Package ‘broom’

July 9, 2020

**Type**        Package

**Title**       Convert Statistical Objects into Tidy Tibbles

**Version**     0.7.0

**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

**URL**         https://broom.tidymodels.org/, http://github.com/tidymodels/broom

**BugReports**  http://github.com/tidymodels/broom/issues

**Depends**     R (>= 3.1)

**Imports**     backports, dplyr, ellipsis, generics (>= 0.0.2), glue, methods, purrr, rlang, stringr, tibble (>= 3.0.0), tidy

**Suggests**    AER, akima, AUC, bbmle, betareg, biglm, binGroup, boot, btergm, car, caret, cluster, coda, covr, drc, e1071, emmeans, epiR, ergm, fixest (>= 0.3.1), gam (>= 1.15), gamlss, gamlss.data, gamlss.dist, gee, geepack, ggplot2, glmnet, glmnetUtils, gmm, Hmisc, irlba, joineRML, Kendall, knitr, ks, Lahman, lavaan, leaps, lfe, lm.beta, lme4, lmodel2, lptest, lmeans, maps, maptools, MASS, Matrix, mclogit, mclust, mediation, metafor, mfx, mgcv, modeldata, modelestes, muhaz, multcomp, network, nnet, orcutt (>= 2.2), ordinal, plm, poLCA, psych, quantreg, rgeos, rmarkdown, robust, robustbase, rsample, sandwich, sp, spdep, spatialreg, speedglm, spelling, statnet.common, survey, survival, systemfit, testthat (>= 2.1.0), tseries, zoo

**VignetteBuilder**    knitr
Encoding UTF-8
LazyData true
RoxygenNote 7.1.0.9000
Language en-US
Collate 'aaa-documentation-helper.R' 'null-and-default-tidiers.R'
'aer-tidiers.R' 'auc-tidiers.R' 'base-tidiers.R'
'bmmc-tidiers.R' 'betareg-tidiers.R' 'biglm-tidiers.R'
'bingroup-tidiers.R' 'boot-tidiers.R' 'broom-package.R'
'broom.R' 'btergm-tidiers.R' 'car-tidiers.R' 'caret-tidiers.R'
'data-frame-tidiers.R' 'deprecated-0-7-0.R' 'drc-tidiers.R'
'emmeans-tidiers.R' 'epiR-tidiers.R' 'ergm-tidiers.R'
'fixest-tidiers.R' 'gam-tidiers.R' 'gamss-tidiers.R' 'gee.R'
'geepack-tidiers.R' 'glmnet-cv-glmnet-tidiers.R'
'glmnet-glmnet-tidiers.R' 'gmm-tidiers.R' 'hmisc-tidiers.R'
'joineerm-tidiers.R' 'kendall-tidiers.R' 'ks-tidiers.R'
'lavaan-tidiers.R' 'leaps.R' 'lfe-tidiers.R' 'list-irlba.R'
'list-optim-tidiers.R' 'list-svd-tidiers.R' 'list-tidiers.R'
'list-xz-tidiers.R' 'lm-beta-tidiers.R' 'lmodel2-tidiers.R'
'lmtest-tidiers.R' 'maps-tidiers.R' 'mass-fitdistr-tidiers.R'
'mass-polr-tidiers.R' 'mass-ridgeglm-tidiers.R'
'mass-lm-tidiers.R' 'mass-rlm-tidiers.R' 'matrix-tidiers.R'
'mclust.R' 'mclogist.R' 'mclust-tidiers.R' 'mediate-tidiers.R'
'mfx-tidiers.R' 'mgcv-tidiers.R' 'muhaz-tidiers.R'
'multcomp-tidiers.R' 'mnet-tidiers.R' 'nobs.R'
'orcutt-tidiers.R' 'ordinal-clm-tidiers.R'
'ordinal-clmm-tidiers.R' 'pam-tidiers.R' 'plm-tidiers.R'
'polca-tidiers.R' 'psych-tidiers.R' 'stats-nls-tidiers.R'
'quantreg-nlrq-tidiers.R' 'quantreg-rq-tidiers.R'
'quantreg-rqs-tidiers.R' 'rma-tidiers.R'
'robus-glmrob-tidiers.R' 'robus-lmrob-tidiers.R'
'robustbase-glmrob-tidiers.R' 'robustbase-lmrob-tidiers.R'
'scam-tidiers.R' 'sp-tidiers.R' 'spdep-tidiers.R'
'speedglm-speedglm-tidiers.R' 'speedglm-speedlm-tidiers.R'
'stats-anova-tidiers.R' 'stats-arima-tidiers.R'
'stats-decompose-tidiers.R' 'stats-factanal-tidiers.R'
'stats-glm-tidiers.R' 'stats-htest-tidiers.R'
'stats-kmeans-tidiers.R' 'stats-loess-tidiers.R'
'stats-mlm-tidiers.R' 'stats-pcomp-tidiers.R'
'stats-smooth.spline-tidiers.R' 'stats-time-series-tidiers.R'
'survey-tidiers.R' 'survival-aareg-tidiers.R'
'survival-cch-tidiers.R' 'survival-coxph-tidiers.R'
'survival-pyears-tidiers.R' 'survival-survdiff-tidiers.R'
'survival-survexp-tidiers.R' 'survival-survfit-tidiers.R'
'survival-survreg-tidiers.R' 'systemfit-tidiers.R'
'tseries-tidiers.R' 'utilities.R' 'zoo-tidiers.R' 'zzz.R'

