamespace("glmnet", quietly = TRUE)) {

library(glmnet)
set.seed(27)

nobs <- 100
nvar <- 50
real <- 5
x <- matrix(rnorm(nobs * nvar), nobs, nvar)
beta <- c(rnorm(real, 0, 1), rep(0, nvar - real))
y <- c(t(beta) %*% t(x)) + rnorm(nvar, sd = 3)

CVfit1 <- cv.glmnet(x, y)
tidy(CVfit1)
glance(CVfit1)

library(ggplot2)
tidied_cv <- tidy(CVfit1)
glance_cv <- glance(CVfit1)

# plot of MSE as a function of lambda
g <- ggplot(tidied_cv, aes(lambda, estimate)) +
  geom_line() +
  scale_x_log10()
g

# plot of MSE as a function of lambda with confidence ribbon

g <- g + geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)
g

# plot of MSE as a function of lambda with confidence ribbon and choices
# of minimum lambda marked

g <- g +
  geom_vline(xintercept = glance_cv$lambda.min) +
  geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
g
tidy.density

(Deprecated) Tidy density objects

Description

(Deprecated) Tidy density objects

Usage

## S3 method for class 'density'
tidy(x, ...)

Arguments

x

A density object returned from stats::density().

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble with two columns: points x where the density is estimated, and estimated density y.

See Also

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, finish_glance(), fix_data_frame(), summary_tidiers, tidy.dist(), tidy.ftable(), tidy.numeric()
tidy.dist  (Deprecated) Tidy dist objects

Description

(Deprecated) Tidy dist objects

Usage

## S3 method for class 'dist'
tidy(x, diagonal = attr(x, "Diag"), upper = attr(x, "Upper"), ...)

Arguments

x      A dist object returned from stats::dist().
diagonal Logical indicating whether or not to tidy the diagonal elements of the distance matrix. Defaults to whatever was based to the diag argument of stats::dist().
upper  Logical indicating whether or not to tidy the upper half of the distance matrix. Defaults to whatever was based to the upper argument of stats::dist().
...

Additional arguments. Not used. Needed to match generic signature only.  

Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

If the distance matrix does not include an upper triangle and/or diagonal, the tidied version will not either.

Value

A tibble::tibble with one row for each pair of items in the distance matrix, with columns:

  item1    First item
  item2    Second item
distance  Distance between items

See Also

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, finish_glance(), fix_data_frame(), summary_tidiers, tidy.density(), tidy.ftable(), tidy.numeric()
Examples

cars_dist <- dist(t(mtcars[, 1:4]))
cars_dist
tidy(cars_dist)
tidy(cars_dist, upper = TRUE)
tidy(cars_dist, diagonal = TRUE)

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'drc'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

- **x**
  A drc object produced by a call to drc::drm().

- **conf.int**
  Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.

- **conf.level**
  The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

- **...**
  Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

The tibble has one row for each curve and term in the regression. The curveid column indicates the curve.
Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.
- `curve`: Index identifying the curve.

See Also
tidy(), drc::drm()

Other drc tidiers: augment.drc(), glance.drc()

Examples

```r
if (requireNamespace("drc", quietly = TRUE)) {

library(drc)

mod <- drm(dead / total ~ conc, type,
  weights = total, data = selenium, fct = LL.2(), type = "binomial"
)

tidy(mod)
tidy(mod, conf.int = TRUE)

glance(mod)

augment(mod, selenium)
}
```

tidy.emmGrid  
*Tidy a(n) emmGrid object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
tidy.emmGrid

Usage

```r
## S3 method for class 'emmGrid'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x`: An `emmGrid` object.
- `conf.int`: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level`: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...`: Additional arguments passed to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`.

Cautionary note: misspecified arguments may be silently ignored!

Details

Returns a data frame with one observation for each estimated marginal mean, and one column for each combination of factors. When the input is a contrast, each row will contain one estimated contrast.

There are a large number of arguments that can be passed on to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`.

Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `df`: Degrees of freedom used by this term in the model.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `std.error`: The standard error of the regression term.
- `estimate`: Expected marginal mean
- `statistic`: T-ratio statistic

See Also

- `tidy()`, `emmeans::ref_grid()`, `emmeans::emmeans()`, `emmeans::contrast()`

Other emmeans tidiers: `tidy.lsmobj()`, `tidy.ref.grid()`, `tidy.summary_emm()`
Examples

```r
if (requireNamespace("emmeans", quietly = TRUE)) {

  library(emmeans)
  # linear model for sales of oranges per day
  oranges_lm1 <- lm(sales1 ~ price1 + price2 + day + store, data = oranges)

  # reference grid; see vignette("basics", package = "emmeans")
  oranges_rg1 <- ref_grid(oranges_lm1)
  td <- tidy(oranges_rg1)
  td

  # marginal averages
  marginal <- emmeans(oranges_rg1, "day")
  tidy(marginal)

  # contrasts
  tidy(contrast(marginal))
  tidy(contrast(marginal, method = "pairwise"))

  # plot confidence intervals
  library(ggplot2)
  ggplot(tidy(marginal, conf.int = TRUE), aes(day, estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

  # by multiple prices
  by_price <- emmeans(oranges_lm1, "day",
    by = "price2",
    at = list(
      price1 = 50, price2 = c(40, 60, 80),
      day = c("2", "3", "4")
    )
  )
  by_price
  tidy(by_price)

  ggplot(tidy(by_price, conf.int = TRUE), aes(price2, estimate, color = day)) +
  geom_line() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

  # joint_tests
  tidy(joint_tests(oranges_lm1))
}
```

**tidy.epi.2by2**

Tidy an *epi.2by2* object
**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'epi.2by2'
tidy(x, parameters = c("moa", "stat"), ...)
```

**Arguments**

- `x` A `epi.2by2` object produced by a call to `epiR::epi.2by2()`
- `parameters` Return measures of association (moa) or test statistics (stat), default is moa (measures of association)
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Details**

The tibble has a column for each of the measures of association or tests contained in `massoc` or `massoc.detail` when `epiR::epi.2by2()` is called.

**Value**

A `tibble::tibble()` with columns:

- `conf.high` Upper bound on the confidence interval for the estimate.
- `conf.low` Lower bound on the confidence interval for the estimate.
- `df` Degrees of freedom used by this term in the model.
- `p.value` The two-sided p-value associated with the observed statistic.
- `statistic` The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `term` The name of the regression term.
- `estimate` Estimated measure of association

**See Also**

`tidy()`, `epiR::epi.2by2()`
Examples

```r
if (requireNamespace("epiR", quietly = TRUE)) {

library(epiR)

dat <- matrix(c(13, 2163, 5, 3349), nrow = 2, byrow = TRUE)

rownames(dat) <- c("DF+", "DF-")
colnames(dat) <- c("FUS+", "FUS-")

fit <- epi.2by2(
  dat = as.table(dat), method = "cross.sectional",
  conf.level = 0.95, units = 100, outcome = "as.columns"
)

tidy(fit, parameters = "moa")
tidy(fit, parameters = "stat")
}
```

tidy.ergm

Tidy a(n) ergm object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

The methods should work with any model that conforms to the `ergm` class, such as those produced from weighted networks by the `ergm.count` package.

Usage

```r
## S3 method for class 'ergm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

- `x`: An `ergm` object returned from a call to `ergm::ergm()`.
- `conf.int`: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level`: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
tidy.ergm

```r
exponentiate Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.

... Additional arguments to pass to `ergm::summary()`. **Cautionary note:** Mis-specified arguments may be silently ignored.

**Value**

A `tibble::tibble` with one row for each coefficient in the exponential random graph model, with columns:

<table>
<thead>
<tr>
<th>term</th>
<th>The term in the model being estimated and tested</th>
</tr>
</thead>
<tbody>
<tr>
<td>estimate</td>
<td>The estimated coefficient</td>
</tr>
<tr>
<td>std.error</td>
<td>The standard error</td>
</tr>
<tr>
<td>mcmc.error</td>
<td>The MCMC error</td>
</tr>
<tr>
<td>p.value</td>
<td>The two-sided p-value</td>
</tr>
</tbody>
</table>

**References**


**See Also**

`tidy()`, `ergm::ergm()`, `ergm::control.ergm()`, `ergm::summary()`

Other `ergm` tidiers: `glance.ergm()`

**Examples**

```r
if (requireNamespace("ergm", quietly = TRUE)) {

library(ergm)
# Using the same example as the ergm package
# Load the Florentine marriage network data
data(florentine)

# Fit a model where the propensity to form ties between families depends on the absolute difference in wealth
gest <- ergm(floomarriage ~ edges + absdiff("wealth"))

# Show terms, coefficient estimates and errors
tidy(gest)

# Show coefficients as odds ratios with a 99% CI
tidy(gest, exponentiate = TRUE, conf.int = TRUE, conf.level = 0.99)

# Take a look at likelihood measures and other
```
tidy.factanal

Tidy a(n) factanal object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'factanal'
tidy(x, ...)
```

Arguments

x
A factanal object created by `stats::factanal()`.

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `variable` Variable under consideration.
- `uniqueness` Proportion of residual, or unexplained variance
- `flX` Factor loading for level X.

See Also

`tidy()`, `stats::factanal()`

Other factanal tidiers: `augment.factanal()`, `glance.factanal()`
set.seed(123)

# data
m1 <- dplyr::tibble(
  v1 = c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 4, 5, 6),
  v2 = c(1, 2, 1, 1, 1, 2, 1, 2, 1, 3, 4, 3, 3, 3, 4, 6, 5),
  v3 = c(3, 3, 3, 3, 3, 1, 1, 1, 1, 1, 1, 1, 1, 1, 5, 4, 6),
  v4 = c(3, 3, 4, 3, 3, 1, 2, 1, 1, 1, 2, 1, 1, 5, 6, 4),
  v5 = c(1, 1, 1, 1, 1, 3, 3, 3, 3, 3, 1, 1, 1, 1, 6, 4, 5),
  v6 = c(1, 1, 1, 2, 1, 3, 3, 3, 4, 3, 1, 1, 1, 2, 1, 6, 5, 4)
)

# new data
m2 <- purrr::map_dfr(m1, rev)

# factor analysis objects
fit1 <- stats::factanal(m1, factors = 3, scores = "Bartlett")
fit2 <- stats::factanal(m1, factors = 3, scores = "regression")

# tidying the object
tidy(fit1)
tidy(fit2)

# augmented dataframe
augment(fit1)
augment(fit2)

# augmented dataframe (with new data)
augment(fit1, data = m2)
augment(fit2, data = m2)

---

**tidy.felm**  
*Tidy a(n) felm object*

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'felm'
tidy(
  x,  
  conf.int = FALSE,
)```

conf.level = 0.95,
fe = FALSE,
se.type = c("default", "iid", "robust", "cluster"),
...
)

Arguments

x
A `felm` object returned from `lfe::felm()`.

conf.int
Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.

conf.level
The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

fe
Logical indicating whether or not to include estimates of fixed effects. Defaults to FALSE.

se.type
Character indicating the type of standard errors. Defaults to using those of the underlying `felm()` model object, e.g. clustered errors for models that were provided a cluster specification. Users can override these defaults by specifying an appropriate alternative: "iid" (for homoskedastic errors), "robust" (for Eicker-Huber-White robust errors), or "cluster" (for clustered standard errors; if the model object supports it).

...
Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

conf.high
Upper bound on the confidence interval for the estimate.

conf.low
Lower bound on the confidence interval for the estimate.

estimate
The estimated value of the regression term.

p.value
The two-sided p-value associated with the observed statistic.

statistic
The value of a T-statistic to use in a hypothesis that the regression term is nonzero.

std.error
The standard error of the regression term.

term
The name of the regression term.

See Also

tidy(), lfe::felm()
Other felm tidiers: augment.felm()
Examples

```r
if (requireNamespace("lfe", quietly = TRUE)) {

library(lfe)

# Use built-in "airquality" dataset
head(airquality)

# No FEs; same as lm()
est0 <- felm(Ozone ~ Temp + Wind + Solar.R, airquality)
tidy(est0)
augment(est0)

# Add month fixed effects
est1 <- felm(Ozone ~ Temp + Wind + Solar.R | Month, airquality)
tidy(est1)
tidy(est1, fe = TRUE)
augment(est1)
glance(est1)

# The "se.type" argument can be used to switch out different standard errors
# types on the fly. In turn, this can be useful exploring the effect of
# different error structures on model inference.
tidy(est1, se.type = "iid")
tidy(est1, se.type = "robust")

# Add clustered SEs (also by month)
est2 <- felm(Ozone ~ Temp + Wind + Solar.R | Month | 0 | Month, airquality)
tidy(est2, conf.int = TRUE)
tidy(est2, conf.int = TRUE, se.type = "cluster")
tidy(est2, conf.int = TRUE, se.type = "robust")
tidy(est2, conf.int = TRUE, se.type = "iid")
}
```

tidy.fitdistr

Tidy an fitdistr object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'fitdistr'
tidy(x, ...)
```
tidy.fixest

Arguments

x  A fitdistr object returned by MASS::fitdistr().
...

Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:

- estimate: The estimated value of the regression term.
- std.error: The standard error of the regression term.
- term: The name of the regression term.

See Also
tidy(), MASS::fitdistr()

Other fitdistr tidiers: glance.fitdistr()

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {
  set.seed(2015)
  x <- rnorm(100, 5, 2)
  library(MASS)
  fit <- fitdistr(x, dnorm, list(mean = 3, sd = 1))
  tidy(fit)
  glance(fit)
}
```

tidy.fixest  Tidy a(n) fixest object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
tidy.fixest

Usage

```r
## S3 method for class 'fixest'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x` A `fixest` object returned from any of the `fixest` estimators
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...` Additional arguments passed to `summary` and `confint`. Important arguments are `se` and `cluster`. Other arguments are `dof`, `exact_dof`, `forceCovariance`, and `keepBounded`. See `summary.fixest`.

Details

The `fixest` package provides a family of functions for estimating models with arbitrary numbers of fixed-effects, in both an OLS and a GLM context. The package also supports robust (i.e. White) and clustered standard error reporting via the generic `summary.fixest()` command. In a similar vein, the `tidy()` method for these models allows users to specify a desired standard error correction either 1) implicitly via the supplied `fixest` object, or 2) explicitly as part of the `tidy` call. See examples below.

Note that `fixest` confidence intervals are calculated assuming a normal distribution – this assumes infinite degrees of freedom for the CI. (This assumption is distinct from the degrees of freedom used to calculate the standard errors. For more on degrees of freedom with clusters and fixed effects, see [https://github.com/lrberge/fixest/issues/6](https://github.com/lrberge/fixest/issues/6) and [https://github.com/sgaure/lfe/issues/1#issuecomment-530646990](https://github.com/sgaure/lfe/issues/1#issuecomment-530646990))

Value

A `tibble::tibble()` with columns:

- `conf.high` Upper bound on the confidence interval for the estimate.
- `conf.low` Lower bound on the confidence interval for the estimate.
- `estimate` The estimated value of the regression term.
- `p.value` The two-sided p-value associated with the observed statistic.
- `statistic` The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error` The standard error of the regression term.
- `term` The name of the regression term.
See Also

`tidy()`, `fixest::feglm()`, `fixest::fenegbin()`, `fixest::feNmlm()`, `fixest::femlm()`, `fixest::feols()`, `fixest::fepois()`

Other fixest tidiers: `augment.fixest()`

Examples

```r
if (requireNamespace("fixest", quietly = TRUE)) {

library(fixest)

gravity <- feols(log(Euros) ~ log(dist_km) | Origin + Destination + Product + Year, trade)

tidy(gravity)
glance(gravity)
augment(gravity, trade)

## To get robust or clustered SEs, users can either:
# 1) Or, specify the arguments directly in the tidy() call

tidy(gravity, conf.int = TRUE, cluster = c("Product", "Year"))
tidy(gravity, conf.int = TRUE, se = "threeway")

# 2) Feed tidy() a summary.fixest object that has already accepted these arguments

grid_month <- summary(gravity, cluster = c("Product", "Year"))
tidy(grid_month, conf.int = TRUE)

# Approach (1) is preferred.

}
```

---

tidy.ftable  (Deprecated) Tidy ftable objects

Description

This function is deprecated. Please use `tibble::as_tibble()` instead.

Usage

```r
## S3 method for class 'ftable'
tidy(x, ...)
```
Arguments

x

An ftable object returned from stats::ftable().

...

Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

An ftable contains a "flat" contingency table. This melts it into a tibble::tibble with one column for each variable, then a Freq column.

See Also

Other deprecated: bootstrap(), confint_tidy(), data.frame_tidiers, finish_glance(), fix_data_frame(), summary_tidiers, tidy.density(), tidy.dist(), tidy.numeric()
Details

Tidy gam objects created by calls to `mgcv::gam()` with `tidy.gam()`.

Value

A `tibble::tibble()` with columns:

- `df`: Degrees of freedom used by this term in the model.
- `meansq`: Mean sum of squares. Equal to total sum of squares divided by degrees of freedom.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `sumsq`: Sum of squares explained by this term.
- `term`: The name of the regression term.

See Also

`tidy()`, `gam::gam()`, `tidy.anova()`, `tidy.gam()`

Other gam tidiers: `glance.Gam()`

Examples

```r
if (requireNamespace("gam", quietly = TRUE)) {

library(gam)
g <- gam(mpg ~ s(hp, 4) + am + qsec, data = mtcars)
tidy(g)glance(g)
}
```

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
Usage

## S3 method for class 'gam'
tidy(
  x,
  parametric = FALSE,
  conf.int = FALSE,
  conf.level = 0.95,
  exponentiate = FALSE,
  ...
)

Arguments

x A gam object returned from a call to mgcv::gam().
parametric Logical indicating if parametric or smooth terms should be tidied. Defaults to FALSE, meaning that smooth terms are tidied by default.
conf.int Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
...

Details

When parametric = FALSE return columns edf and ref.df rather than estimate and std.error.

Value

A tibble::tibble() with columns:
estimate The estimated value of the regression term.
p.value The two-sided p-value associated with the observed statistic.
statistic The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error The standard error of the regression term.
term The name of the regression term.
Tidy a(n) garch object

Description
Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'garch'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

x        A garch object returned by \texttt{tseries::garch()}.
conf.int Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level The confidence level to use for the confidence interval if \texttt{conf.int = TRUE}. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:

- conf.high: Upper bound on the confidence interval for the estimate.
- conf.low: Lower bound on the confidence interval for the estimate.
- estimate: The estimated value of the regression term.
- p.value: The two-sided p-value associated with the observed statistic.
- statistic: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- std.error: The standard error of the regression term.
- term: The name of the regression term.