NeedsCompilation no
Author David Robinson [aut],
R topics documented:

Shabbh 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]

Maintainer  Alex Hayes <alexpghayes@gmail.com>

Repository  CRAN

Date/Publication  2020-07-09 12:30:09 UTC

R topics documented:

augment.betamfx ....................................................... 10
augment.betareg ..................................................... 12
augment.clm .......................................................... 14
augment.coxph ......................................................... 16
augment.decomposed.ts ............................................. 19
augment.drc ........................................................... 21
augment.factanal ...................................................... 23
augment.felm .......................................................... 25
augment.fixest ......................................................... 27
augment.glm .......................................................... 29
augment.glmRob ......................................................... 31
augment.glmrob ......................................................... 32
augment.htest .......................................................... 34
augment.ivreg .......................................................... 36
augment.kmeans ........................................................ 38
augment.lm ............................................................ 39
augment.lmRob ........................................................ 42
augment.lmrob ........................................................ 44
augment.loess .......................................................... 46
<table>
<thead>
<tr>
<th>R topics documented:</th>
</tr>
</thead>
<tbody>
<tr>
<td>augment.Mclust</td>
</tr>
<tr>
<td>augment.mfx</td>
</tr>
<tr>
<td>augment.mjoint</td>
</tr>
<tr>
<td>augment.nlrq</td>
</tr>
<tr>
<td>augment.nls</td>
</tr>
<tr>
<td>augment.pam</td>
</tr>
<tr>
<td>augment.plm</td>
</tr>
<tr>
<td>augment.polCA</td>
</tr>
<tr>
<td>augment.polr</td>
</tr>
<tr>
<td>augment.prcomp</td>
</tr>
<tr>
<td>augment.rlm</td>
</tr>
<tr>
<td>augment.rma</td>
</tr>
<tr>
<td>augment.rq</td>
</tr>
<tr>
<td>augment.rqs</td>
</tr>
<tr>
<td>augment.sarlm</td>
</tr>
<tr>
<td>augment.smooth.spline</td>
</tr>
<tr>
<td>augment.speedlm</td>
</tr>
<tr>
<td>augment.stl</td>
</tr>
<tr>
<td>augment.survsreg</td>
</tr>
<tr>
<td>augment_columns</td>
</tr>
<tr>
<td>bootstrap</td>
</tr>
<tr>
<td>confint_tidy</td>
</tr>
<tr>
<td>data.frame_tidiers</td>
</tr>