See Also

tidy(), tseries::garch()

Other garch tidiers: glance.garch()

Examples

```r
if (requireNamespace("tseries", quietly = TRUE)) {

library(tseries)

data(EuStockMarkets)
dax <- diff(log(EuStockMarkets))[, "DAX"]
dax.garch <- garch(dax)
dax.garch

tidy(dax.garch)

}
Tidy a(n) geeglm object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'geeglm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

- `x` A geeglm object returned from a call to `geepack::geeglm()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `exponentiate` Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ... where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Details

If `conf.int = TRUE`, the confidence interval is computed with the an internal `confint.geeglm()` function.

If you have missing values in your model data, you may need to refit the model with `na.action = na.exclude` or deal with the missingness in the data beforehand.

Value

A `tibble::tibble()` with columns:

- `conf.high` Upper bound on the confidence interval for the estimate.
tidy.glht

 conf.low  Lower bound on the confidence interval for the estimate.
estimate  The estimated value of the regression term.
p.value  The two-sided p-value associated with the observed statistic.
statistic  The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error  The standard error of the regression term.
term  The name of the regression term.

See Also

tidy(), geepack::geeglm()

Examples

if (requireNamespace("geepack", quietly = TRUE)) {
  library(geepack)
data(state)
  ds <- data.frame(state.region, state.x77)
  geefit <- geeglm(Income ~ Frost + Murder,
                   id = state.region,
                   data = ds, family = gaussian,
                   corstr = "exchangeable")
  tidy(geefit)
  tidy(geefit, conf.int = TRUE)
}

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'glht'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
Arguments

- **x**: A glht object returned by `multcomp::glht()`.
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- **conf.level**: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- **contrast**: Levels being compared.
- **estimate**: The estimated value of the regression term.
- **null.value**: Value to which the estimate is compared.

See Also

- `tidy()`, `multcomp::glht()`
- Other multcomp tidiers: `tidy.cld()`, `tidy.confint.glht()`, `tidy.summary.glht()`

Examples

```r
if (requireNamespace("multcomp", quietly = TRUE)) {

library(multcomp)
library(ggplot2)

amod <- aov(breaks ~ wool + tension, data = warpbreaks)
wht <- glht(amod, linfct = mcp(tension = "Tukey"))

 tidy(wht)
 ggplot(wht, aes(lhs, estimate)) + geom_point()

 CI <- confint(wht)
tidy(CI)
 ggplot(CI, aes(lhs, estimate, ymin = lwr, ymax = upr)) + geom_pointrange()

tidy(summary(wht))

```
```r
ggplot(mapping = aes(lhs, estimate)) +
  geom_linerange(aes(ymin = lwr, ymax = upr), data = CI) +
  geom_point(aes(size = p), data = summary(wht)) +
  scale_size(trans = "reverse")

cld <- cld(wht)
tidy(cld)
```

---

**tidy.glm**

*Tidy a(n) glm object*

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'glm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

**Arguments**

- `x` A `glm` object returned from `stats::glm()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `exponentiate` Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
See Also

stats::glm()

Other lm tidiers: augment.glm(), augment.lm(), glance.glm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.lm.beta(), tidy.lm(), tidy.mlm(), tidy.summary.lm()

tidy.glmnet

Tidy a(n) glmnet object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'glmnet'
tidy(x, return_zeros = FALSE, ...)

Arguments

x A glmnet object returned from glmnet::glmnet().

return_zeros Logical indicating whether coefficients with value zero should be included in the results. Defaults to FALSE.

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ... where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.1vel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

Note that while this representation of GLMs is much easier to plot and combine than the default structure, it is also much more memory-intensive. Do not use for large, sparse matrices.

No augment method is yet provided even though the model produces predictions, because the input data is not tidy (it is a matrix that may be very wide) and therefore combining predictions with it is not logical. Furthermore, predictions make sense only with a specific choice of lambda.
Value

A `tibble::tibble()` with columns:

- `dev.ratio`: Fraction of null deviance explained at each value of lambda.
- `estimate`: The estimated value of the regression term.
- `lambda`: Value of penalty parameter lambda.
- `step`: Which step of lambda choices was used.
- `term`: The name of the regression term.

See Also

- `tidy()`
- `glmnet::glmnet()`

Other glmnet tidiers: `glance.cv.glmnet()`, `glance.glmnet()`, `tidy.cv.glmnet()`

Examples

```r
if (requireNamespace("glmnet", quietly = TRUE)) {

library(glmnet)
set.seed(2014)
x <- matrix(rnorm(100 * 20), 100, 20)
y <- rnorm(100)
fit1 <- glmnet(x, y)
tidy(fit1)
glance(fit1)

library(dplyr)
library(ggplot2)
tidied <- tidy(fit1) %>% filter(term != "(Intercept)")
ggplot(tidied, aes(step, estimate, group = term)) + geom_line()
ggplot(tidied, aes(lambda, estimate, group = term)) + geom_line() + scale_x_log10()
ggplot(tidied, aes(lambda, dev.ratio)) + geom_line()

# works for other types of regressions as well, such as logistic
g2 <- sample(1:2, 100, replace = TRUE)
fit2 <- glmnet(x, g2, family = "binomial")
tidy(fit2)
}
```
tidy.glmRob  

_Tidy a(n) glmRob object_

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'glmRob'
tidy(x, ...)
```

**Arguments**

- `x`  
  A glmRob object returned from `robust::glmRob()`.

- `...`  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

**Details**

For tidiers for robust models from the MASS package see `tidy.rlm()`.

**See Also**

`robust::glmRob()`

Other robust tidiers: `augment.lmRob()`, `glance.glmRob()`, `glance.lmRob()`, `tidy.lmRob()`

**Examples**

```r
if (requireNamespace("robust", quietly = TRUE)) {
  library(robust)

  gm <- glmRob(am ~ wt, data = mtcars, family = "binomial")

  tidy(gm)
  glance(gm)
}
```
Tidy a(n) glmrob object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'glmrob'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x`: A glmrob object returned from `robustbase::glmrob()`.
- `conf.int`: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level`: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

For tidiers for robust models from the **MASS** package see `tidy.rlm()`.

Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.
See Also

robustbase::glmrob()

Other robustbase tidiers: augment.glmrob(), augment.lmrob(), glance.lmrob(), tidy.lmrob()

Examples

library(robustbase)
# From the robustbase::lmrob examples:
data(coleman)
set.seed(0)

m <- robustbase::lmrob(Y ~ ., data = coleman)
tidy(m)
augment(m)
glance(m)

# From the robustbase::glmrob examples:
data(carrots)
Rfit <- glmrob(cbind(success, total - success) ~ logdose + block,
               family = binomial, data = carrots, method = "Mqle",
               control = glmrobMqle.control(tcc = 1.2))
tidy(Rfit)
augment(Rfit)

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'gmm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)

Arguments

x

A gmm object returned from gmm::gmm().

conf.int

Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level

The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

exponentiate

Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.

...

Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.

See Also

`tidy()`, `gmm::gmm()`

Other gmm tidiers: `glance.gmm()`

Examples

```r
if (requireNamespace("gmm", quietly = TRUE)) {

library(gmm)

# examples come from the "gmm" package
## CAPM test with GMM
data(Finance)
$r <- Finance[1:300, 1:10]
rm <- Finance[1:300, "rm"]
rf <- Finance[1:300, "rf"]

z <- as.matrix(r - rf)
t <- nrow(z)
```
zm <- rm - rf
h <- matrix(zm, t, 1)
res <- gmm(z ~ zm, x = h)

# tidy result
 tidy(res)
tidy(res, conf.int = TRUE)
tidy(res, conf.int = TRUE, conf.level = .99)

# coefficient plot
library(ggplot2)
library(dplyr)

tidy(res, conf.int = TRUE) %>%
  mutate(variable = reorder(term, estimate)) %>%
  ggplot(aes(estimate, variable)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0, color = "red", lty = 2)

# from a function instead of a matrix
g <- function(theta, x) {
  gmat <- cbind(e, e * c(x[, 1]))
  return(gmat)
}
x <- as.matrix(cbind(rm, r))
res_black <- gmm(g, x = x, t0 = rep(0, 11))

tidy(res_black)
tidy(res_black, conf.int = TRUE)

## APT test with Fama-French factors and GMM
f1 <- zm
f2 <- Finance[1:300, "hml"] - rf
f3 <- Finance[1:300, "smb"] - rf
h <- cbind(f1, f2, f3)
res2 <- gmm(z ~ f1 + f2 + f3, x = h)
td2 <- tidy(res2, conf.int = TRUE)
td2

# coefficient plot

```r
td2 %>%
  mutate(variable = reorder(term, estimate)) %>%
  ggplot(aes(estimate, variable)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_vline(xintercept = 0, color = "red", lty = 2)
```
Tidy/glance a(n) htest object

Description
For models that have only a single component, the tidy() and glance() methods are identical. Please see the documentation for both of those methods.

Usage

```r
## S3 method for class 'htest'
tidy(x, ...)

## S3 method for class 'htest'
glance(x, ...)
```

Arguments

- `x`: An htest object, such as those created by `stats::cor.test()`, `stats::t.test()`, `stats::wilcox.test()`, `stats::chisq.test()`, etc.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautious note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with columns:

- `alternative`: Alternative hypothesis (character).
- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `estimate1`: Sometimes two estimates are computed, such as in a two-sample t-test.
- `estimate2`: Sometimes two estimates are computed, such as in a two-sample t-test.
- `method`: Method used.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `parameter`: The parameter being modeled.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
See Also

`tidy()`, `stats::cor.test()`, `stats::t.test()`, `stats::wilcox.test()`, `stats::chisq.test()`

Other htest tidiers: `augment.htest()`, `tidy.pairwise.htest()`, `tidy.power.htest()`

Examples

```r
tt <- t.test(rnorm(10))
tidy(tt)

# same output for all htests

tt <- t.test(mpg ~ am, data = mtcars)
tidy(tt)

wt <- wilcox.test(mpg ~ am, data = mtcars, conf.int = TRUE, exact = FALSE)
tidy(wt)

ct <- cor.test(mtcars$wt, mtcars$mpg)
tidy(ct)

chit <- chisq.test(xtabs(Freq ~ Sex + Class, data = as.data.frame(Titanic)))
tidy(chit)
augment(chit)
```

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'ivreg'
tidy(x, conf.int = FALSE, conf.level = 0.95, instruments = FALSE, ...)
```

Arguments

- `x` An ivreg object created by a call to `AER::ivreg()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
instruments Logical indicating whether to return coefficients from the second-stage or diagnostics tests for each endogenous regressor (F-statistics). Defaults to FALSE.

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

This tidier currently only supports ivreg-classed objects outputted by the AER package. The ivreg package also outputs objects of class ivreg, and will be supported in a later release.

Value

A tibble::tibble() with columns:

- conf.high: Upper bound on the confidence interval for the estimate.
- conf.low: Lower bound on the confidence interval for the estimate.
- estimate: The estimated value of the regression term.
- p.value: The two-sided p-value associated with the observed statistic.
- p.value.Sargan: p-value for Sargan test of overidentifying restrictions.
- p.value.weakinst: p-value for weak instruments test.
- statistic: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- statistic.Sargan: Statistic for Sargan test of overidentifying restrictions.
- statistic.weakinst: Statistic for Wu-Hausman test.
- std.error: The standard error of the regression term.
- term: The name of the regression term.

See Also

tidy(), AER::ivreg()

Other ivreg tidiers: augment.ivreg(), glance.ivreg()
Examples

```r
if (requireNamespace("AER", quietly = TRUE)) {

library(AER)

data("CigarettesSW", package = "AER")

ivr <- ivreg(
  log(packs) ~ income | population,
  data = CigarettesSW,
  subset = year == "1995"
)

summary(ivr)

tidy(ivr)
tidy(ivr, conf.int = TRUE)
tidy(ivr, conf.int = TRUE, instruments = TRUE)

augment(ivr)
augment(ivr, data = CigarettesSW)
augment(ivr, newdata = CigarettesSW)

glance(ivr)
}
```

tidy.kappa

Tidy a(n) kappa object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'kappa'
tidy(x, ...)
```

Arguments

- `x` A kappa object returned from `psych::cohen.kappa()`.
Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in . . . where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Details**

Note that confidence level (alpha) for the confidence interval cannot be set in `tidy`. Instead you must set the alpha argument to `psych::cohen.kappa()` when creating the kappa object.

**Value**

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `type`: Either 'weighted' or 'unweighted'.

**See Also**

`tidy()`, `psych::cohen.kappa()`

**Examples**

```r
if (requireNamespace("psych", quietly = TRUE)) {

library(psych)

rater1 <- 1:9
rater2 <- c(1, 3, 1, 6, 1, 5, 5, 6, 7)
ck <- cohen.kappa(cbind(rater1, rater2))
tidy(ck)

# graph the confidence intervals
library(ggplot2)
ggplot(tidy(ck), aes(estimate, type)) + geom_point() + geom_errorbarh(aes(xmin = conf.low, xmax = conf.high))
}
```
Tidy a(n) kde object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'kde'
tidy(x, ...)
```

Arguments

- `x` A kde object returned from `ks::kde()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

Returns a data frame in long format with four columns. Use `tidyr::pivot_wider(..., names_from = variable, values_from = value)` on the output to return to a wide format.

Value

A `tibble::tibble()` with columns:

- `estimate` The estimated value of the regression term.
- `obs` weighted observed number of events in each group.
- `value` The value/estimate of the component. Results from data reshaping.
- `variable` Variable under consideration.

See Also

`tidy(), ks::kde()`
Examples

```r
if (requireNamespace("ks", quietly = TRUE)) {

library(ks)

dat <- replicate(2, rnorm(100))
k <- kde(dat)

td <- tidy(k)

td

library(ggplot2)
library(dplyr)
library(tidyr)

td %>%
  pivot_wider(c(obs, estimate),
              names_from = variable,
              values_from = value
    ) %>%
  ggplot(aes(x1, x2, fill = estimate)) +
  geom_tile() +
  theme_void()

# also works with 3 dimensions

dat3 <- replicate(3, rnorm(100))
k3 <- kde(dat3)

td3 <- tidy(k3)

td3
}
```

**tidy.Kendall**

*Tidy a(n) Kendall object*

### Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

### Usage

```r
## S3 method for class 'Kendall'
tidy(x, ...)
```
Arguments

- **x**
  A Kendall object returned from a call to `Kendall::Kendall()`, `Kendall::MannKendall()`, or `Kendall::SeasonalMannKendall()`.

- **...**
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A **tibble::tibble()** with columns:

- `kendall_score`: Kendall score.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `var_kendall_score`: Variance of the `kendall_score`.
- `statistic`: Kendall’s tau statistic
- `denominator`: The denominator, which is `tau = kendall_score/denominator`.

See Also

`tidy()`, `Kendall::Kendall()`, `Kendall::MannKendall()`, `Kendall::SeasonalMannKendall()`

Examples

```r
if (requireNamespace("Kendall", quietly = TRUE)) {

library(Kendall)

A <- c(2.5, 2.5, 2.5, 2.5, 5, 6.5, 6.5, 10, 10, 10, 10, 10, 14, 14, 14, 16, 17)
B <- c(1, 1, 1, 1, 2, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 2, 2)

f_res <- Kendall(A, B)
tidy(f_res)

s_res <- MannKendall(B)
tidy(s_res)

t_res <- SeasonalMannKendall(ts(A))
tidy(t_res)
}
```
Tidy a(n) kmeans object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'kmeans'
tidy(x, col.names = colnames(x$centers), ...)
```

Arguments

- `x`: A `kmeans` object created by `stats::kmeans()`.
- `col.names`: Dimension names. Defaults to the names of the variables in `x`. Set to NULL to get names x1, x2, ....
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

For examples, see the `kmeans` vignette.

Value

A `tibble::tibble()` with columns:

- `cluster`: A factor describing the cluster from 1:k.
- `size`: Number of points assigned to cluster.
- `withinss`: The within-cluster sum of squares.

See Also

- `tidy()`, `stats::kmeans()`
- Other kmeans tidiers: `augment.kmeans()`, `glance.kmeans()`
Examples

```r
## Not run:
library(cluster)
library(dplyr)
library(modeldata)
data(hpc_data)
x <- hpc_data[, 2:5]
fit <- pam(x, k = 4)
tidy(fit)
glance(fit)
augment(fit, x)
## End(Not run)
```

tidy.lavaan   

Tidy a(n) lavaan object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'lavaan'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x` A lavaan object, such as those returned from `lavaan::cfa()` and `lavaan::sem()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...` Additional arguments passed to `lavaan::parameterEstimates()`. **Cautionary note:** Misspecified arguments may be silently ignored.
Value

A `tibble::tibble()` with one row for each estimated parameter and columns:

- **term**: The result of `paste(lhs, op, rhs)`
- **op**: The operator in the model syntax (e.g. `~~` for covariances, or `~` for regression parameters)
- **group**: The group (if specified) in the lavaan model
- **estimate**: The parameter estimate (may be standardized)
- **std.error**: The standard error
- **statistic**: The z value returned by `lavaan::parameterEstimates()`
- **p.value**: The p-value
- **conf.low**: The lower confidence interval
- **conf.high**: The upper confidence interval
- **std.lv**: Standardized estimates based on the variances of the (continuous) latent variables only
- **std.all**: Standardized estimates based on both the variances of both (continuous) observed and latent variables.
- **std.nox**: Standardized estimates based on both the variances of both (continuous) observed and latent variables, but not the variances of exogenous covariates.

See Also

`tidy()`, `lavaan::cfa()`, `lavaan::sem()`, `lavaan::parameterEstimates()`

Other lavaan tidiers: `glance.lavaan()`

Examples

```r
if (requireNamespace("lavaan", quietly = TRUE)) {

## Not run:
library(lavaan)

cfa.fit <- cfa("F =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9",
    data = HolzingerSwineford1939, group = "school"
)

tidy(cfa.fit)

## End(Not run)
}
```
tidy.lm

_Tidy a(n) lm object_

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'lm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- **x**: An lm object created by `stats::lm()`.
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- **conf.level**: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **...**: Additional arguments. Not used. Needed to match generic signature only. _Cautionary note_: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

If the linear model is an mlm object (multiple linear model), there is an additional column `response`. See `tidy.mlm()`.

Value

A `tibble::tibble()` with columns:

- **conf.high**: Upper bound on the confidence interval for the estimate.
- **conf.low**: Lower bound on the confidence interval for the estimate.
- **estimate**: The estimated value of the regression term.
- **p.value**: The two-sided p-value associated with the observed statistic.
- **statistic**: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error The standard error of the regression term.
term The name of the regression term.

See Also

tidy(), stats::summary.lm()

Other lm tidiers: augment.glm(), augment.lm(), glance.glm(), glance.lm(), glance.summary.lm(),
glance.svyglm(), tidy.glm(), tidy.lm.beta(), tidy.mlm(), tidy.summary.lm()

Examples

library(ggplot2)
library(dplyr)

mod <- lm(mpg ~ wt + qsec, data = mtcars)
tidy(mod)
glance(mod)

# coefficient plot
d <- tidy(mod, conf.int = TRUE)
ggplot(d, aes(estimate, term, xmin = conf.low, xmax = conf.high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0, lty = 4) +
  geom_errorbarh()

# Aside: There are tidy() and glance() methods for lm.summary objects too.
# This can be useful when you want to conserve memory by converting large lm
# objects into their leaner summary.lm equivalents.
s <- summary(mod)
tidy(s, conf.int = TRUE)
glance(s)

augment(mod)
augment(mod, mtcars, interval = "confidence")

# predict on new data
newdata <- mtcars %>%
  head(6) %>%
  mutate(wt = wt + 1)
augment(mod, newdata = newdata)

# ggplot2 example where we also construct 95% prediction interval
mod2 <- lm(mpg ~ wt, data = mtcars) ## simpler bivariate model since we're plotting in 2D
au <- augment(mod2, newdata = newdata, interval = "prediction")
ggplot(au, aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted)) +
tidy.lm.beta

Tidy a(n) lm.beta object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'lm.beta'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

x An lm.beta object created by lm.beta::lm.beta.

conf.int Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
tidy.lm.beta

conf.level  The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

...  Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

If the linear model is an mlm object (multiple linear model), there is an additional column response.

If you have missing values in your model data, you may need to refit the model with na.action = na.exclude.

Value

A tibble::tibble() with columns:

conf.high  Upper bound on the confidence interval for the estimate.
conf.low   Lower bound on the confidence interval for the estimate.
estimate  The estimated value of the regression term.
p.value   The two-sided p-value associated with the observed statistic.
statistic The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error  The standard error of the regression term.
term      The name of the regression term.

See Also

Other lm tidiers: augment.glm(), augment.lm(), glance.glm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm(), tidy.mlm(), tidy.summary.lm()
group <- gl(2, 10, 20, labels = c("Ctl", "Trt"))
weight <- c(ctl, trt)
mod2 <- lm(weight ~ group)
std2 <- lm.beta(mod2)
tidy(std2, conf.int = TRUE)

### S3 method for class 'lmodel2'
tidy(x, ...)

#### Arguments

x | A lmodel2 object returned by lmodel2::lmodel2().

... | Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

#### Details

There are always only two terms in an lmodel2: "Intercept" and "Slope". These are computed by four methods: OLS (ordinary least squares), MA (major axis), SMA (standard major axis), and RMA (ranged major axis).

The returned p-value is one-tailed and calculated via a permutation test. A permutational test is used because distributional assumptions may not be valid. More information can be found in vignette("mod2user", package = "lmodel2").
Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `term`: The name of the regression term.
- `method`: Either OLS/MA/SMA/RMA

See Also

- `tidy()`, `lmodel2::lmodel2()`
- Other lmodel2 tidiers: `glance.lmodel2()`

Examples

```r
if (requireNamespace("lmodel2", quietly = TRUE)) {

library(lmodel2)

data(mod2ex2)
Ex2.res <- lmodel2(Prey ~ Predators, data = mod2ex2, "relative", "relative", 99)
Ex2.res

tidy(Ex2.res)

glance(Ex2.res)

# this allows coefficient plots with ggplot2
library(ggplot2)

ggplot(tidy(Ex2.res), aes(estimate, term, color = method)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high))
}
```

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
Usage

```r
## S3 method for class 'lmRob'
tidy(x, ...)
```

Arguments

- `x` A `lmRob` object returned from `robust::lmRob()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Details

For tidiers for robust models from the `MASS` package see `tidy.rlm()`.

See Also

- `robust::lmRob()`

Other robust tidiers: `augment.lmRob()`, `glance.glmRob()`, `glance.lmRob()`, `tidy.glmRob()`

Examples

```r
if (requireNamespace("robust", quietly = TRUE)) {
  library(robust)
  m <- lmRob(mpg ~ wt, data = mtcars)

  tidy(m)
augment(m)
glance(m)
}
```

---

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
tidy.lmrob

Usage

```r
## S3 method for class 'lmrob'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x` A `lmrob` object returned from `robustbase::lmrob()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

For tidiers for robust models from the `MASS` package see `tidy.rlm()`.

See Also

- `robustbase::lmrob()`
- Other robustbase tidiers: `augment.glmrob()`, `augment.lmrob()`, `glance.lmrob()`, `tidy.glmrob()`

Examples

```r
library(robustbase)
# From the robustbase::lmrob examples:
data(coleman)
set.seed(0)

m <- robustbase::lmrob(Y ~ ., data = coleman)
tidy(m)
augment(m)
glance(m)

# From the robustbase::glmrob examples:
data(carrots)
Rfit <- glmrob(cbind(success, total - success) ~ logdose + block,
               family = binomial, data = carrots, method = "Mqle",
               control = glmrobMqle.control(tcc = 1.2))
tidy(Rfit)
augment(Rfit)
```
tidy.lsmobj

Tidy a(n) lsmobj object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'lsmobj'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

x
An lsmobj object.

conf.int
Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.

conf.level
The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

...
Additional arguments passed to emmeans::summary.emmGrid() or lsmeans::summary.ref.grid()

Cautionary note: misspecified arguments may be silently ignored!

Details

Returns a data frame with one observation for each estimated marginal mean, and one column for each combination of factors. When the input is a contrast, each row will contain one estimated contrast.

There are a large number of arguments that can be passed on to emmeans::summary.emmGrid() or lsmeans::summary.ref.grid().

Value

A tibble::tibble() with columns:

conf.high
Upper bound on the confidence interval for the estimate.

conf.low
Lower bound on the confidence interval for the estimate.

contrast
Levels being compared.

df
Degrees of freedom used by this term in the model.

null.value
Value to which the estimate is compared.

p.value
The two-sided p-value associated with the observed statistic.

std.error
The standard error of the regression term.

estimate
Expected marginal mean

statistic
T-ratio statistic
See Also

tidy(), emmeans::ref_grid(), emmeans::emmeans(), emmeans::contrast()

Other emmeans tidiers: tidy.emmGrid(), tidy.ref.grid(), tidy.summary_emm()

Examples

if (requireNamespace("emmeans", quietly = TRUE)) {

library(emmeans)

# linear model for sales of oranges per day
oranges_lm1 <- lm(sales1 ~ price1 + price2 + day + store, data = oranges)

# reference grid; see vignette("basics", package = "emmeans")
oranges_rg1 <- ref_grid(oranges_lm1)
td <- tidy(oranges_rg1)
td

# marginal averages
marginal <- emmeans(oranges_rg1, "day")
tidy(marginal)

# contrasts
tidy(contrast(marginal))
tidy(contrast(marginal, method = "pairwise"))

# plot confidence intervals
library(ggplot2)
ggplot(tidy(marginal, conf.int = TRUE), aes(day, estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# by multiple prices
by_price <- emmeans(oranges_lm1, "day",
  by = "price2",
  at = list(
    price1 = 50, price2 = c(40, 60, 80),
    day = c("2", "3", "4")
  )
)
by_price

by_price

ggplot(tidy(by_price, conf.int = TRUE), aes(price2, estimate, color = day)) +
  geom_line() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# joint_tests

}
Tidy a(n) manova object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'manova'
tidy(x, test = "Pillai", ...)
```

Arguments

- `x` A manova object return from `stats::manova()`.
- `test` One of "Pillai" (Pillai’s trace), "Wilks" (Wilk’s lambda), "Hotelling-Lawley" (Hotelling-Lawley trace) or "Roy" (Roy’s greatest root) indicating which test statistic should be used. Defaults to "Pillai".
- `...` Arguments passed on to `stats::summary.manova` object

Details

Depending on which test statistic is specified only one of `pillai`, `wilks`, `hl` or `roy` is included.

Value

A `tibble::tibble()` with columns:

- `den.df` Degrees of freedom of the denominator.
- `num.df` Degrees of freedom.
- `p.value` The two-sided p-value associated with the observed statistic.
- `statistic` The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `term` The name of the regression term.
- `pillai` Pillai’s trace.
- `wilks` Wilk’s lambda.
- `hl` Hotelling-Lawley trace.
- `roy` Roy’s greatest root.
tidy.map

See Also
tidy(), stats::summary.manova()
Other anova tidiers: glance.aov(), tidy.TukeyHSD(), tidy.anova(), tidy.aovlist(), tidy.aov()

Examples

npk2 <- within(npk, foo <- rnorm(24))
m <- manova(cbind(yield, foo) ~ block + N * P * K, npk2)
tidy(m)

Description
Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'map'
tidy(x, ...)

Arguments

x A map object returned from maps::map().
...

Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:
term The name of the regression term.
long Longitude.
lata Latitude.

Remaining columns give information on geographic attributes and depend on the inputted map object. See ?maps::map for more information.
See Also

tidy(), maps::map()

Examples

if (requireNamespace("maps", quietly = TRUE)) {
  library(maps)
  library(ggplot2)

  ca <- map("county", "ca", plot = FALSE, fill = TRUE)
  tidy(ca)
  qplot(long, lat, data = ca, geom = "polygon", group = group)

  tx <- map("county", "texas", plot = FALSE, fill = TRUE)
  tidy(tx)
  qplot(long, lat,
        data = tx, geom = "polygon", group = group,
        colour = I("white")
  )
}

 tidy.margins

Tidy a(n) margins object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'margins'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

x A margins object returned from margins::margins().
conf.int Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in . . . where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

**Details**

The `margins` package provides a way to obtain coefficient marginal effects for a variety of (non-linear) models, such as logit or models with multiway interaction terms. Note that the `glance.margins()` method requires rerunning the underlying model again, which can take some time. Similarly, an `augment.margins()` method is not currently supported, but users can simply run the underlying model to obtain the same information.

**Value**

A tibble::tibble() with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.

**See Also**

`tidy()`,

```
margins::margins()
```

**Examples**

```r
if (requireNamespace("margins", quietly = TRUE)) {

  library(margins)

  ## Example 1: Logit model ##
  mod_log <- glm(am ~ cyl + hp + wt, data = mtcars, family = binomial)

  # Get tidied "naive" model coefficients
  tidy(mod_log)

  # Convert to marginal effects with margins::margins()
  marg_log <- margins(mod_log)
  ```
# Get tidied marginal effects
tidy(marg_log)
tidy(marg_log, conf.int = TRUE)

# Example 2: Threeway interaction terms

mod_ie <- lm(mpg ~ wt * cyl * disp, data = mtcars)

# Get tidied "naive" model coefficients
tidy(mod_ie)

# Convert to marginal effects with margins::margins()
marg_ie0 <- margins(mod_ie)

# Marginal effects evaluated at specific values of a variable (here: cyl)
marg_ie1 <- margins(mod_ie, at = list(cyl = c(4, 6, 8)))
tidy(marg_ie1)

# Marginal effects of one interaction variable (here: wt), modulated at
# specific values of the two other interaction variables (here: cyl and drat)
marg_ie2 <- margins(mod_ie,
    variables = "wt", ## Main var
    at = list(cyl = c(4, 6, 8), drat = c(3, 3.5, 4))) ## Modulating vars
tidy(marg_ie2)

---

### tidy.Mclust

**Tidy a(n) Mclust object**

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'Mclust'
tidy(x, ...)
```
Arguments

- **x**
  
  An `Mclust` object return from `mclust::Mclust()`.

- **...**
  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- **proportion**
  
  The mixing proportion of each component

- **size**
  
  Number of points assigned to cluster.

- **mean**
  
  The mean for each component. In case of 2+ dimensional models, a column with the mean is added for each dimension. NA for noise component

- **variance**
  
  In case of one-dimensional and spherical models, the variance for each component, omitted otherwise. NA for noise component

- **component**
  
  Cluster id as a factor.

See Also

- `tidy()`, `mclust::Mclust()`

Other mclust tidiers: `augment.Mclust()`

Examples

```r
if (requireNamespace("mclust", quietly = TRUE)) {

library(dplyr)
library(mclust)
set.seed(27)

centers <- tibble::tibble(
  cluster = factor(1:3),
  num_points = c(100, 150, 50), # number points in each cluster
  x1 = c(5, 0, -3), # x1 coordinate of cluster center
  x2 = c(-1, 1, -2) # x2 coordinate of cluster center
)

points <- centers %>%
  mutate(
    x1 = purrr::map2(num_points, x1, rnorm),
    x2 = purrr::map2(num_points, x2, rnorm)
  ) %>%
  dplyr::select(-num_points, -cluster)
}
```
tidy.mediate

Tidy a(n) mediate object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'mediate'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

x A mediate object produced by a call to mediation::mediate().

conf.int Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.

conf.level The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

The tibble has four rows. The first two indicate the mediated effect in the control and treatment groups, respectively. And the last two the direct effect in each group.
Value

A tibble::tibble() with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.

See Also

tidy(), mediation::mediate()

Examples

```r
if (requireNamespace("mediation", quietly = TRUE)) {

  library(mediation)
  data(jobs)

  b <- lm(job_seek ~ treat + econ_hard + sex + age, data = jobs)
  c <- lm(depress2 ~ treat + job_seek + econ_hard + sex + age, data = jobs)
  mod <- mediate(b, c, sims = 50, treat = "treat", mediator = "job_seek")

  tidy(mod)
  tidy(mod, conf.int = TRUE)
  tidy(mod, conf.int = TRUE, conf.level = .99)
}
```

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

The particular functions below provide generic tidy methods for objects returned by the mfx package, preserving the calculated marginal effects instead of the naive model coefficients. The returned tidy tibble will also include an additional "atmean" column indicating how the marginal effects were originally calculated (see Details below).
Usage

## S3 method for class 'mfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'logitmfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'negbinmfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'poissonmfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

## S3 method for class 'probitmfx'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

x
A logitmfx, negbinmfx, poissonmfx, or probitmfx object. (Note that beta.mfx objects receive their own set of tidiers.)

conf.int
Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.

conf.level
The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

...
Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

The mfx package provides methods for calculating marginal effects for various generalized linear models (GLMs). Unlike standard linear models, estimated model coefficients in a GLM cannot be directly interpreted as marginal effects (i.e., the change in the response variable predicted after a one unit change in one of the regressors). This is because the estimated coefficients are multiplicative, dependent on both the link function that was used for the estimation and any other variables that were included in the model. When calculating marginal effects, users must typically choose whether they want to use i) the average observation in the data, or ii) the average of the sample marginal effects. See vignette("mfxarticle") from the mfx package for more details.

Value

A tibble::tibble() with columns:
tidy.mfx

conf.high  Upper bound on the confidence interval for the estimate.
conf.low   Lower bound on the confidence interval for the estimate.
estimate  The estimated value of the regression term.
p.value    The two-sided p-value associated with the observed statistic.
statistic  The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error  The standard error of the regression term.
term       The name of the regression term.
atmean     TRUE if the marginal effects were originally calculated as the partial effects for the average observation. If FALSE, then these were instead calculated as average partial effects.

See Also
tidy(), mfx::logitmfx(), mfx::negbinmfx(), mfx::poissonmfx(), mfx::probitmfx()

Other mfx tidiers:  augment.betamfx(), augment.mfx(), glance.betamfx(), glance.mfx(), tidy.betamfx()

Examples

if (requireNamespace("mfx", quietly = TRUE)) {

    ## Not run:
    library(mfx)

    ## Get the marginal effects from a logit regression
    mod_logmfx <- logitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)
tidy(mod_logmfx, conf.int = TRUE)

    ## Compare with the naive model coefficients of the same logit call (not run)
    # tidy(glm(am ~ cyl + hp + wt, family = binomial, data = mtcars), conf.int = TRUE)

    augment(mod_logmfx)
    glance(mod_logmfx)

    ## Another example, this time using probit regression
    mod_probmfx <- probitmfx(am ~ cyl + hp + wt, atmean = TRUE, data = mtcars)
tidy(mod_probmfx, conf.int = TRUE)
augment(mod_probmfx)
glance(mod_probmfx)

    ## End(Not run)
}


Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'mjoint'


tidy(
  x,
  component = "survival",
  conf.int = FALSE,
  conf.level = 0.95,
  boot.se = NULL,
  ...
)
```

Arguments

- **x**: An mjoint object returned from `joineRML::mjoint()`.
- **component**: Character specifying whether to tidy the survival or the longitudinal component of the model. Must be either "survival" or "longitudinal". Defaults to "survival".
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- **conf.level**: The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **boot.se**: Optionally a bootSE object from `joineRML::bootSE()`. If specified, calculates confidence intervals via the bootstrap. Defaults to NULL, in which case standard errors are calculated from the empirical information matrix.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an `augment()` method that does not accept a newdata argument, it will use the default value for the data argument.
tidy.mjoint

Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.

See Also

tidy(), joineRML::mjoint(), joineRML::bootSE()

Other mjoint tidiers: glance.mjoint()

Examples

```r
if (requireNamespace("joineRML", quietly = TRUE)) {

  ## Not run:
  # Fit a joint model with bivariate longitudinal outcomes
  library(joineRML)
  data(heart.valve)
  hvd <- heart.valve[!is.na(heart.valve$log.grad) &
                     !is.na(heart.valve$log.lvmi) &
                     heart.valve$num <= 50, ]
  fit <- mjoint(
    formLongFixed = list(
      "grad" = log.grad ~ time + sex + hs,
      "lvmi" = log.lvmi ~ time + sex
    ),
    formLongRandom = list(
      "grad" = ~ 1 | num,
      "lvmi" = ~ time | num
    ),
    formSurv = Surv(fuyrs, status) ~ age,
    data = hvd,
    inits = list("gamma" = c(0.11, 1.51, 0.80)),
    timeVar = "time"
  )

  # Extract the survival fixed effects
  tidy(fit)

  # Extract the longitudinal fixed effects
  tidy(fit, component = "longitudinal")
}
```
# Extract the survival fixed effects with confidence intervals
tidy(fit, ci = TRUE)

# Extract the survival fixed effects with confidence intervals based
# on bootstrapped standard errors
bSE <- bootSE(fit, nboot = 5, safe.boot = TRUE)
tidy(fit, boot_se = bSE, ci = TRUE)

# Augment original data with fitted longitudinal values and residuals
hvd2 <- augment(fit)

# Extract model statistics
glance(fit)

## End(Not run)

### tidy.mle2

**Tidy a(n) mle2 object**

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'mle2'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

**Arguments**

- `x`: An mle2 object created by a call to `bbmle::mle2()`.
- `conf.int`: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- `conf.level`: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with columns:

- `conf.high`  Upper bound on the confidence interval for the estimate.
- `conf.low`    Lower bound on the confidence interval for the estimate.
- `estimate`   The estimated value of the regression term.
- `p.value`    The two-sided p-value associated with the observed statistic.
- `statistic`  The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`  The standard error of the regression term.
- `term`       The name of the regression term.

See Also

tidy(), bbmle::mle2(), tidy_optim()

Examples

```r
if (requireNamespace("bbmle", quietly = TRUE)) {

library(bbmle)

x <- 0:10
y <- c(26, 17, 13, 12, 20, 5, 9, 8, 5, 4, 8)
d <- data.frame(x, y)

fit <- mle2(y ~ dpois(lambda = ymean),
            start = list(ymean = mean(y)), data = d)

tidy(fit)
}
```

---

**tidy.mlm**  
_Tidy a(n) mlm object_

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
Usage

### S3 method for class 'mlm'

tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

- **x**: An `mlm` object created by `stats::lm()` with a matrix as the response.
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- **conf.level**: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Details

In contrast to `lm` object (simple linear model), tidy output for `mlm` (multiple linear model) objects contain an additional column `response`.

If you have missing values in your model data, you may need to refit the model with `na.action = na.exclude`.

Value

A `tibble::tibble()` with columns:

- **conf.high**: Upper bound on the confidence interval for the estimate.
- **conf.low**: Lower bound on the confidence interval for the estimate.
- **estimate**: The estimated value of the regression term.
- **p.value**: The two-sided p-value associated with the observed statistic.
- **statistic**: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- **std.error**: The standard error of the regression term.
- **term**: The name of the regression term.

See Also

tidy()

Other lm tidiers: `augment.glm()`, `augment.lm()`, `glance.glm()`, `glance.lm()`, `glance.summary.lm()`, `glance.svyglm()`, `tidy.glm()`, `tidy.lm.beta()`, `tidy.lm()`, `tidy.summary.lm()`
**Examples**

```r
mod <- lm(cbind(mpg, disp) ~ wt, mtcars)
tidy(mod, conf.int = TRUE)
```

**Description**

These methods tidy the coefficients of mnl and nl models generated by the functions of the mlogit package.