<tr>
<td>durbinWatsonTest_tidiers</td>
</tr>
<tr>
<td>finish_glance</td>
</tr>
<tr>
<td>fix_data_frame</td>
</tr>
<tr>
<td>glance.aareg</td>
</tr>
<tr>
<td>glance.aov</td>
</tr>
<tr>
<td>glance.Arima</td>
</tr>
<tr>
<td>glance.betamfx</td>
</tr>
<tr>
<td>glance.betareg</td>
</tr>
<tr>
<td>glance.bigm</td>
</tr>
<tr>
<td>glance.binDesign</td>
</tr>
<tr>
<td>glance.cch</td>
</tr>
<tr>
<td>glance.clm</td>
</tr>
<tr>
<td>glance.clmm</td>
</tr>
<tr>
<td>glance.coxph</td>
</tr>
<tr>
<td>glance.cv.glmnet</td>
</tr>
<tr>
<td>glance.drc</td>
</tr>
<tr>
<td>glance.ergm</td>
</tr>
<tr>
<td>glance.factanal</td>
</tr>
<tr>
<td>glance.felm</td>
</tr>
<tr>
<td>glance.fitdistr</td>
</tr>
<tr>
<td>glance.fixest</td>
</tr>
<tr>
<td>glance.gam</td>
</tr>
<tr>
<td>glance.garch</td>
</tr>
<tr>
<td>glance.geeglm</td>
</tr>
<tr>
<td>glance.glm</td>
</tr>
<tr>
<td>Function</td>
</tr>
<tr>
<td>--------------</td>
</tr>
<tr>
<td>glance.glmnet</td>
</tr>
<tr>
<td>glance.glmRob</td>
</tr>
<tr>
<td>glance.gmm</td>
</tr>
<tr>
<td>glance.ivreg</td>
</tr>
<tr>
<td>glance.kmeans</td>
</tr>
<tr>
<td>glance.lavaan</td>
</tr>
<tr>
<td>glance.lm</td>
</tr>
<tr>
<td>glance.lmodel2</td>
</tr>
<tr>
<td>glance.lmRob</td>
</tr>
<tr>
<td>glance.lmrob</td>
</tr>
<tr>
<td>glance.Mclust</td>
</tr>
<tr>
<td>glance.mfx</td>
</tr>
<tr>
<td>glance.mjoint</td>
</tr>
<tr>
<td>glance.muhaz</td>
</tr>
<tr>
<td>glance.multinom</td>
</tr>
<tr>
<td>glance.nlrq</td>
</tr>
<tr>
<td>glance.nls</td>
</tr>
<tr>
<td>glance.orcutt</td>
</tr>
<tr>
<td>glance.pam</td>
</tr>
<tr>
<td>glance.plm</td>
</tr>
<tr>
<td>glance.poLCA</td>
</tr>
<tr>
<td>glance.polr</td>
</tr>
<tr>
<td>glance.pyears</td>
</tr>
<tr>
<td>glance.ridgelm</td>
</tr>
<tr>
<td>glance.rlm</td>
</tr>
<tr>
<td>glance.rma</td>
</tr>
<tr>
<td>glance.rq</td>
</tr>
<tr>
<td>glance.sarlm</td>
</tr>
<tr>
<td>glance.smooth.spline</td>
</tr>
<tr>
<td>glance.speedglm</td>
</tr>
<tr>
<td>glance.speedlm</td>
</tr>
<tr>
<td>glance.survdiff</td>
</tr>
<tr>
<td>glance.survexp</td>
</tr>
<tr>
<td>glance.survfit</td>
</tr>
<tr>
<td>glance.survreg</td>
</tr>
<tr>
<td>glance.svyglm</td>
</tr>
<tr>
<td>glance.svyolr</td>
</tr>
<tr>
<td>glance_optim</td>
</tr>
<tr>
<td>list_tidiers</td>
</tr>
<tr>
<td>null_tidiers</td>
</tr>
<tr>
<td>sparse_tidiers</td>
</tr>
<tr>
<td>sp_tidiers</td>
</tr>
<tr>
<td>summary_tidiers</td>
</tr>
<tr>
<td>tidy.aareg</td>
</tr>
<tr>
<td>tidy.acf</td>
</tr>
<tr>
<td>tidy.anova</td>
</tr>
<tr>
<td>tidy.aov</td>
</tr>
<tr>
<td>tidy.aovlist</td>
</tr>
</tbody>
</table>
R topics documented:

<table>
<thead>
<tr>
<th>Package</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>tidy.Arima</td>
<td>194</td>
</tr>
<tr>
<td>tidy.betamfx</td>
<td>195</td>
</tr>
<tr>
<td>tidy.betareg</td>
<td>197</td>
</tr>
<tr>
<td>tidy.biglm</td>
<td>199</td>
</tr>
<tr>
<td>tidy.binDesign</td>
<td>200</td>
</tr>
<tr>
<td>tidy.binWidth</td>
<td>201</td>
</tr>
<tr>
<td>tidy.boot</td>
<td>203</td>
</tr>
<tr>
<td>tidy.btergm</td>
<td>204</td>
</tr>
<tr>
<td>tidy.cch</td>
<td>206</td>
</tr>
<tr>
<td>tidy.cld</td>
<td>208</td>
</tr>
<tr>
<td>tidy.clm</td>
<td>209</td>
</tr>
<tr>
<td>tidy.clmm</td>
<td>211</td>
</tr>
<tr>
<td>tidy.coeftest</td>
<td>213</td>
</tr>
<tr>
<td>tidy.confint.glht</td>
<td>214</td>
</tr>
<tr>
<td>tidy.confusionMatrix</td>
<td>215</td>
</tr>
<tr>
<td>tidy.coxph</td>
<td>217</td>
</tr>
<tr>
<td>tidy.cv.glmnet</td>
<td>219</td>
</tr>
<tr>
<td>tidy.density</td>
<td>221</td>
</tr>
<tr>
<td>tidy.dist</td>
<td>222</td>
</tr>
<tr>
<td>tidy.drc</td>
<td>223</td>
</tr>
<tr>
<td>tidy.emmGrid</td>
<td>224</td>
</tr>
<tr>
<td>tidy.epi.2by2</td>
<td>226</td>
</tr>
<tr>
<td>tidy.ergm</td>
<td>228</td>
</tr>
<tr>
<td>tidy.factanal</td>
<td>230</td>
</tr>
<tr>
<td>tidy.felm</td>
<td>231</td>
</tr>
<tr>
<td>tidy.fitdistr</td>
<td>233</td>
</tr>
<tr>
<td>tidy.fixest</td>
<td>234</td>
</tr>
<tr>
<td>tidy.ftable</td>
<td>236</td>
</tr>
<tr>
<td>tidy.gam</td>
<td>237</td>
</tr>
<tr>
<td>tidy.gamlss</td>
<td>238</td>
</tr>
<tr>
<td>tidy.garch</td>
<td>239</td>
</tr>
<tr>
<td>tidy.geeglm</td>
<td>240</td>
</tr>
<tr>
<td>tidy.glht</td>
<td>242</td>
</tr>
<tr>
<td>tidy.glm</td>
<td>243</td>
</tr>
<tr>
<td>tidy.glmnet</td>
<td>244</td>
</tr>
<tr>
<td>tidy.glmRob</td>
<td>246</td>
</tr>
<tr>
<td>tidy.glmrob</td>
<td>247</td>
</tr>
<tr>
<td>tidy.gmm</td>
<td>248</td>
</tr>
<tr>
<td>tidy.htest</td>
<td>251</td>
</tr>
<tr>
<td>tidy.ivreg</td>
<td>252</td>
</tr>
<tr>
<td>tidy.kappa</td>
<td>254</td>
</tr>
<tr>
<td>tidy.kde</td>
<td>255</td>
</tr>
<tr>
<td>tidy.Kendall</td>
<td>257</td>
</tr>
<tr>
<td>tidy.kmeans</td>
<td>258</td>
</tr>
<tr>
<td>tidy.lavaan</td>
<td>259</td>
</tr>
<tr>
<td>tidy.lm</td>
<td>261</td>
</tr>
<tr>
<td>tidy.lm.beta</td>
<td>263</td>
</tr>
<tr>
<td>tidy.lmodel2</td>
<td>265</td>
</tr>
</tbody>
</table>
tidy.lmRob ......................................................... 266
 tidy.lmrob ...................................................... 267
 tidy.lsmobj ...................................................... 268
 tidy.manova ..................................................... 270
 tidy.map ......................................................... 272
 tidy.Mclust ..................................................... 273
 tidy.mediate ..................................................... 274
 tidy.mfx ......................................................... 276
 tidy.mjoint ...................................................... 278
 tidy.mle2 ......................................................... 280
 tidy.mlm ........................................................ 282
 tidy.muhaz ....................................................... 283
 tidy.multinom ................................................... 284
 tidy.nlrq ........................................................ 285
 tidy.nls ........................................................ 287
 tidy.numeric ..................................................... 288
 tidy.orcutt ...................................................... 289
 tidy.pairwise.htest ............................................. 290
 tidy.pam ......................................................... 292
 tidy.plm ........................................................ 293
 tidy.polr ........................................................ 296
 tidy.power.htest ................................................ 298
 tidy.prcomp ..................................................... 300
 tidy.pyears ..................................................... 302
 tidy.rcorr ....................................................... 303
 tidy.ref.grid ..................................................... 305
 tidy.regsubsets ................................................ 307
 tidy.ridgelm .................................................... 308
 tidy.rlm ........................................................ 309
 tidy.rma ........................................................ 310
 tidy.roc ......................................................... 312
 tidy.rq .......................................................... 313
 tidy.rqs ........................................................ 315
 tidy.sarlm ....................................................... 316
 tidy.spec ........................................................ 317
 tidy.speedglm ................................................. 318
 tidy.speedlm ................................................... 320
 tidy.summary.glht .............................................. 321
 tidy.summary_emm .............................................. 322
 tidy.survdiff .................................................... 324
 tidy.survexp .................................................... 326
 tidy.survfit ..................................................... 327
 tidy.survreg ..................................................... 329
 tidy.svyglm ...................................................... 330
 tidy.svyolr ...................................................... 331
 tidy.systemfit .................................................. 333
 tidy.table ....................................................... 334
augment.betamfx