**Usage**

```r
## S3 method for class 'mlogit'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

**Arguments**

- `x` an object returned from `mlogit::mlogit()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

**Value**

A `tibble::tibble()` with columns:

- `conf.high` Upper bound on the confidence interval for the estimate.
- `conf.low` Lower bound on the confidence interval for the estimate.
- `estimate` The estimated value of the regression term.
- `p.value` The two-sided p-value associated with the observed statistic.
- `statistic` The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error` The standard error of the regression term.
- `term` The name of the regression term.
See Also

tidy(), mlogit::mlogit()

Other mlogit tidiers: augment.mlogit(), glance.mlogit()

Examples

if (requireNamespace("mlogit", quietly = TRUE)) {

## Not run:
library(mlogit)
data("Fishing", package = "mlogit")
Fish <- dfidx(Fishing, varying = 2:9, shape = "wide", choice = "mode")
m <- mlogit(mode ~ price + catch | income, data = Fish)
tidy(m)
augment(m)
glance(m)

## End(Not run)
}

---

**tidy.muhaz**  
*Tidy a(n) muhaz object*

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'muhaz'
tidy(x, ...)
```

**Arguments**

- `x`  
  A muhaz object returned by `muhaz::muhaz()`.

- `...`  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
tidy.multinom

Value
A `tibble::tibble()` with columns:

- **time**: Point in time.
- **estimate**: Estimated hazard rate.

See Also
- `tidy()`, muhaz::muhaz()
- Other muhaz tidiers: `glance.muhaz()`

Examples

```r
if (requireNamespace("muhaz", quietly = TRUE)) {
  if (requireNamespace("survival", quietly = TRUE)) {

    library(muhaz)
    library(survival)

    x <- muhaz(ovarian$futime, ovarian$fustat)
    tidy(x)
    glance(x)

  }
}
```

```
tidy.multinom  Tidying methods for multinomial logistic regression models

Description
These methods tidy the coefficients of multinomial logistic regression models generated by `multinom` of the `nnet` package.

Usage
```
## S3 method for class 'multinom'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments
- **x**: A `multinom` object returned from `nnet::multinom()`.
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`. 
conf.level The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

exponentiate Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:

- conf.high Upper bound on the confidence interval for the estimate.
- conf.low Lower bound on the confidence interval for the estimate.
- estimate The estimated value of the regression term.
- p.value The two-sided p-value associated with the observed statistic.
- statistic The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- std.error The standard error of the regression term.
- term The name of the regression term.
- y.value The response level.

See Also
tidy(), nnet::multinom()

Other multinom tidiers: glance.multinom()

Examples

```r
if (requireNamespace("nnet", quietly = TRUE)) {

library(nnet)
library(MASS)

eexample(birthwt)
bwt.mu <- multinom(low ~ ..., bwt)
tidy(bwt.mu)
glance(bwt.mu)

#* This model is a truly terrible model
```
**tidy.negbin**

```r
#
# but it should show you what the output looks like in a multinomial logistic regression

fit.gear <- multinom(gear ~ mpg + factor(am), data = mtcars)
tidy(fit.gear)
glance(fit.gear)
```

---

**tidy.negbin**

Tidy a(n) negbin object

---

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'negbin'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

**Arguments**

- `x` A `glm.nb` object returned by `MASS::glm.nb()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `exponentiate` Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

**See Also**

- `MASS::glm.nb()`
- Other `glm.nb` tidiers: `glance.negbin()`
Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {

library(MASS)

r <- glm.nb(Days ~ Sex/(Age + Eth*Lrn), data = quine)

tidy(r)
glance(r)
}
```

tidy.nlrq  

*Tidy a(n) nlrq object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'nlrq'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- **x**  
  A nlrq object returned from `quantreg::nlrq()`.

- **conf.int**  
  Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.

- **conf.level**  
  The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

- **...**  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf_level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
tidy.nls

Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.

See Also

tidy(), quantreg::nlrq()

Other quantreg tidiers: augment.nlrq(), augment.rqs(), augment.rq(), glance.nlrq(), glance.rq(), tidy.rqs(), tidy.rq()

tidy.nls

Tidy a(n) nls object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'nls'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x`: An `nls` object returned from `stats::nls()`.
- `conf.int`: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level`: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Value**

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.

**See Also**

`tidy`, `stats::nls`, `stats::summary.nls()`

Other nls tidiers: `augment.nls()`, `glance.nls()`

**Examples**

```r
n <- nls(mpg ~ k * e^wt, data = mtcars, start = list(k = 1, e = 2))

tidy(n)
augment(n)
glance(n)

library(ggplot2)
ggplot(augment(n), aes(wt, mpg)) +
  geom_point() +
  geom_line(aes(y = .fitted))

newdata <- head(mtcars)
newdata$wt <- newdata$wt + 1
augment(n, newdata = newdata)
```
tidy.numeric  

**Tidy atomic vectors**

**Description**

Vector tidiers are deprecated and will be removed from an upcoming release of broom.

**Usage**

```r
## S3 method for class 'numeric'
tidy(x, ...)

## S3 method for class 'character'
tidy(x, ...)

## S3 method for class 'logical'
tidy(x, ...)
```

**Arguments**

- `x` An object of class "numeric", "integer", "character", or "logical". Most likely a named vector
- `...` Extra arguments (not used)

**Details**

Turn atomic vectors into data frames, where the names of the vector (if they exist) are a column and the values of the vector are a column.

**See Also**

Other deprecated: `bootstrap()`, `confint_tidy()`, `data.frame_tidiers`, `finish_glance()`, `fix_data_frame()`, `summary_tidiers`, `tidy.density()`, `tidy.dist()`, `tidy.ftable()`

**Examples**

```r
## Not run:
x <- 1:5
names(x) <- letters[1:5]
tidy(x)

## End(Not run)
```
Tidy a(n) orcutt object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'orcutt'
 tidy(x, ...)
```

Arguments

- `x` An orcutt object returned from `orcutt::cochrane.orcutt()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lev = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with columns:

- `estimate` The estimated value of the regression term.
- `p.value` The two-sided p-value associated with the observed statistic.
- `statistic` The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error` The standard error of the regression term.
- `term` The name of the regression term.

See Also

`orcutt::cochrane.orcutt()`

Other orcutt tidiers: `glance.orcutt()`
Examples

```r
if (requireNamespace("orcutt", quietly = TRUE)) {
  library(orcutt)
  reg <- lm(mpg ~ wt + qsec + disp, mtcars)
  tidy(reg)
  co <- cochrane.orcutt(reg)
  tidy(co)
  glance(co)
}
```

---

**tidy.pairwise.htest**  
*Tidy a(n) pairwise.htest object*

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'pairwise.htest'
tidy(x, ...)
```

**Arguments**

- `x`: A pairwise.htest object such as those returned from `stats::pairwise.t.test()` or `stats::pairwise.wilcox.test()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.lvel = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Details

Note that in one-sided tests, the alternative hypothesis of each test can be stated as "group1 is greater/less than group2".

Note also that the columns of group1 and group2 will always be a factor, even if the original input is (e.g.) numeric.

Value

A `tibble::tibble()` with columns:

- `group1`: First group being compared.
- `group2`: Second group being compared.
- `p.value`: The two-sided p-value associated with the observed statistic.

See Also

`stats::pairwise.t.test()`, `stats::pairwise.wilcox.test()`, `tidy()`

Other `htest` tidiers: `augment.htest()`, `tidy.htest()`, `tidy.power.htest()`

Examples

```r
attach(airquality)
Month <- factor(Month, labels = month.abb[5:9])
ptt <- pairwise.t.test(Ozone, Month)
tidy(ptt)

library(modeldata)
data(hpc_data)
attach(hpc_data)
ptt2 <- pairwise.t.test(compounds, class)
tidy(ptt2)

tidy(pairwise.t.test(compounds, class, alternative = "greater"))
tidy(pairwise.t.test(compounds, class, alternative = "less"))
tidy(pairwise.wilcoxon.test(compounds, class))
```

---

Tidy summaries information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
tidy.pam

Usage

```r
## S3 method for class 'pam'
tidy(x, col.names = paste0("x", 1:ncol(x$medoids)), ...)```

Arguments

- `x`: An pam object returned from `cluster::pam()`
- `col.names`: Column names in the input data frame. Defaults to the names of the variables in `x`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a newdata argument, it will use the default value for the data argument.

Details

For examples, see the pam vignette.

Value

A `tibble::tibble()` with columns:

- `size`: Size of each cluster.
- `max.diss`: Maximal dissimilarity between the observations in the cluster and that cluster's medoid.
- `avg.diss`: Average dissimilarity between the observations in the cluster and that cluster's medoid.
- `diameter`: Diameter of the cluster.
- `separation`: Separation of the cluster.
- `avg.width`: Average silhouette width of the cluster.
- `cluster`: A factor describing the cluster from 1:k.

See Also

`tidy()`, `cluster::pam()`

Other pam tidiers: `augment.pam()`, `glance.pam()`

Examples

```r
## Not run:
library(dplyr)
library(ggplot2)
library(cluster)
```
library(modeldata)
data(hpc_data)

x <- hpc_data[, 2:5]
p <- pam(x, k = 4)
tidy(p)
glance(p)
augment(p, x)

augment(p, x) %>%
  ggplot(aes(compounds, input_fields)) +
  geom_point(aes(color = .cluster)) +
  geom_text(aes(label = cluster), data = tidy(p), size = 10)

## End(Not run)

---

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'plm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

**Arguments**

- `x` A `plm` object returned by `plm::plm()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
tidy.plm

Value

A `tibble::tibble()` with columns:

- **conf.high**: Upper bound on the confidence interval for the estimate.
- **conf.low**: Lower bound on the confidence interval for the estimate.
- **estimate**: The estimated value of the regression term.
- **p.value**: The two-sided p-value associated with the observed statistic.
- **statistic**: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- **std.error**: The standard error of the regression term.
- **term**: The name of the regression term.

See Also

`tidy(), plm::plm(), tidy.lm()`

Other plm tidiers: `augment.plm(), glance.plm()`

Examples

```r
if (requireNamespace("plm", quietly = TRUE)) {

library(plm)
data("Produc", package = "plm")
zz <- plm(log(gsp) ~ log(pcap) + log(pc) + log(emp) + unemp,
        data = Produc, index = c("state", "year")
)
summary(zz)
tidy(zz)
tidy(zz, conf.int = TRUE)
tidy(zz, conf.int = TRUE, conf.level = 0.9)

augment(zz)
glance(zz)
}
```
Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'poLCA'
 tidy(x, ...)
```

Arguments

- `x`  
  A poLCA object returned from `polca::poLCA()`.
- `...`  
  Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `class`  
  The class under consideration.
- `outcome`  
  Outcome of manifest variable.
- `std.error`  
  The standard error of the regression term.
- `variable`  
  Manifest variable
- `estimate`  
  Estimated class-conditional response probability

See Also

tidy(), polca::polCA()
Other poLCA tidiers: augment.poLCA(), glance.poLCA()
Examples

```r
if (requireNamespace("poLCA", quietly = TRUE)) {

library(poLCA)
library(dplyr)

data(values)
f <- cbind(A, B, C, D) ~ 1
M1 <- poLCA(f, values, nclass = 2, verbose = FALSE)

M1
tidy(M1)
augment(M1)
glance(M1)

library(ggplot2)
ggplot(tidy(M1), aes(factor(class), estimate, fill = factor(outcome))) +
geom_bar(stat = "identity", width = 1) +
facet_wrap(~variable)
## Three-class model with a single covariate.

data(election)
f2a <- cbind(
  MORALG, CARESG, KNOWG, LEADG, DISHONG, INTELG,
  MORALB, CARESB, KNOWB, LEADB, DISHONB, INTELB
) ~ PARTY
nes2a <- poLCA(f2a, election, nclass = 3, nrep = 5, verbose = FALSE)

td <- tidy(nes2a)
td

# show
ggplot(td, aes(outcome, estimate, color = factor(class), group = class)) +
geom_line() +
facet_wrap(~variable, nrow = 2) +
theme(axis.text.x = element_text(angle = 90, hjust = 1))

au <- augment(nes2a)
au
count(au, .class)

# if the original data is provided, it leads to NAs in new columns
# for rows that weren't predicted
au2 <- augment(nes2a, data = election)
au2
dim(au2)
}
```
Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

## Usage

```r
## S3 method for class 'polr'
tidy(
x,
conf.int = FALSE,
conf.level = 0.95,
exponentiate = FALSE,
p.values = FALSE,
...
)
```

## Arguments

- **x**: A `polr` object returned from `MASS::polr()`.
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- **conf.level**: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **exponentiate**: Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
- **p.values**: Logical. Should p-values be returned, based on chi-squared tests from `MASS::dropterm()`. Defaults to `FALSE`.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Details

In broom 0.7.0 the coefficient_type column was renamed to coef.type, and the contents were changed as well. Now the contents are coefficient and scale, rather than coefficient and zeta.

Calculating p.values with the dropterm() function is the approach suggested by the MASS package author. This approach is computationally intensive so that p.values are only returned if requested explicitly. Additionally, it only works for models containing no variables with more than two categories. If this condition is not met, a message is shown and NA is returned instead of p-values.

Value

A tibble::tibble() with columns:

- conf.high: Upper bound on the confidence interval for the estimate.
- conf.low: Lower bound on the confidence interval for the estimate.
- estimate: The estimated value of the regression term.
- p.value: The two-sided p-value associated with the observed statistic.
- statistic: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- std.error: The standard error of the regression term.
- term: The name of the regression term.

See Also

tidy.MASS::polr()

Other ordinal tidiers: augment.clm(), augment.polr(), glance.clmm(), glance.clm(), glance.polr(), glance.svyolr(), tidy.clmm(), tidy.clm(), tidy.svyolr()

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {

library(MASS)

fit <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
tidy(fit, exponentiate = TRUE, conf.int = TRUE)

glance(fit)
augment(fit, type.predict = "class")

fit2 <- polr(factor(gear) ~ am + mpg + qsec, data = mtcars)
tidy(fit, p.values = TRUE)

}
```
tidy.power.htest  

_Tidy a(n) power.htest object_

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'power.htest'
tidy(x, ...)
```

**Arguments**

- `x`  
  A `power.htest` object such as those returned from `stats::power.t.test()`.  
- `...`  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

**Value**

A `tibble::tibble()` with columns:

- `delta`  
  True difference in means.  
- `n`  
  Number of observations by component.  
- `power`  
  Power achieved for given value of `n`.  
- `sd`  
  Standard deviation.  
- `sig.level`  
  Significance level (Type I error probability).

**See Also**

- `stats::power.t.test()`  
- Other htest tidiers: `augment.htest()`, `tidy.htest()`, `tidy.pairwise.htest()`
Examples

```r
ptt <- power.t.test(n = 2:30, delta = 1)
tidy(ptt)

library(ggplot2)

ggplot(tidy(ptt), aes(n, power)) +
        geom_line()
```

---

**tidy.prcomp**

*Tidy a(n) prcomp object*

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'prcomp'
tidy(x, matrix = "u", ...)
```

**Arguments**

- `x`: A `prcomp` object returned by `stats::prcomp()`.
- `matrix`: Character specifying which component of the PCA should be tidied.
  - "u", "samples", "scores", or "x": returns information about the map from the original space into principle components space.
  - "v", "rotation", "loadings" or "variables": returns information about the map from principle components space back into the original space.
  - "d", "eigenvalues" or "pcs": returns information about the eigenvalues.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.1vel = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

**Details**

See [https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca](https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca) for information on how to interpret the various tidied matrices. Note that SVD is only equivalent to PCA on centered data.
Value

A `tibble::tibble` with columns depending on the component of PCA being tidied.

If `matrix` is "u", "samples", "scores", or "x" each row in the tidied output corresponds to the original data in PCA space. The columns are:

- **row**: ID of the original observation (i.e. rowname from original data).
- **PC**: Integer indicating a principal component.
- **value**: The score of the observation for that particular principal component. That is, the location of the observation in PCA space.

If `matrix` is "v", "rotation", "loadings" or "variables", each row in the tidied output corresponds to information about the principle components in the original space. The columns are:

- **row**: The variable labels (colnames) of the data set on which PCA was performed.
- **PC**: An integer vector indicating the principal component.
- **value**: The value of the eigenvector (axis score) on the indicated principal component.

If `matrix` is "d", "eigenvalues" or "pcs", the columns are:

- **PC**: An integer vector indicating the principal component.
- **std.dev**: Standard deviation explained by this PC.
- **percent**: Fraction of variation explained by this component (a numeric value between 0 and 1).
- **cumulative**: Cumulative fraction of variation explained by principle components up to this component (a numeric value between 0 and 1).

See Also

`stats::prcomp()`, `svd_tidiers`

Other svd tidiers: `augment.prcomp()`, `tidy_irlba()`, `tidy_svd()`

Examples

```r
pc <- prcomp(USArrests, scale = TRUE)

# information about rotation
tidy(pc)

# information about samples (states)
tidy(pc, "samples")

# information about PCs
tidy(pc, "pcs")

# state map
library(dplyr)
library(ggplot2)
```
tidy.pyears

```r
pc %>%
  tidy(matrix = "samples") %>%
  mutate(region = tolower(row)) %>%
  inner_join(map_data("state"), by = "region") %>%
  ggplot(aes(long, lat, group = group, fill = value)) +
  geom_polygon() +
  facet_wrap(~PC) +
  theme_void() +
  ggtitle("Principal components of arrest data")
au <- augment(pc, data = USArrests)
au

  ggplot(au, aes(.fittedPC1, .fittedPC2)) +
  geom_point() +
  geom_text(aes(label = .rownames), vjust = 1, hjust = 1)
```

Tidy a(n) pyears object

Description
Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage
```r
## S3 method for class 'pyears'
tidy(x, ...)
```

Arguments
- `x` A `pyears` object returned from `survival::pyears()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Details
- `expected` is only present in the output when if a `ratetable` term is present.
- If the `data.frame = TRUE` argument is supplied to `pyears`, this is simply the contents of `x$data`. 
Value

A `tibble::tibble()` with columns:

- `expected`: Expected number of events.
- `pyears`: Person-years of exposure.
- `n`: Number of subjects contributing time.
- `event`: Observed number of events.

See Also

- `tidy()`, `survival::pyears()`
- Other `pyears` tidiers: `glance.pyears()`
- Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {

  library(survival)

  temp.yr <- tcut(mgus$dxyr, 55:92, labels = as.character(55:91))
  temp.age <- tcut(mgus$age, 34:101, labels = as.character(34:100))
  ptime <- ifelse(is.na(mgus$pctime), mgus$futime, mgus$pctime)
  pstat <- ifelse(is.na(mgus$pctime), 0, 1)
  pfit <- pyears(Surv(ptime / 365.25, pstat) ~ temp.yr + temp.age + sex, mgus,
                  data.frame = TRUE)
  tidy(pfit)
  glance(pfit)

  # if data.frame argument is not given, different information is present in
  # output
  pfit2 <- pyears(Surv(ptime / 365.25, pstat) ~ temp.yr + temp.age + sex, mgus)
  tidy(pfit2)
  glance(pfit2)
}
```
Tidy a(n) rcorr object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'rcorr'
tidy(x, diagonal = FALSE, ...)
```

Arguments

- `x` An `rcorr` object returned from `Hmisc::rcorr()`.
- `diagonal` Logical indicating whether or not to include diagonal elements of the correlation matrix, or the correlation of a column with itself. For the elements, estimate is always 1 and `p.value` is always `NA`. Defaults to `FALSE`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

Suppose the original data has columns A and B. In the correlation matrix from `rcorr` there may be entries for both the `cor(A, B)` and `cor(B, A)`. Only one of these pairs will ever be present in the tidy output.

Value

A `tibble::tibble()` with columns:

- `column1` Name or index of the first column being described.
- `column2` Name or index of the second column being described.
- `estimate` The estimated value of the regression term.
- `p.value` The two-sided p-value associated with the observed statistic.
- `n` Number of observations used to compute the correlation.
See Also

`tidy()`, `Hmisc::rcorr()`

Examples

```r
if (requireNamespace("Hmisc", quietly = TRUE)) {

library(Hmisc)

mat <- replicate(52, rnorm(100))
# add some NAs
mat[sample(length(mat), 2000)] <- NA
# also column names
colnames(mat) <- c(LETTERS, letters)

rc <- rcorr(mat)

td <- tidy(rc)

td

library(ggplot2)
ggplot(td, aes(p.value)) +
  geom_histogram(binwidth = .1)

ggplot(td, aes(estimate, p.value)) +
  geom_point() +
  scale_y_log10()
}
```

---

**tidy.ref.grid**

* Tidy a(n) ref.grid object

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'ref.grid'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```
tidy.ref.grid

Arguments

x
A ref.grid object created by `emmeans::ref_grid()`.

conf.int
Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.

conf.level
The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

... Additional arguments passed to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`.

Cautionary note: misspecified arguments may be silently ignored!

Details

Returns a data frame with one observation for each estimated marginal mean, and one column for each combination of factors. When the input is a contrast, each row will contain one estimated contrast.

There are a large number of arguments that can be passed on to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`.

Value

A `tibble::tibble()` with columns:

- conf.high Upper bound on the confidence interval for the estimate.
- conf.low Lower bound on the confidence interval for the estimate.
- df Degrees of freedom used by this term in the model.
- p.value The two-sided p-value associated with the observed statistic.
- std.error The standard error of the regression term.
- estimate Expected marginal mean
- statistic T-ratio statistic

See Also

`tidy()`, `emmeans::ref_grid()`, `emmeans::emmeans()`, `emmeans::contrast()`

Other emmeans tidiers: `tidy.emmGrid()`, `tidy.lsmobj()`, `tidy.summary_emm()`

Examples

```r
if (requireNamespace("emmeans", quietly = TRUE)) {

library(emmeans)
# linear model for sales of oranges per day
oranges_lm1 <- lm(sales1 ~ price1 + price2 + day + store, data = oranges)

# reference grid; see vignette("basics", package = "emmeans")
oranges_rg1 <- ref_grid(oranges_lm1)

oranges_rg1 <- tidy(oranges_rg1)
}
```
tidy.regsubsets

Tidy a(n) regsubsets object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'regsubsets'
tidy(x, ...)

```
# marginal averages
marginal <- emmeans(oranges_rg1, "day")
tidy(marginal)

# contrasts
tidy(contrast(marginal))
tidy(contrast(marginal, method = "pairwise"))

# plot confidence intervals
library(ggplot2)
ggplot(tidy(marginal, conf.int = TRUE), aes(day, estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# by multiple prices
by_price <- emmeans(oranges_lm1, "day",
  by = "price2",
  at = list(
    price1 = 50, price2 = c(40, 60, 80),
    day = c("2", "3", "4")
  )
)
by_price
tidy(by_price)

ggplot(tidy(by_price, conf.int = TRUE), aes(price2, estimate, color = day)) +
  geom_line() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# joint_tests
tidy(joint_tests(oranges_lm1))
```

```
Arguments

x  A regsubsets object created by leaps::regsubsets().
...
\(\ldots\)  Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in \(\ldots\), where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.1vel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A \texttt{tibble::tibble()} with columns:

- \texttt{r.squared}  R squared statistic, or the percent of variation explained by the model.
- \texttt{adj.r.squared}  Adjusted R squared statistic
- \texttt{BIC}  Bayesian information criterion for the component.
- \texttt{mallows_cp}  Mallow’s Cp statistic.

See Also

tidy(), leaps::regsubsets()

Examples

```r
if (requireNamespace("leaps", quietly = TRUE)) {
  all_fits <- leaps::regsubsets(hp ~ ., mtcars)
  tidy(all_fits)
}
```

tidy.ridgelm  \textit{Tidy a(n) ridgelm object}

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class \'ridgelm\'
 tidy(x, \ldots)
```
**Arguments**

- `x`: A `ridgelm` object returned from `MASS::lm.ridge()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Value**

A `tibble::tibble()` with columns:

- `GCV`: Generalized cross validation error estimate.
- `lambda`: Value of penalty parameter lambda.
- `term`: The name of the regression term.
- `estimate`: Estimate of scaled coefficient using this lambda.
- `scale`: Scaling factor of estimated coefficient.

**See Also**

`tidy()`, `MASS::lm.ridge()`

Other `ridgelm` tidiers: `glance.ridgelm()`

**Examples**

```r
if (requireNamespace("MASS", quietly = TRUE)) {

  names(longley)[1] <- "y"
  fit1 <- MASS::lm.ridge(y ~ ., longley)
  tidy(fit1)

  fit2 <- MASS::lm.ridge(y ~ ., longley, lambda = seq(0.001, .05, .001))
  td2 <- tidy(fit2)
  g2 <- glance(fit2)

  # coefficient plot
  library(ggplot2)
  ggplot(td2, aes(lambda, estimate, color = term)) + geom_line()

  # GCV plot
  ggplot(td2, aes(lambda, GCV)) + geom_line()

  # add line for the GCV minimizing estimate
```
tidy.rlm

Tidy a(n) rlm object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'rlm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- **x**: An `rlm` object returned by `MASS::rlm()`.
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- **conf.level**: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

See Also

- `MASS::rlm()`
- Other rlm tidiers: `augment.rlm()`, `glance.rlm()`
**tidy.rma**  
*Tidy a(n) rma object*

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'rma'
tidy(
x,  
conf.int = FALSE,  
conf.level = 0.95,  
exponentiate = FALSE,  
include_studies = FALSE,  
measure = "GEN",  
...)
```

**Arguments**

- `x` An `rma` object such as those created by `metafor::rma()`, `metafor::rma.uni()`, `metafor::rma.glmm()`, `metafor::rma.mh()`, `metafor::rma.mv()`, or `metafor::rma.peto()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `exponentiate` Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
- `include_studies` Logical. Should individual studies be included in the output? Defaults to `FALSE`.
- `measure` Measure type. See `metafor::escalc()`
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the individual study.
- `type`: The estimate type (summary vs individual study).

Examples

```r
if (requireNamespace("metafor", quietly = TRUE)) {
  library(metafor)
  df <- escalc(
    measure = "RR",
    ai = tpos,
    bi = tneg,
    ci = cpos,
    di = cneg,
    data = dat.bcg
  )
  meta_analysis <- rma(yi, vi, data = df, method = "EB")
  tidy(meta_analysis)
}
```

---

tidy.roc

**Tidy a(n) roc object**

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
Usage

```r
## S3 method for class 'roc'
tidy(x, ...)
```

Arguments

- `x` An `roc` object returned from a call to `AUC::roc()`.  
- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `cutoff` The cutoff used for classification. Observations with predicted probabilities above this value were assigned class 1, and observations with predicted probabilities below this value were assigned class 0.
- `fpr` False positive rate.
- `tpr` The true positive rate at the given cutoff.

See Also

- `tidy()`, `AUC::roc()`

Examples

```r
if (requireNamespace("AUC", quietly = TRUE)) {

  library(AUC)
  data(churn)
  r <- roc(churn$predictions, churn$labels)

  td <- tidy(r)
  td

  library(ggplot2)
  ggplot(td, aes(fpr, tpr)) +
  geom_line()

  # compare the ROC curves for two prediction algorithms

  library(dplyr)
  library(tidyrr)
}
tidy.rq <- churn %>%
  pivot_longer(contains("predictions"),
    names_to = "algorithm",
    values_to = "value"
  ) %>%
  nest(data = -algorithm) %>%
  mutate(tidy_roc = purrr::map(data, ~ tidy(roc(.x$value, .x$labels)))) %>%
  unnest(tidy_roc)

ggplot(rocs, aes(fpr, tpr, color = algorithm)) +
  geom_line()

---

tidy.rq  
_Tidy a(n) rq object_

### Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

### Usage

```r
## S3 method for class 'rq'
tidy(x, se.type = NULL, conf.int = FALSE, conf.level = 0.95, ...)
```

### Arguments

- **x**: An `rq` object returned from `quantreg::rq()`.
- **se.type**: Character specifying the method to use to calculate standard errors. Passed to `quantreg::summary.rq()` `se` argument. Defaults to "rank" if the sample size is less than 1000, otherwise defaults to "nid".
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- **conf.level**: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **...**: Additional arguments passed to `quantreg::summary.rq()`.

### Details

If `se.type = "rank"` confidence intervals are calculated by `summary.rq` and `statistic` and `p.value` values are not returned. When only a single predictor is included in the model, no confidence intervals are calculated and the confidence limits are set to NA.
Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `estimate`: The estimated value of the regression term.
- `p.value`: The two-sided p-value associated with the observed statistic.
- `statistic`: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error`: The standard error of the regression term.
- `term`: The name of the regression term.

See Also

tidy(), quantreg::rq()

Other quantreg tidiers: augment.nlrq(), augment.rqs(), augment.rq(), glance.nlrq(), glance.rq(), tidy.nlrq(), tidy.rqs()

tidy.rqs is a tidy() method for `rqs` objects.

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'rqs'
tidy(x, se.type = "rank", conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x`: An `rqs` object returned from `quantreg::rq()`.
- `se.type`: Character specifying the method to use to calculate standard errors. Passed to `quantreg::summary.rq()` se argument. Defaults to "rank".
- `conf.int`: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- `conf.level`: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...`: Additional arguments passed to `quantreg::summary.rqs()`.
Details

If se.type = "rank" confidence intervals are calculated by summary.rq. When only a single predictor is included in the model, no confidence intervals are calculated and the confidence limits are set to NA.

Value

A `tibble::tibble()` with columns:

- `conf.high`  Upper bound on the confidence interval for the estimate.
- `conf.low`   Lower bound on the confidence interval for the estimate.
- `estimate`  The estimated value of the regression term.
- `p.value`   The two-sided p-value associated with the observed statistic.
- `statistic` The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error` The standard error of the regression term.
- `term`      The name of the regression term.
- `quantile`  Linear conditional quantile.

See Also

tidy(), quantreg::rq()

Other quantreg tidiers: augment.nlrq(), augment.rqs(), augment.rq(), glance.nlrq(), glance.rq(), tidy.nlrq(), tidy.rq()

Description

These methods tidy the coefficients of spatial autoregression models generated by functions in the spatialreg package.

Usage

```r
## S3 method for class 'sarlm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```
Arguments

- **x**: An object returned from `spatialreg::lagsarlm()` or `spatialreg::errorsarlm()`.
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- **conf.level**: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- **conf.high**: Upper bound on the confidence interval for the estimate.
- **conf.low**: Lower bound on the confidence interval for the estimate.
- **estimate**: The estimated value of the regression term.
- **p.value**: The two-sided p-value associated with the observed statistic.
- **statistic**: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- **std.error**: The standard error of the regression term.
- **term**: The name of the regression term.

See Also

- `tidy()`, `spatialreg::lagsarlm()`, `spatialreg::errorsarlm()`, `spatialreg::sacsarlm()`

Other spatialreg tidiers: `augment.sarlm()`, `glance.sarlm()`

Examples

```r
## Not run:
library(spatialreg)
data(oldcol, package = "spdep")
listw <- spdep::nb2listw(COL.nb, style = "W")

crime_sar <- lagsarlm(CRIME ~ INC + HOVAL, 
    data = COL.OLD, 
    listw = listw, 
    method = "eigen")
```
tidy.spec

Tidy a(n) spec object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'spec'
tidy(x, ...)
```

Arguments

- `x`: A spec object created by `stats::spectrum()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with columns:

- `freq` Vector of frequencies at which the spectral density is estimated.
- `spec` Vector (for univariate series) or matrix (for multivariate series) of estimates of the spectral density at frequencies corresponding to `freq`.

See Also

`tidy()`, `stats::spectrum()`

Other time series tidiers: `tidy.acf()`, `tidy.ts()`, `tidy.zoo()`

Examples

```r
spc <- spectrum(lh)
tidy(spc)

library(ggplot2)
ggplot(tidy(spc), aes(freq, spec)) + geom_line()
```

### tidy.speedglm

#### Tidy a(n) speedglm object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'speedglm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

- `x` A `speedglm` object returned from `speedglm::speedglm()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
tidy.speedglm

exponentiate Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.

Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.level = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value A tibble::tibble() with columns:

  conf.high  Upper bound on the confidence interval for the estimate.
  conf.low   Lower bound on the confidence interval for the estimate.
  estimate   The estimated value of the regression term.
  p.value    The two-sided p-value associated with the observed statistic.
  statistic  The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
  std.error  The standard error of the regression term.
  term       The name of the regression term.

See Also

speedglm::speedglm()

Other speedlm tidiers: augment.speedlm(), glance.speedglm(), glance.speedlm(), tidy.speedlm()

Examples

if (requireNamespace("speedglm", quietly = TRUE)) {

library(speedglm)

clotting <- data.frame(
  u = c(5, 10, 15, 20, 30, 40, 60, 80, 100),
  lot1 = c(118, 58, 42, 35, 27, 25, 21, 19, 18)
)

fit <- speedglm(lot1 ~ log(u), data = clotting, family = Gamma(log))

tidy(fit)

}
tidy.speedlm  

Tidy a(n) speedlm object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'speedlm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

x  
A speedlm object returned from speedglm::speedlm().

conf.int  
Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.

conf.level  
The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

...  
Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:

conf.high  
Upper bound on the confidence interval for the estimate.

conf.low  
Lower bound on the confidence interval for the estimate.

estimate  
The estimated value of the regression term.

p.value  
The two-sided p-value associated with the observed statistic.

statistic  
The value of a T-statistic to use in a hypothesis that the regression term is non-zero.

std.error  
The standard error of the regression term.

term  
The name of the regression term.
See Also

speedglm::speedlm(), tidy.lm()

Other speedlm tidiers: augment.speedlm(), glance.speedglm(), glance.speedlm(), tidy.speedglm()

Examples

if (requireNamespace("speedglm", quietly = TRUE)) {
  mod <- speedglm::speedlm(mpg ~ wt + qsec, data = mtcars, fitted = TRUE)
  tidy(mod)
  glance(mod)
  augment(mod)
}

tidy.summary.glht Tidy a(n) summary.glht object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'summary.glht'
tidy(x, ...)

Arguments

x A summary.glht object created by calling multcomp::summary.glht() on a glht object created with multcomp::glht().

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with columns:

- **contrast**: Levels being compared.
- **estimate**: The estimated value of the regression term.
- **null.value**: Value to which the estimate is compared.
- **p.value**: The two-sided p-value associated with the observed statistic.
- **statistic**: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- **std.error**: The standard error of the regression term.

See Also

- `tidy()`, `multcomp::summary.glht()`, `multcomp::glht()`
- Other multcomp tidiers: `tidy.cld()`, `tidy.confint.glht()`, `tidy.glht()`

Examples

```r
if (requireNamespace("multcomp", quietly = TRUE)) {

library(multcomp)
library(ggplot2)

amod <- aov(breaks ~ wool + tension, data = warpbreaks)
wht <- glht(amod, linfct = mcp(tension = "Tukey"))

# tidy

 tidy(wht)
 ggplot(wht, aes(lhs, estimate)) +
   geom_point()

CI <- confint(wht)
tidy(CI)
 ggplot(CI, aes(lhs, estimate, ymin = lwr, ymax = upr)) +
   geom_pointrange()

tidy(summary(wht))
 ggplot(mapping = aes(lhs, estimate)) +
   geom_linerange(aes(ymin = lwr, ymax = upr), data = CI) +
   geom_point(aes(size = p), data = summary(wht)) +
   scale_size(trans = "reverse")

cld <- cld(wht)
tidy(cld)
}
```
tidy.summary.lm

Tidy a(n) summary.lm object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'summary.lm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

- `x` A `summary.lm` object created by `stats::summary.lm()`.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Details

The `tidy.summary.lm()` method is a potentially useful alternative to `tidy.lm()`. For instance, if users have already converted large `lm` objects into their leaner `summary.lm` equivalents to conserve memory.

Value

A `tibble::tibble()` with columns:

- `conf.high` Upper bound on the confidence interval for the estimate.
- `conf.low` Lower bound on the confidence interval for the estimate.
- `estimate` The estimated value of the regression term.
- `p.value` The two-sided p-value associated with the observed statistic.
statistic The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
std.error The standard error of the regression term.
term The name of the regression term.

See Also

tidy(), stats::summary.lm()

Other lm tidiers: augment.glm(), augment.lm(), glance.glm(), glance.lm(), glance.summary.lm(), glance.svyglm(), tidy.glm(), tidy.lm.beta(), tidy.lm(), tidy.mlm()

Examples

mod <- lm(mpg ~ wt + qsec, data = mtcars)
modsumm <- summary(mod)
tidy(mod, conf.int = TRUE)
tidy(modsumm, conf.int = TRUE) # same

glance(mod)
glance(modsumm) # mostly the same, except for a few missing columns
Details

Returns a data frame with one observation for each estimated marginal mean, and one column for each combination of factors. When the input is a contrast, each row will contain one estimated contrast.

There are a large number of arguments that can be passed on to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`.

Value

A `tibble::tibble()` with columns:

- `conf.high`  Upper bound on the confidence interval for the estimate.
- `conf.low`    Lower bound on the confidence interval for the estimate.
- `contrast`   Levels being compared.
- `den.df`     Degrees of freedom of the denominator.
- `df`         Degrees of freedom used by this term in the model.
- `null.value` Value to which the estimate is compared.
- `num.df`     Degrees of freedom.
- `p.value`    The two-sided p-value associated with the observed statistic.
- `std.error`  The standard error of the regression term.
- `level1`     One level of the factor being contrasted
- `level2`     The other level of the factor being contrasted
- `term`       Model term in joint tests
- `estimate`   Expected marginal mean
- `statistic`  T-ratio statistic or F-ratio statistic

See Also

`tidy()`, `emmeans::ref_grid()`, `emmeans::emmeans()`, `emmeans::contrast()`

Other emmeans tidiers: `tidy.emmGrid()`, `tidy.lsmobj()`, `tidy.ref.grid()`

Examples

```r
if (requireNamespace("emmeans", quietly = TRUE)) {

library(emmeans)
# linear model for sales of oranges per day
oranges_lm1 <- lm(sales1 ~ price1 + price2 + day + store, data = oranges)

# reference grid; see vignette("basics", package = "emmeans")
oranges_rgl <- ref_grid(oranges_lm1)
td <- tidy(oranges_rgl)
td
```

# marginal averages
marginal <- emmeans(oranges_rg1, "day")
tidy(marginal)

# contrasts
tidy(contrast(marginal))
tidy(contrast(marginal, method = "pairwise"))

# plot confidence intervals
library(ggplot2)
ggplot(tidy(marginal, conf.int = TRUE), aes(day, estimate)) +
  geom_point() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# by multiple prices
by_price <- emmeans(oranges_lm1, "day",
  by = "price2",
  at = list(
    price1 = 50, price2 = c(40, 60, 80),
    day = c("2", "3", "4")
  )
)
by_price
tidy(by_price)

ggplot(tidy(by_price, conf.int = TRUE), aes(price2, estimate, color = day)) +
  geom_line() +
  geom_errorbar(aes(ymin = conf.low, ymax = conf.high))

# joint_tests
tidy(joint_tests(oranges_lm1))

---

### tidy.survdiff

**Tidy a(n) survdiff object**

**Description**

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

**Usage**

```r
## S3 method for class 'survdiff'
tidy(x, ...)
```
Arguments

x  An survdiff object returned from survival::survdiff().

... Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with columns:

exp  Weighted expected number of events in each group.

N  Number of subjects in each group.

obs  weighted observed number of events in each group.

See Also

 tidy().survival::survdiff()

Other survdiff tidiers: glance.survdiff()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survexp(), tidy.survfit(), tidy.survreg()

Examples

if (requireNamespace("survival", quietly = TRUE)) {

library(survival)

s <- survdiff(
  Surv(time, status) ~ pat.karno + strata(inst),
  data = lung
)

tidy(s)

}
tidy.survexp

Describes

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'survexp'
tidy(x, ...)
```

Arguments

- `x` An survexp object returned from `survival::survexp()`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `n.risk` Number of individuals at risk at time zero.
- `time` Point in time.
- `estimate` Estimate survival

See Also

- `tidy()`, `survival::survexp()`
- Other survexp tidiers: `glance.survexp()`
- Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survfit()`, `tidy.survreg()`
Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {

library(survival)
sexpfit <- survexp(
  futime ~ 1,
  rmap = list(
    sex = "male",
    year = accept.dt,
    age = (accept.dt - birth.dt)
  ),
  method = "conditional",
  data = jasa
)

 tidy(sexpfit)
 glance(sexpfit)
}
```

**tidy.survfit**  
*Tidy a(n) survfit object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'survfit'
tidy(x, ...)
```

Arguments

- `x`  
  An `survfit` object returned from `survival::survfit()`.

- `...`  
  Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble()` with columns:

- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `n.censor`: Number of censored events.
- `n.event`: Number of events at time t.
- `n.risk`: Number of individuals at risk at time zero.
- `std.error`: The standard error of the regression term.
- `time`: Point in time.
- `estimate`: Estimate of survival or cumulative incidence rate when multistate
- `state`: State if multistate `survfit` object input
- `strata`: Strata if stratified `survfit` object input

See Also

`tidy()`, `survival::survfit()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdif()`, `glance.survep()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdif()`, `tidy.survep()`, `tidy.survreg()`

Examples

```r
if (requireNamespace("survival", quietly = TRUE)) {

library(survival)
cfit <- coxph(Surv(time, status) ~ age + sex, lung)
sfit <- survfit(cfit)

tidy(sfit)
glance(sfit)

library(ggplot2)
ggplot(tidy(sfit), aes(time, estimate)) +
  geom_line() +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)

# multi-state
fitCI <- survfit(Surv(stop, status * as.numeric(event), type = "mstate") ~ 1,
data = mgus1, subset = (start == 0))

fitCI

library(ggplot2)
ggplot(tidy(fitCI), aes(time, estimate, group = state)) +
  geom_line(aes(color = state)) +
  geom_ribbon(aes(ymin = conf.low, ymax = conf.high), color = "blue", alpha = .25)
}
```
tidy.survreg

```r
tidy.survreg = Tidy a(n) survreg object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'survreg'
tidy(x, conf.level = 0.95, conf.int = FALSE, ...)
```

Arguments

- `x` An `survreg` object returned from `survival::survreg()`.
- `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- `...` Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `conf.high` Upper bound on the confidence interval for the estimate.
- `conf.low` Lower bound on the confidence interval for the estimate.
- `estimate` The estimated value of the regression term.
- `p.value` The two-sided p-value associated with the observed statistic.
- `statistic` The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- `std.error` The standard error of the regression term.
- `term` The name of the regression term.
See Also

```
tidy(), survival::survreg()
```

Other survreg tidiers: `augment.survreg()`, `glance.survreg()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survep()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survep()`, `tidy.survfit()`

Examples

```
if (requireNamespace("survival", quietly = TRUE)) {

library(survival)

sr <- survreg(
  Surv(futime, fustat) ~ ecog.ps + rx,
  ovarian,
  dist = "exponential"
)

 tidy(sr)
 augment(sr, ovarian)
 glance(sr)

# coefficient plot
 td <- tidy(sr, conf.int = TRUE)
 library(ggplot2)
 ggplot(td, aes(estimate, term)) +
   geom_point() +
   geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) +
   geom_vline(xintercept = 0)
}
```

```
tidy.svyglm

Tidy a(n) svyglm object
```

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.
Usage

## S3 method for class 'svyglm'

```r
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
```

Arguments

- **x**
  - A `svyglm` object returned from `survey::svyglm()`.
- **conf.int**
  - Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- **conf.level**
  - The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **exponentiate**
  - Logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
- **...**
  - Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

See Also

- `survey::svyglm()`, `stats::glm()`

---

**tidy.svyolr**

*Tidy a(n) svyolr object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

## S3 method for class 'svyolr'

tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  exponentiate = FALSE,
  p.values = FALSE,
  ...
)
```
Arguments

x A svyolr object returned from \texttt{survey::svyolr()}.  

\textbf{conf.int} Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to \texttt{FALSE}.  

\textbf{conf.level} The confidence level to use for the confidence interval if \texttt{conf.int = TRUE}. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.  

\textbf{exponentiate} Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to \texttt{FALSE}.  

\textbf{p.values} Logical. Should p-values be returned, based on chi-squared tests from \texttt{MASS::dropterm()}. Defaults to \texttt{FALSE}.  

\textbf{...} Additional arguments. Not used. Needed to match generic signature only. \textbf{Cautionary note:} Misspelled arguments will be absorbed in \ldots, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass \texttt{conf.level = 0.9}, all computation will proceed using \texttt{conf.level = 0.95}. Additionally, if you pass \texttt{newdata = my_tibble} to an \texttt{augment()} method that does not accept a \texttt{newdata} argument, it will use the default value for the \texttt{data} argument.

Details

In broom 0.7.0 the \texttt{coefficient_type} column was renamed to \texttt{coef.type}, and the contents were changed as well. Now the contents are \texttt{coefficient} and \texttt{scale}, rather than \texttt{coefficient} and \texttt{zeta}.  

Calculating \texttt{p.values} with the \texttt{dropterm()} function is the approach suggested by the MASS package author. This approach is computationally intensive so that \texttt{p.values} are only returned if requested explicitly. Additionally, it only works for models containing no variables with more than two categories. If this condition is not met, a message is shown and NA is returned instead of \texttt{p.values}.

Value

A \texttt{tibble::tibble()} with columns:

\begin{itemize}
  \item \textbf{conf.high} Upper bound on the confidence interval for the estimate.
  \item \textbf{conf.low} Lower bound on the confidence interval for the estimate.
  \item \textbf{estimate} The estimated value of the regression term.
  \item \textbf{p.value} The two-sided p-value associated with the observed statistic.
  \item \textbf{statistic} The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
  \item \textbf{std.error} The standard error of the regression term.
  \item \textbf{term} The name of the regression term.
\end{itemize}
tidy.systemfit

See Also
tidy, survey::svyolr()
Other ordinal tidiers: augment.clm(), augment.polr(), glance.clmm(), glance.clm(), glance.polr(), glance.svyolr(), tidy.clmm(), tidy.clm(), tidy.polr()

Examples

```r
if (requireNamespace("MASS", quietly = TRUE)) {
  library(MASS)
  fit <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
  tidy(fit, exponentiate = TRUE, conf.int = TRUE)
  glance(fit)
}
```

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'systemfit'
tidy(x, conf.int = TRUE, conf.level = 0.95, ...)
```

Arguments

- `x`: A systemfit object produced by a call to `systemfit::systemfit()`.
- `conf.int`: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
- `conf.level`: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed
using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to a `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

**Details**

This tidy method works with any model objects of class `systemfit`. Default returns a tibble of six columns.

**Value**

A tibble::tibble() with columns:

- `conf.high` Upper bound on the confidence interval for the estimate.
- `conf.low` Lower bound on the confidence interval for the estimate.
- `estimate` The estimated value of the regression term.
- `p.value` The two-sided p-value associated with the observed statistic.
- `std.error` The standard error of the regression term.
- `term` The name of the regression term.

**See Also**

`tidy()`, `systemfit::systemfit()`

**Examples**

```r
if (requireNamespace("systemfit", quietly = TRUE)) {

  set.seed(27)

  library(systemfit)

  df <- data.frame(
    X = rnorm(100),
    Y = rnorm(100),
    Z = rnorm(100),
    W = rnorm(100)
  )

  fit <- systemfit(formula = list(Y ~ Z, W ~ X), data = df, method = "SUR")

  tidy(fit)

  tidy(fit, conf.int = TRUE)
}
```
Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Deprecated. Please use `tibble::as_tibble()` instead.

Usage

```r
## S3 method for class 'table'
tidy(x, ...)
```

Arguments

- `x`: A `base::table` object.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lev = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

Directly calls `tibble::as_tibble()` on a `base::table` object, which does the same things as `base::as.data.frame.table()` but also gives the returned object `tibble::tibble` class.

Value

A `tibble::tibble` in long-form containing frequency information for the table in a `Freq` column. The result is much like what you get from `tidyr::pivot_longer()`.

See Also

`tibble::as_tibble.table()`
Tidy a(n) ts object

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'ts'
tidy(x, ...)
```

Arguments

- **x**: A univariate or multivariate ts times series object.
- **...**: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note**: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.1vel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Details

- `series` column is only present for multivariate ts objects.

Value

A `tibble::tibble()` with columns:

- **index**: Index (i.e. date or time) for a 'ts' or 'zoo' object.
- **series**: Name of the series (present only for multivariate time series).
- **value**: The value/estimate of the component. Results from data reshaping.

See Also

- `tidy()`, `stats::ts()`

Other time series tidiers: `tidy.acf()`, `tidy.spec()`, `tidy.zoo()`
Examples

```r
set.seed(678)
tidy(ts(1:10, frequency = 4, start = c(1959, 2)))

z <- ts(matrix(rnorm(300), 100, 3), start = c(1961, 1), frequency = 12)
colnames(z) <- c("Aa", "Bb", "Cc")
tidy(z)
```

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```r
## S3 method for class 'TukeyHSD'
tidy(x, ...)
```

Arguments

- `x`: A TukeyHSD object return from `stats::TukeyHSD()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `adj.p.value`: P-value adjusted for multiple comparisons.
- `conf.high`: Upper bound on the confidence interval for the estimate.
- `conf.low`: Lower bound on the confidence interval for the estimate.
- `contrast`: Levels being compared.
- `estimate`: The estimated value of the regression term.
- `null.value`: Value to which the estimate is compared.
- `term`: The name of the regression term.
See Also

`tidy()`, `stats::TukeyHSD()`

Other anova tidiers: `glance.aov()`, `tidy.anova()`, `tidy.aovlist()`, `tidy.aov()`, `tidy.manova()`

Examples

```r
fm1 <- aov(breaks ~ wool + tension, data = warpbreaks)
thsd <- TukeyHSD(fm1, "tension", ordered = TRUE)
tidy(thsd)

# may include comparisons on multiple terms
fm2 <- aov(mpg ~ as.factor(gear) * as.factor(cyl), data = mtcars)
tidy(TukeyHSD(fm2))
```

## S3 method for class 'varest'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

- **x**: A `varest` object produced by a call to `vars::VAR()`.
- **conf.int**: Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
- **conf.level**: The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
- **...**: Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.level = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Details

The tibble has one row for each term in the regression. The component column indicates whether a particular term was used to model either the "mean" or "precision". Here the precision is the inverse of the variance, often referred to as phi. At least one term will have been used to model the precision phi.

The vars package does not include a confint method and does not report confidence intervals for varest objects. Setting the tidy argument conf.int = TRUE will return a warning.

Value

A tibble::tibble() with columns:

- conf.high: Upper bound on the confidence interval for the estimate.
- conf.low: Lower bound on the confidence interval for the estimate.
- estimate: The estimated value of the regression term.
- p.value: The two-sided p-value associated with the observed statistic.
- statistic: The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- std.error: The standard error of the regression term.
- term: The name of the regression term.
- component: Whether a particular term was used to model the mean or the precision in the regression. See details.

See Also

tidy(), vars::VAR()

Examples

```r
if (requireNamespace("vars", quietly = TRUE)) {

library(vars)
data("Canada", package = "vars")

mod <- VAR(Canada, p = 1, type = "both")

tidy(mod)
glance(mod)
}
```
Description
Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies across models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage
```r
## S3 method for class 'zoo'
tidy(x, ...)
```

Arguments
- `x`: A `zoo` object such as those created by `zoo::zoo()`.
- `...`: Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value
A `tibble::tibble()` with columns:
- `index`: Index (i.e. date or time) for a ‘ts’ or ‘zoo’ object.
- `series`: Name of the series (present only for multivariate time series).
- `value`: The value/estimate of the component. Results from data reshaping.

See Also
- `tidy()`, `zoo::zoo()`
- Other time series tidiers: `tidy.acf()`, `tidy.spec()`, `tidy.ts()`

Examples
```r
if (requireNamespace("zoo", quietly = TRUE)) {

library(zoo)
library(ggplot2)

```
tidy_irlba

set.seed(1071)

# data generated as shown in the zoo vignette
Z.index <- as.Date(sample(12450:12500, 10))
Z.data <- matrix(rnorm(30), ncol = 3)
colnames(Z.data) <- c("Aa", "Bb", "Cc")
Z <- zoo(Z.data, Z.index)
tidy(Z)

ggplot(tidy(Z), aes(index, value, color = series)) +
  geom_line()

ggplot(tidy(Z), aes(index, value)) +
  geom_line() +
  facet_wrap(~series, ncol = 1)

Zrolled <- rollmean(Z, 5)

}
using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

**Details**

A very thin wrapper around `tidy_svd()`.

**Value**

A `tibble::tibble` with columns depending on the component of PCA being tidied.

If `matrix` is "u", "samples", "scores", or "x" each row in the tidied output corresponds to the original data in PCA space. The columns are:

- **row**: ID of the original observation (i.e. rowname from original data).
- **PC**: Integer indicating a principal component.
- **value**: The score of the observation for that particular principal component. That is, the location of the observation in PCA space.

If `matrix` is "v", "rotation", "loadings" or "variables", each row in the tidied output corresponds to information about the principle components in the original space. The columns are:

- **row**: The variable labels (colnames) of the data set on which PCA was performed.
- **PC**: An integer vector indicating the principal component.
- **value**: The value of the eigenvector (axis score) on the indicated principal component.

If `matrix` is "d", "eigenvalues" or "pcs", the columns are:

- **PC**: An integer vector indicating the principal component.
- **std.dev**: Standard deviation explained by this PC.
- **percent**: Fraction of variation explained by this component (a numeric value between 0 and 1).
- **cumulative**: Cumulative fraction of variation explained by principle components up to this component (a numeric value between 0 and 1).

See Also

`tidy(), irlba::irlba()`

Other list tidiers: `glance_optim(), list_tidiers, tidy_optim(), tidy_svd(), tidy_xyz()`

Other svd tidiers: `augment.prcomp(), tidy.prcomp(), tidy_svd()`
Examples

```r
if (requireNamespace("modeldata", quietly = TRUE)) {

library(modeldata)
data(hpc_data)

mat <- scale(as.matrix(hpc_data[, 2:5]))
s <- svd(mat)

tidy_u <- tidy(s, matrix = "u")
tidy_u

tidy_d <- tidy(s, matrix = "d")
tidy_d

tidy_v <- tidy(s, matrix = "v")
tidy_v

library(ggplot2)
library(dplyr)

ggplot(tidy_d, aes(PC, percent)) +
  geom_point() +
  ylab("% of variance explained")

 tidy_u %>%
  mutate(class = hpc_data$class[row]) %>%
  ggplot(aes(class, value)) +
  geom_boxplot() +
  facet_wrap(~PC, scale = "free_y")
}
```

---

**tidy_optim**

*Tidy an optim object masquerading as list*

**Description**

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, `stats::optim()`, `svd()` and `akima::interp()` produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves implemented as functions of the form `tidy_<function>` or `glance_<function>` and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.
Usage

    tidy_optim(x, ...)

Arguments

x  A list returned from `stats::optim()`.
...
Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.

Value

A `tibble::tibble()` with columns:

- `parameter`: The parameter being modeled.
- `std.error`: The standard error of the regression term.
- `value`: The value/estimate of the component. Results from data reshaping.

`std.error` is only provided as a column if the Hessian is calculated.

Note

This function assumes that the provided objective function is a negative log-likelihood function. Results will be invalid if an incorrect function is supplied.

`tidy(o)` `glance(o)`

See Also

    `tidy()`, `stats::optim()`

Other list tidiers: `glance_optim()`, `list_tidiers`, `tidy_irlba()`, `tidy_svd()`, `tidy_xyz()`

Examples

```r
o <- optim(c(1, 1, 1), f)
```
tidy_svd

Tidy a(n) svd object masquerading as list

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example,
\texttt{stats::optim()}, \texttt{svd()} and \texttt{akima::interp()} produce consistent output, but because they do not
have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying
method to apply to the list. Those tidiers are themselves are implemented as functions of the form
tidy\_\texttt{<function>} or glance\_\texttt{<function>} and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.

Usage

\texttt{tidy_svd(x, \texttt{matrix} = "u", \ldots)}

Arguments

\texttt{x} \hspace{1cm} A list with components \texttt{u}, \texttt{d}, \texttt{v} returned by \texttt{base::svd}.

\texttt{matrix} \hspace{1cm} Character specifying which component of the PCA should be tidied.

- "u", "samples", "scores", or "x": returns information about the map from the original space into principle components space.
- "v", "rotation", "loadings" or "variables": returns information about the map from principle components space back into the original space.
- "d", "eigenvalues" or "pcs": returns information about the eigenvalues.

Additional arguments. Not used. Needed to match generic signature only. Cautionary note: Misspelled arguments will be absorbed in \ldots, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass \texttt{conf.lvel = 0.9}, all computation will proceed using \texttt{conf.lvel = 0.95}. Additionally, if you pass \texttt{newdata = my_tibble} to an \texttt{augment()} method that does not accept a \texttt{newdata} argument, it will use the default value for the data argument.

Details

See https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca for information on how to interpret the various tidied matrices. Note that SVD is only equivalent to PCA on centered data.

Value

A \texttt{tibble::tibble} with columns depending on the component of PCA being tidied.

If \texttt{matrix} is "u", "samples", "scores", or "x" each row in the tidied output corresponds to the original data in PCA space. The columns are:
tidy_svd

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>ID of the original observation (i.e. rowname from original data).</td>
</tr>
<tr>
<td>PC</td>
<td>Integer indicating a principal component.</td>
</tr>
<tr>
<td>value</td>
<td>The score of the observation for that particular principal component. That is, the location of the observation in PCA space.</td>
</tr>
</tbody>
</table>

If `matrix` is "v", "rotation", "loadings" or "variables", each row in the tidied output corresponds to information about the principle components in the original space. The columns are:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>The variable labels (colnames) of the data set on which PCA was performed</td>
</tr>
<tr>
<td>PC</td>
<td>An integer vector indicating the principal component</td>
</tr>
<tr>
<td>value</td>
<td>The value of the eigenvector (axis score) on the indicated principal component</td>
</tr>
</tbody>
</table>

If `matrix` is "d", "eigenvalues" or "pcs", the columns are:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>An integer vector indicating the principal component</td>
</tr>
<tr>
<td>std.dev</td>
<td>Standard deviation explained by this PC</td>
</tr>
<tr>
<td>percent</td>
<td>Fraction of variation explained by this component (a numeric value between 0 and 1).</td>
</tr>
<tr>
<td>cumulative</td>
<td>Cumulative fraction of variation explained by principle components up to this component (a numeric value between 0 and 1).</td>
</tr>
</tbody>
</table>

See Also

- `base::svd()`
- Other svd tidiers: `augment.prcomp()`, `tidy.prcomp()`, `tidy_irlba()`
- Other list tidiers: `glance_optim()`, `list_tidiers`, `tidy_irlba()`, `tidy_optim()`, `tidy_xyz()`

Examples

```r
if (requireNamespace("modeldata", quietly = TRUE)) {

  library(modeldata)
  data(hpc_data)

  mat <- scale(as.matrix(hpc_data[, 2:5]))
  s <- svd(mat)

  tidy_u <- tidy(s, matrix = "u")
  tidy_u

  tidy_d <- tidy(s, matrix = "d")
  tidy_d

  tidy_v <- tidy(s, matrix = "v")
  tidy_v

  library(ggplot2)
  library(dplyr)
```
tidy_xyz

Tidy a(n) xyz object masquerading as list

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, `stats::optim()`, `svd()` and `akima::interp()` produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves are implemented as functions of the form `tidy_<function>` or `glance_<function>` and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.

xyz lists (lists where \( x \) and \( y \) are vectors of coordinates and \( z \) is a matrix of values) are typically used by functions such as `graphics::persp()` or `graphics::image()` and returned by interpolation functions such as `akima::interp()`.

Usage

`tidy_xyz(x, ...)`

Arguments

- `x` A list with component \( x \), \( y \) and \( z \), where \( x \) and \( y \) are vectors and \( z \) is a matrix. The length of \( x \) must equal the number of rows in \( z \) and the length of \( y \) must equal the number of columns in \( z \).

- `...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the data argument.
Value

A `tibble::tibble` with vector columns x, y and z.

See Also

`tidy()`, `graphics::persp()`, `graphics::image()`, `akima::interp()`

Other list tidiers: `glance_optim()`, `list_tidiers`, `tidy_irlba()`, `tidy_optim()`, `tidy_svd()`

Examples

```r
A <- list(x = 1:5, y = 1:3, z = matrix(runif(5 * 3), nrow = 5))
image(A)
tidy(A)
```
Index

* Arima tidiers
  glance.Arima, 102
tidy.Arima, 222
* aareg tidiers
  glance.aareg, 99
tidy.aareg, 216
* anova tidiers
  glance.aov, 101
tidy.anova, 218
tidy.aov, 220
tidy.aovlist, 221
tidy.manova, 304
tidy.TukeyHSD, 379
* betareg tidiers
  tidy.betareg, 225
* biglm tidiers
  glance.biglm, 106
tidy.biglm, 227
* bingroup tidiers
  glance.binDesign, 108
tidy.binDesign, 228
tidy.binWidth, 230
* car tidiers
  durbinWatsonTest_tidiers, 97
  leveneTest_tidiers, 210
* cch tidiers
  glance.cch, 110
glance.survfit, 201
tidy.cch, 234
* cmprsk tidiers
  glance.crr, 119
tidy.crr, 247
* coeftest tidiers
  tidy.coeftest, 241
* coeftest_tidiers
  glance.coeftest, 115
* coxph tidiers
  augment.coxph, 17
glance.coxph, 117
tidy.coxph, 245
* crr tidiers
  glance.crr, 119
* decompose tidiers
  augment.decomposed.ts, 20
  augment.stl, 88
* deprecated
  bootstrap, 93
  confint_tidy, 94
  data.frame_tidiers, 95
  finish_glance, 98
  fix_data_frame, 99
  summary_tidiers, 215
  tidy.density, 251
  tidy.dist, 252
  tidy.ftable, 266
  tidy.numeric, 327
* drc tidiers
  augment.drc, 22
  glance.drc, 123
  tidy.drc, 253
* emmeans tidiers
  tidy.emmGrid, 254
  tidy.lsmobj, 302
  tidy.ref.grid, 344
  tidy.summary_emm, 364
* epiR tidiers
  tidy.epi.2by2, 256
* ergm tidiers
  glance.ergm, 124
  tidy.ergm, 258
* factanal tidiers
  augment.factanal, 25
  glance.factanal, 125
  tidy.factanal, 260
* felm tidiers
  augment.felm, 26
  tidy.felm, 261
* fitdistr tidiers
glance.fitdistr, 129
tidy.fitdistr, 263
* fixest tidiers
  augment.fixest, 28
tidy.fixest, 264
* gam tidiers
  glance.Gam, 132
tidy.Gam, 267
* garch tidiers
  glance.garch, 135
tidy.garch, 270
* geepack tidiers
  glance.geeglm, 136
* glm.nb tidiers
  glance.negbin, 169
tidy.negbin, 323
* glmnet tidiers
  glance.cv.glmnet, 120
glance.glmnet, 139
tidy.cv.glmnet, 249
tidy.glmnet, 276
* glm tidiers
  glance.gmm, 142
tidy.gmm, 280
* htest tidiers
  augment.htest, 38
tidy.htest, 283
tidy.pairwise.htest, 329
tidy.power.htest, 338
* ivreg tidiers
  augment.ivreg, 40
glance.ivreg, 144
tidy.ivreg, 284
* kmeans tidiers
  augment.kmeans, 42
glance.kmeans, 146
tidy.kmeans, 291
* lavaan tidiers
  glance.lavaan, 148
tidy.lavaan, 292
* list tidiers
  glance_optim, 209
  list_tidiers, 211
tidy_irlba, 383
tidy_optim, 385
tidy_svd, 387
tidy_xyz, 389
* lm tidiers
  augment.glm, 33
  augment.lm, 44
  glance.glm, 138
glance.lm, 150
  glance.summary.lm, 195
glance.svyglm, 204
tidy.glm, 275
tidy.lm, 294
tidy.lm.beta, 296
tidy.mlm, 317
tidy.summary.lm, 363
* lmodel2 tidiers
  glance.lmodel2, 152
tidy.lmodel2, 298
* margins tidiers
  tidy.margins, 306
* mclust tidiers
  augment.Mclust, 53
tidy.Mclust, 308
* mediate tidiers
  tidy.mediate, 310
* mfx tidiers
  augment.betamfx, 10
  augment.mfx, 55
  glance.betamfx, 103
  glance.mfx, 160
tidy.betamfx, 223
tidy.mfx, 311
* mgcv tidiers
  glance.gam, 134
tidy.gam, 268
* mjoint tidiers
  glance.mjoint, 163
tidy.mjoint, 314
* mlogit tidiers
  augment.mlogit, 61
  glance.mlogit, 165
tidy.mlogit, 319
* muhaz tidiers
  glance.muhaz, 166
tidy.muhaz, 320
* multcomp tidiers
  tidy.cld, 235
tidy.confint.glht, 242
tidy.glht, 273
tidy.summary.glht, 361
* multinom tidiers
  glance.multinom, 168
tidy.multinom, 321
* nls tidiers
  augment.nls, 64
glance.nls, 172
tidy.nls, 325
* orcutt tidiers
glance.orcutt, 173
tidy.orcutt, 328
* ordinal tidiers
  augment.clm, 15
  augment.polr, 72
  glance.clm, 111
glance.clmm, 113
  glance.polr, 180
  glance.svyolr, 206
tidy.clm, 237
tidy.clmm, 239
tidy.polr, 336
tidy.svyolr, 373
* pam tidiers
  augment.pam, 66
glance.pam, 175
tidy.pam, 330
* plm tidiers
  augment.plm, 68
glance.plm, 176
tidy.plm, 332
* poLCA tidiers
  augment.poLCA, 70
glance.poLCA, 178
tidy.poLCA, 334
* pyears tidiers
  glance.pyears, 182
tidy.pyears, 341
* quantreg tidiers
  augment.nlrq, 62
  augment.rq, 80
  augment.rqs, 82
glance.nlrq, 171
  glance.rq, 188
tidy.nlrq, 324
tidy.rq, 353
tidy.rqs, 354
* ridgelm tidiers
  glance.ridgelm, 183
tidy.ridgelm, 347
* rlm tidiers
  augment.rlm, 76
glance.rlm, 185
tidy.rlm, 349
* robust tidiers
  augment.lmRob, 47
glance.glmRob, 141
  glance.lmRob, 154
tidy.glmRob, 278
tidy.lmRob, 299
* robustbase tidiers
  augment(glmrob, 36
  augment.lmrob, 49
glance.lmrob, 155
tidy.glmrob, 279
tidy.lmrob, 300
* smoothing spline tidiers
  augment.smooth.spline, 85
glance.smooth.spline, 191
* spatialreg tidiers
  augment.sarlm, 83
glance.sarlm, 189
tidy.sarlm, 355
* speedlm tidiers
  augment.speedlm, 87
glance.speedglm, 192
glance.speedlm, 194
tidy.speedglm, 358
tidy.speedlm, 360
* survdiff tidiers
  glance.survdiff, 198
tidy.survdiff, 366
* survexp tidiers
  glance.survexp, 199
tidy.survexp, 368
* survey tidiers
  tidy.svyglm, 372
* survfit tidiers
  tidy.survfit, 369
* survival tidiers
  augment.coxph, 17
  augment.survreg, 90
glance.aareg, 99
glance.cch, 110
glance.coxph, 117
glance.pyears, 182
glance.survdiff, 198
glance.survexp, 199
glance.survfit, 201
glance.survreg, 203
tidy.aareg, 216  
tidy.cch, 234  
tidy.coxph, 245  
tidy.pyears, 341  
tidy.survdiff, 366  
tidy.survexp, 368  
tidy.survfit, 369  
tidy.survreg, 371

* survreg tidiers
  augment.survreg, 90  
glance.survreg, 203  
tidy.survreg, 371

* svd tidiers
  augment.prcomp, 74  
tidy.prcomp, 339  
tidy.irlba, 383  
tidy.svd, 387

* systemfit tidiers
  tidy.systemfit, 375

* time series tidiers
  tidy.acf, 217  
tidy.spec, 357  
tidy.ts, 378  
tidy.zoo, 382

* vars tidiers
  tidy.varest, 380

aareg_tidiers (tidy.aareg), 216
AER::ivreg(), 40, 41, 145, 146, 284, 285
aer_tidiers (tidy.ivreg), 284
akima:: interp(), 209, 211, 383, 385, 387, 389, 390
Arima_tidiers (tidy.Arima), 222
AUC::roc(), 352
auc_tidiers (tidy.roc), 351
augment, 81, 83
augment.betamfx, 10, 57, 104, 162, 224, 313
augment.betareg, 13
augment.betareg(), 12
augment.clm, 15, 74, 112, 114, 181, 207, 238, 240, 337, 375
augment.coxph, 17, 91, 100, 111, 118, 183, 199, 200, 202, 204, 217, 235, 246, 342, 367, 368, 370, 372
augment.data.frame
  (data.frame_tidiers), 95
augment.decomposed.ts, 20, 90
augment.drc, 22, 124, 254
augment.factorial, 25, 126, 260
augment.felm, 26, 262
augment.fixest, 28, 266
augment.gam, 31
augment.glm, 33, 46, 139, 151, 196, 205, 276, 295, 297, 318, 364
augment.glm(), 57
augment.glmRob, 35
augment.glmrob, 36, 50, 156, 280, 301
augment.htest, 38, 284, 330, 338
augment.ivreg, 40, 146, 285
augment.kmeans, 42, 147, 291
augment.lm, 35, 44, 139, 151, 196, 205, 276, 295, 297, 318, 364
augment.lmRob, 47, 142, 155, 278, 300
augment.lmrob, 38, 49, 156, 280, 301
augment.loess, 51
augment.logitmfx (augment.mfx), 55
augment.Mclust, 53, 309
augment.mfx, 12, 55, 104, 162, 224, 313
augment.mjoint, 58
augment.mlogit, 61, 166, 320
augment.negbinmfx (augment.mfx), 55
augment.nlrq, 62, 81, 83, 172, 189, 325, 354, 355
augment.nls, 64, 173, 326
augment.NULL (null_tidiers), 212
augment.pam, 66, 176, 331
augment.plm, 68, 177, 333
augment.poissonmfx (augment.mfx), 55
augment.poLCA, 70, 179, 334
augment.polr, 16, 72, 112, 114, 181, 207, 238, 240, 337, 375
augment.prcomp, 74, 340, 384, 388
augment.probimfx (augment.mfx), 55
augment.rlm, 76, 186, 349
augment.rma, 78
augment.rq, 63, 80, 83, 172, 189, 325, 354, 355
augment.rqs, 63, 81, 82, 172, 189, 325, 354, 355
augment.sarlm, 83, 190, 356
augment.smooth.spline, 85, 192
augment.speedlm, 87, 193, 195, 359, 361
augment.stl, 21, 88
augment.survecr, 19, 90, 100, 111, 118, 183, 199, 200, 202, 204, 217, 235, 246, 342, 367, 368, 370, 372
augment_columns, 92
base::as.data.frame.table(), 377
base::data.frame, 11, 14, 16, 18, 23, 25, 27, 29, 32, 34, 37, 40, 43, 45, 48, 50, 51, 53, 57, 59, 63, 64, 66, 68, 70, 73, 75, 76, 80, 82, 86, 87, 91
base::data.frame, 11, 14, 16, 18, 23, 29, 32, 34, 37, 41, 45, 48, 50, 51, 57, 63, 65, 73, 75, 77, 81, 83, 88, 91
base::svd(), 211, 387, 388
base::table, 377
bbmle::mle2(), 316, 317
bbmle_tidiers (tidy.mle2), 316
betareg::betareg(), 14, 105, 106, 225, 226
betareg::predict.betareg(), 12
betareg::residuals.betareg(), 12
betareg_tidiers (tidy.betareg), 225
biglm::biglm(), 107, 227, 228
biglm::biglm(), 107, 227, 228
bindesign_tidiers (tidy.binDesign), 228
binGroup::binDesign, 109
binGroup::binDesign(), 109, 229
binGroup::binWidth(), 230
binwidth_tidiers (tidy.binWidth), 230
boot::boot(), 231, 232
boot::boot.ci(), 231, 232
boot::tsboot(), 232
boot_tidiers (tidy.boot), 231
bootstrap, 93, 94, 96, 98, 99, 215, 251, 252, 267, 327
btergm::btergm(), 233
btergm_tidiers (tidy.btergm), 233
car::Anova(), 219
car::durbinWatsonTest(), 97
car::leveneTest(), 210, 211, 219–221
caret::confusionMatrix(), 244, 245
caret_tidiers (tidy.confusionMatrix), 244
cch_tidiers (tidy.cch), 234
cfa_tidiers (tidy.lavaan), 292
classification::pam(), 66, 67, 175, 176, 331
cmprsk::crr(), 119, 120, 248
cmprsk_tidiers (tidy.crr), 247
coeffest_tidiers (tidy.coefest), 241
coeffest(), 94
coeffest_tidiers (tidy.coefest), 241
confint(), 94
confint_tidiers (tidy.coefest), 241
deco_tidy (tidy.decompose), 215
deco_tidy (tidy.decompose), 215, 252, 267, 327
decompose_tidiers
(augment.decomposed.ts), 20
drc::drm(), 23, 24, 123, 124, 253, 254
drc_tidy (tidy.drc), 253
durbinWatsonTest_tidy, 97, 211
durbinWatsonTest_tidy, 97, 211
durbinWatsonTest_tidy, 97, 211
emmeans::contrast(), 255, 303, 345, 365
emmeans::emmeans(), 255, 303, 345, 365
emmeans::ref_grid(), 255, 303, 345, 365
emmeans::summary.emmGrid(), 255, 302, 345, 364, 365
emmeans::tidiers (tidy.lsmobj), 302
epiR::epi.2by2(), 257
epiR_tidiers (tidy.epi.2by2), 256
ergm::control.ergm(), 259
ergm::ergm(), 125, 258, 259
ergm::summary(), 125, 259
ergm::summary.ergm(), 125
ergm_tidiers (tidy.ergm), 258

factanal_tidiers (tidy.factanal), 260
felm_tidiers (tidy.felm), 261
finish_glance, 93, 94, 96, 98, 99, 215, 251, 252, 267, 327
fitdistr_tidiers (tidy.fitdistr), 263
fix_data_frame, 93, 94, 96, 98, 99, 215, 251, 252, 267, 327
fixest::feglm(), 30, 266
fixest::femlm(), 30, 266
fixest::fenegbin(), 30, 266
fixest::fnmlm(), 30, 266
fixest::feols(), 30, 266
fixest::feools(), 30, 266

geeglm_tidiers (tidy.geeglm), 272
geepack:::geglm(), 137, 272, 273
ggepack_tidiers (tidy.geepack), 272
gg() , 97, 100, 102, 106, 107, 109, 111, 116, 118, 120, 121, 124–126, 133, 135–137, 140, 143, 146, 147, 149, 151, 153, 154, 166–168, 170, 172, 174, 176, 177, 179, 183, 184, 186, 189, 190, 196, 199, 200, 202, 204, 208, 210, 211, 283
glance.aareg, 19, 91, 99, 111, 118, 183, 199, 200, 202, 204, 217, 235, 246, 342, 367, 368, 370, 372
glance.aov, 101, 219, 220, 222, 305, 380
glance.Arima, 102, 223
glance.betamfx, 12, 57, 103, 162, 224, 313
glance.betareg, 105
glance.betareg(). 104
glance.biglm, 106, 228
glance.binDesign, 108, 229, 230


glance.cch, 19, 91, 100, 110, 118, 183, 199, 200, 202, 204, 217, 235, 246, 342, 367, 368, 370, 372
glance.clm, 16, 74, 111, 114, 181, 207, 238, 240, 337, 375
glance.clm, 16, 74, 112, 113, 181, 207, 238, 240, 337, 375
glance.coeftest, 115
glance.coxph, 19, 91, 100, 111, 117, 183, 199, 200, 202, 204, 217, 235, 246, 342, 367, 368, 370, 372
glance.crr, 119, 248
glance.cv.glmnet, 120, 140, 250, 277
glance.data.frame (data.frame_tidiers), 95
glance.drc, 24, 123, 254
glance.durbinWatsonTest (durbinWatsonTest_tidiers), 97
glance.ergm, 124, 259
glance.factor, 26, 125, 260
glance.felm, 127
glance.fitdistr, 129, 264
glance.fixest, 130
glance.Gam, 132, 268
glance.gam, 134, 270
glance.gam(), 133
glance.garch, 135, 271
glance.geeglm, 136
glance.glm, 35, 46, 138, 151, 196, 205, 276, 295, 297, 318, 364
glance.glm(), 161, 162
glance.glmmnet, 121, 139, 250, 277
glance.glmmRob, 48, 141, 155, 278, 300
glance.gmm, 142, 281
glance.htest (tidy.htest), 283
glance.ivreg, 41, 144, 285
glance.kmeans, 43, 146, 291
glance.lavaan, 148, 293
glance.list (list_tidiers), 211
glance.lm, 35, 46, 139, 150, 196, 205, 276, 295, 297, 318, 364
glance.lm(), 196
glance.lmodel2, 152, 299
glance.lmRob, 48, 142, 154, 278, 300
glance.lmrob, 38, 50, 155, 280, 301
glance.logitmfx (gimfix.mfx), 160
glance.margins, 157
glance.Mclust, 159
glance.mfx, 12, 57, 104, 160, 224, 313
glance.mjolnir, 163, 315
glance.mlogit, 62, 165, 320
glance.muhaz, 166, 321
glance.multinom, 168, 322
glance.negbin, 169, 323
glance.negbinmfx (gimfix.mfx), 160
glance.nlrmq, 63, 81, 83, 171, 189, 325, 354,
INDEX

355
glance.nls, 65, 172, 326
glance.NULL (null_tidiers), 212
glance.optim (glance_optim), 209
glance.orcutt, 173, 328
glance.pam, 67, 175, 331
glance.plm, 69, 176, 333
glance.poissonmfx (glance.mfx), 160
glance.poLCA, 71, 178, 334
glance.polr, 16, 74, 112, 114, 180, 207, 238, 240, 337, 375
glance.probitmfx (glance.mfx), 160
glance.pyears, 19, 91, 100, 111, 118, 182, 199, 200, 202, 204, 217, 235, 246, 342, 367, 368, 370, 372
glance.ridgelm, 183, 348
glance.rlm, 77, 185, 349
glance.rma, 186
glance.rq, 63, 81, 83, 172, 188, 325, 354, 355
glance.sarlm, 85, 189, 356
glance.smooth.spline, 86, 191
glance.speedglm, 88, 192, 195, 359, 361
glance.speedlm, 88, 193, 194, 359, 361
glance.summary.lm, 35, 45, 139, 151, 195, 205, 276, 295, 297, 318, 364
glance.summary.lm(), 151, 196
glance.summaryDefault (summary_tidiers), 215
glance.survdiff, 19, 91, 100, 111, 118, 183, 198, 200, 202, 204, 217, 235, 246, 342, 367, 368, 370, 372
glance.survell, 19, 91, 100, 111, 118, 183, 199, 199, 202, 204, 217, 235, 246, 342, 367, 368, 370, 372
glance.survfit, 19, 91, 100, 111, 118, 183, 199, 200, 201, 204, 217, 235, 246, 342, 367, 368, 370, 372
glance.survexp, 19, 91, 100, 111, 118, 183, 199, 200, 202, 203, 217, 235, 246, 342, 367, 368, 370, 372
glance.survevreg, 19, 91, 100, 111, 118, 183, 199, 200, 202, 203, 217, 235, 246, 342, 367, 368, 370, 372
glance.svvglm, 35, 45, 139, 151, 196, 204, 276, 295, 297, 318, 364
glance.svyolr, 16, 74, 112, 114, 181, 206, 238, 240, 337, 375
glance.varest, 208
glance_optim, 209, 212, 384, 386, 388, 390
glm.nb_tidiers (glm.nb), 169
glmnet::cv.glmnet(), 121, 249, 250

397
glmnet::glmnet(), 139, 140, 276, 277
glmnet_tidiers (tidy_glmnet), 276
gmm::gmm(), 142, 143, 280, 281
gmm_tidiers (tidy_gmm), 280
graphics::image(), 389, 390
graphics::persp(), 389, 390

Hmisc::rcorr(), 343, 344
Hmisc_tidiers (tidy_rcorr), 343
htest_tidiers (tidy_htest), 283

irlba::irlba(), 383, 384
irlba_tidiers (tidy_irlba), 383
ivreg_tidiers (tidy_ivreg), 284

joineRML::bootSE(), 314, 315
joineRML::fitted.mjoint(), 59
joineRML::mjoint(), 59, 163, 164, 314, 315
joineRML::residuals.mjoint(), 59
joineRML_tidiers (tidy_mjoint), 314

kappa_tidiers (tidy_kappa), 286
kde_tidiers (tidy_kde), 288
Kendall::Kendall(), 290
Kendall::MannKendall(), 290
Kendall::SeasonalMannKendall(), 290
Kendall_tidiers (tidy_Kendall), 289
kendall_tidiers (tidy_Kendall), 289
kmeans_tidiers (tidy_kmeans), 291
ks::kde(), 288
ks_tidiers (tidy_kde), 288

lavaan::cfa(), 148, 149, 292, 293
lavaan::fitmeasures(), 149
lavaan::parameterEstimates(), 292, 293
lavaan::sem(), 148, 149, 292, 293
lavaan_tidiers (tidy_lavaan), 292
leaps::regsubsets(), 347
leaps_tidiers (tidy_regsubsets), 346
leveneTest_tidiers, 97, 210
lfe::felm(), 27, 128, 262
lfe_tidiers (tidy_felm), 261
list_tidiers, 210, 211, 384, 386, 388, 390
lm.beta::lm.beta, 296
lm_tidiers (tidy_lm), 294
lmodel2::lmodel2(), 153, 298, 299
lmodel2_tidiers (tidy_lmodel2), 298
lmtest::coeftest(), 115, 116, 241, 242
lmtest_tidiers (tidy_coeftest), 241
loess_tidiers(augment.loess), 51
lsmeans::summary.ref.grid(), 255, 302, 345, 364, 365

mass::polr(), 305, 306
mass::lm.ridge(), 305
mass::glm.nb(), 305

maps::map(), 305, 306
maps_tidiers(tidy.map), 305
margins::margins(), 157, 306, 307
margins_tidiers(tidy.margins), 306

mass::dropterm(), 336, 374
mass::fitdistr(), 129, 130, 264
mass::glm.nb(), 170, 323
mass::lm.ridge(), 184, 348
mass::polr(), 73, 180, 181, 336, 337
mass::rlm(), 76, 77, 185, 186, 349
mass::select.ridgelm(), 184
mclust::Mclust(), 53, 54, 159, 309
mclust_tidiers(tidy.Mclust), 308

mean, 95

mediate_tidiers(tidy.mediate), 310
mediation::mediate(), 310, 311
metafor::escalc(), 350
metafor::rma(), 78, 187, 350
metafor::rma.glmm(), 78, 187, 350
metafor::rma.mh(), 78, 187, 350
metafor::rma.mv(), 78, 187, 350
metafor::rma.peto(), 78, 187, 350
metafor::rma.uni(), 78, 187, 350

mfx::betamfx(), 12, 104, 224
mfx::logitmfx(), 57, 162, 313
mfx::negbimfx(), 57, 162, 313
mfx::poissonmfx(), 57, 162, 313
mfx::probitmfx(), 57, 162, 313
mgcv::gam(), 31, 32, 133–135, 268–270
mgcv_tidiers(tidy.gam), 268
mjoint_tidiers(tidy.mjoint), 314
ml2_tidiers(tidy.ml2), 316
mlogit::mlogit(), 61, 165, 166, 319, 320
mlogit_tidiers(tidy.mlogit), 319
muhaz::muhaz(), 167, 320, 321
muhaz_tidiers(tidy.muhaz), 320
multcomp::clsd(), 236
multcomp::confint.glht(), 236, 243
multcomp::glht(), 236, 243, 274, 361, 362
multcomp::summary.glht(), 236, 361, 362
multcomp_tidiers(tidy.glht), 273
multinom_tidiers(tidy.multinom), 321

nls_tidiers(tidy.nls), 325

nnet::multinom(), 168, 321, 322
nnet_tidiers(tidy.multinom), 321
null_tidiers, 212

optim_tidiers(tidy.optim), 385
orcutt::cochrane.orcutt(), 174, 328
orcuttidiers(tidy.orcutt), 328
ordinal::clm(), 16, 112, 237, 238
ordinal::clmm(), 114, 239, 240
ordinal::confint.clm(), 238, 240
ordinal::predict.clm(), 16
ordinal_tidiers(tidy.clm), 237

pam_tidiers(tidy.pam), 330
plm::plm(), 68, 69, 177, 332, 333
plm_tidiers(tidy.plm), 332
polc::polc(), 70, 71, 178, 179, 334
polc_tidiers(tidy.polc), 334
polr_tidiers(tidy.polr), 336
prcomp_tidiers(tidy.prcomp), 339
predict.fixest, 29
psych::cohen.kappa(), 286, 287
psych_tidiers(tidy.kappa), 286
purrr::map(), 189
purrr::map_df(), 212
pyeas_tidiers(tidy.pyeas), 341

qr, 304
quantreg::nlrq(), 63, 171, 172, 324, 325
quantreg::predict.rq, 81, 83
quantreg::predict.rq(), 81
quantreg::predict.rqs(), 83
quantreg::rq(), 80–83, 188, 189, 353–355
quantreg::summary.rq(), 353, 354
quantreg::summary.rqs(), 354
quantreg_tidiers(tidy.rq), 353
rcorr_tidiers(tidy.rcorr), 343
ridgelm_tidiers(tidy.ridgelm), 347
rlm_tidiers(glance.rlm), 185
robust::glmRob(), 141, 142, 278
robust::lmRob(), 48, 154, 155, 300
robust_tidiers(tidy.lmRob), 299
robustbase::glmRob(), 37, 38, 279, 280
robustbase::lmRob(), 50, 156, 301
robustbase_tidiers(tidy.lmRob), 300
roc_tidiers(tidy.roc), 351
roc_tidiers(tidy.rq), 353
rosltidiers(tidy.rqs), 354

INDEX
INDEX

rsample::bootstraps(). 232

sem_tidiers (tidy.lavaan), 292
sexpfit_tidiers (tidy.survexp), 368
smooth.spline_tidiers
  (augment.smooth.spline), 85
sp_tidiers. 214
sparse_tidiers. 213
spatialreg::errorsarlm(), 84, 190, 356
spatialreg::lagsarlm(), 84, 190, 356
spatialreg_tidiers (tidy.sarlm), 355
speedglm::speedglm(), 193, 358, 359
speedglm::speedglm(), 87, 88, 193–195, 360, 361
speedglm_tidiers (tidy.speedglm), 358
speededm_tidiers (tidy.speededlm), 360
splines::ns(), 11, 13, 15, 17, 20, 23, 25, 27,
  29, 31, 33, 35, 36, 38, 40, 42, 44, 48,
  49, 53, 55, 59, 61, 64, 66, 68, 70, 73,
  75, 76, 78, 80, 82, 84, 87, 89, 90
stats::acf(), 218
stats::anova(), 219
stats::aov(), 101, 220–222
stats::arima(), 102, 103, 222, 223
stats::ccf(), 218
stats::chisq.test(), 39, 283, 284
stats::cooks.distance(), 14, 32
stats::cor.test(), 39, 283, 284
stats::decompose(), 21
stats::density(), 251
stats::dist(), 252
stats::factanal(), 25, 26, 126, 260
stats::ftable(), 267
stats::glm(), 34, 35, 138, 139, 205, 275,
  276, 273
stats::kmeans(), 14, 34, 279
stats::lm(), 26, 44, 150, 196, 294, 318
stats::loess(), 51, 52
stats::manova(), 304
stats::na.action, 19, 46, 52
stats::nls(), 64, 65, 172, 173, 325, 326
stats::optim(), 209–211, 383, 385–387,
  389
stats::pca(), 218
stats::pairwise.t.test(), 329, 330
stats::pairwise.wilcox.test(), 329, 330
stats::poly(), 11, 13, 15, 17, 20, 23, 25, 27,
  29, 31, 33, 35, 36, 38, 40, 42, 44, 48,
  49, 53, 55, 59, 61, 64, 66, 68, 70, 73,
  75, 76, 78, 80, 82, 84, 87, 89, 90
stats::power.t.test(), 338
stats::prcomp(), 75, 339, 340
stats::predict(), 14, 18, 32, 37, 91
stats::predict.glm(), 34, 57
stats::predict.lm(), 46
stats::predict.loess(), 52
stats::predict.nls(), 65
stats::predict.smooth.spline(), 86
stats::residuals(), 14, 18, 32, 37, 91
stats::residuals.glm(), 34, 57
stats::rstandard.glm(), 34, 57
stats::smooth.spline(), 86, 191, 192
stats::spectrum(), 357, 358
stats::stl(), 89, 90
stats::summary.lm(), 295, 363, 364
stats::summary.manova(), 304
stats::summary.manova(), 305
stats::summary.nls(), 326
stats::t.test(), 39, 283, 284
stats::ts(), 378
stats::TukeyHSD(), 379, 380
stats::wilcox.test(), 39, 283, 284
summary(). 215
summary.fixest, 29, 131, 265
summary.survfit(). 201
summary_tidiers, 93, 94, 96, 98, 99, 201, 205,
  251, 252, 267, 277
survdiff_tidiers (tidy.survdiff), 366
survexp_tidiers (tidy.survexp), 368
survival::anova.svyglm(), 205
survey::svyglm(). 205, 373
survey::svyolr(). 207, 374, 375
survfit_tidiers (tidy.survfit), 369
survival::aareg(), 100, 216, 217
survival::cch(), 110, 111, 234, 235
survival::coxph(), 18, 19, 117, 118, 246
survival::pyears(), 182, 183, 341, 342
survival::Surv(), 11, 13, 16, 17, 20, 23, 25,
  27, 29, 31, 33, 36, 38, 40, 42, 44, 48,
  49, 53, 55, 59, 61, 64, 66, 68, 70, 73,
  75, 76, 78, 80, 82, 84, 87, 89, 90
survival::survdiff(), 198, 199, 367
survival::survexp(), 200, 368
survival::survfit(), 201, 202, 369, 370
survival::survreg(), 91, 203, 204, 371, 372
INDEX

tidy.ergm, 125, 258
tidy.factanal, 26, 126, 260
tidy.felm, 27, 261
tidy.fitdistr, 130, 263
tidy.fixest, 30, 264
tidy.ftable, 93, 94, 96, 98, 99, 215, 251, 252, 266, 327
tidy.Gam, 133, 267
tidy.gam, 135, 268
tidy.gam(), 268
tidy.garch, 136, 270
tidy.geeglm, 272
tidy.glht, 236, 243, 273, 362
tidy.glm, 35, 46, 139, 151, 196, 205, 275, 295, 297, 318, 364
tidy.glmm, 121, 140, 250, 276
tidy.glmmRob, 48, 142, 155, 278, 300
tidy.glmmrob, 38, 50, 156, 279, 301
tidy.gmm, 143, 280
tidy.htest, 39, 283, 330, 338
tidy.irlba (tidy.irlba), 383
tidy.ivreg, 41, 146, 284
tidy.kappa, 286
tidy.kde, 288
tidy.Kendall, 289
tidy.kmeans, 43, 147, 291
tidy.lavaan, 149, 292
tidy.leveneTest (leveneTest_tidiers), 210
tidy.leveneTest(), 219–221
tidy.Line (sp_tidiers), 214
tidy.Lines (sp_tidiers), 214
tidy.list (list_tidiers), 211
tidy.lm, 35, 46, 139, 151, 196, 205, 276, 294, 297, 318, 364
tidy.lm(), 333, 361, 363
tidy.lm.beta, 35, 46, 139, 151, 196, 205, 276, 295, 296, 318, 364
tidy.lmodel2, 153, 298
tidy.lmRob, 48, 142, 155, 278, 299
tidy.lmrob, 38, 50, 156, 280, 300
tidy.logical (tidy.numeric), 327
tidy.logitmfx (tidy.mfx), 311
tidy.lsmobj, 255, 302, 345, 365
tidy.manova, 102, 219, 220, 222, 304, 380
tidy.map, 305
tidy.margins, 306
tidy.Mclust, 54, 308
tidy.mediate, 310
tidy.mfx, 12, 57, 104, 162, 224, 311
tidy.mjoint, 164, 314
tidy.mle2, 316
tidy.mlm, 35, 46, 139, 151, 196, 205, 276, 295, 297, 317, 364
tidy.mlm(), 294
tidy.mlogit, 62, 166, 319
tidy.muhaz, 167, 320
tidy.multinom, 168, 321
tidy.negbin, 170, 323
tidy.negbinmfx (tidy.mfx), 311
tidy.nlrq, 63, 81, 83, 172, 189, 324, 354, 355
tidy.nls, 65, 173, 325
tidy.NULL (null_tidiers), 212
tidy.numeric, 93, 94, 96, 98, 99, 215, 251, 252, 267, 327
tidy.optim (tidy_optim), 385
tidy.orcutt, 174, 328
tidy.pairwise.htest, 39, 284, 329, 338
tidy.pam, 67, 176, 330
tidy.plm, 69, 177, 332
tidy.poissonmfx (tidy.mfx), 311
tidy.polic, 71, 179, 334
tidy.polr, 16, 74, 112, 114, 181, 207, 238, 240, 336, 375
tidy.Polygon (sp_tidiers), 214
tidy.Polygons (sp_tidiers), 214
tidy.power.htest, 39, 284, 330, 338
tidy.prcomp, 75, 339, 384, 388
tidy.probitmfx (tidy.mfx), 311
tidy.pyeas, 19, 91, 100, 111, 118, 183, 199, 200, 202, 204, 217, 235, 246, 341, 367, 368, 370, 372
tidy.rcorr, 343
tidy.ref.grid, 255, 303, 344, 365
tidy.regsubsets, 346
tidy.ridgelm, 184, 347
tidy.rlm, 77, 186, 349
tidy.rlm(), 37, 48, 50, 156, 278, 279, 300, 301
tidy.rma, 350
tidy.roc, 351
tidy.rq, 63, 81, 83, 172, 189, 325, 353, 355
tidy.rqs, 63, 81, 83, 172, 189, 325, 354, 354
tidy.sarlm, 85, 190, 355
tidy.sparseMatrix (sparse_tidiers), 213
tidy.SpatialLinesDataFrame
(sp_tidiers), 214
tidy.SpatialPolygons (sp_tidiers), 214
tidy.SpatialPolygonsDataFrame
(sp_tidiers), 214
tidy.spec, 218, 357, 378, 382
tidy.speedglm, 88, 193, 195, 358, 361
tidy.speedlm, 88, 193, 195, 359, 360
tidy.summary.glht, 236, 243, 274, 361
tidy.summary.lm, 35, 46, 139, 151, 196, 205, 276, 295, 297, 318, 363
tidy.summary_emm, 255, 303, 345, 364
tidy.summaryDefault (summary_tidiers), 215
tidy.survdiff, 19, 91, 100, 111, 118, 183, 199, 200, 202, 204, 217, 235, 246, 342, 366, 368, 370, 372
tidy.survexp, 19, 91, 100, 111, 118, 183, 199, 200, 202, 204, 217, 235, 246, 342, 367, 368, 370, 372
tidy.survfit, 19, 91, 100, 111, 118, 183, 199, 200, 202, 204, 217, 235, 246, 342, 367, 368, 369, 372
tidy.survreg, 19, 91, 100, 111, 118, 183, 199, 200, 202, 204, 217, 235, 246, 342, 367, 368, 370, 371
tidy.svysurvdiff, 19, 91, 100, 111, 118, 183, 199, 200, 202, 204, 217, 235, 246, 342, 366, 368, 370, 372
zoo::zoo(), 382
zoo_tidiers(tidy.zoo), 382