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 all columns used to fit the model are present.

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"),
)```
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 betareg::predict.betareg(). Defaults to "response".
type.residuals  Character indicating type of residuals to use. Passed to the type argument of betareg::residuals.betareg(). Defaults to "sweighted2".

...  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 augment method wraps augment.betareg() for mfx::betamfx() objects.

Value

A tibble::tibble() with columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.cooks</td>
<td>Cooks distance.</td>
</tr>
<tr>
<td>.fitted</td>
<td>Fitted or predicted value.</td>
</tr>
<tr>
<td>.resid</td>
<td>The difference between observed and fitted values.</td>
</tr>
</tbody>
</table>

See Also

augment.betareg(), mfx::betamfx()

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

### 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 all columns used to fit the model are present.

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 '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 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.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 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()`

Examples

```r
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 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 all columns used to fit the model are present.

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 'clm'
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
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)

```

---

**`augment.coxph`**

*Augment data with information from a(n) coxph 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 all columns used to fit the model are present.

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 `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 '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.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.

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
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)

c1 <- clogit(case ~ tocc + tocc:education + strata(id), logan2)
```
```r
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()
```

---

**augment.decomposed.ts**  
*Augment data with information from a(n) decomposed.ts 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 all columns used to fit the model are present.

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 '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)
```
```r
decoms <- 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 a(n) 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 all columns used to fit the model are present.

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.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 confidence interval for fitted values.
- `.conf.low`: Lower bound on confidence interval for fitted values.
- `.cooks.d`: Cooks distance.
- `.fitted`: Fitted or predicted value.
- `.resid`: The difference between observed and fitted values.
- `.se.fit`: Standard errors of fitted values.

### See Also

- `augment()`, `drc::drm()`
- Other `drc` tidiers: `glance.drc()`, `tidy.drc()`

### Examples

```r
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)
```

## 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 all columns used to fit the model are present.

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 '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.
- `...` 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

When data is not supplied augment.factanal returns one row for each observation, with a factor score column added for each factor X, (.fSX). This is because stats::factanal(), unlike other stats methods like stats::lm(), does not retain the original data.

When data is supplied, augment.factanal returns one row for each observation, with a factor score column added for each factor X, (.fSX).
### 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 all columns used to fit the model are present.

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 original 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.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(), lfe::felm()

Other felm tidiers: tidy.felm()

Examples

library(lfe)

N <- 1e2
DT <- data.frame(
  id = sample(5, N, TRUE),
  v1 = sample(5, N, TRUE),
  v2 = sample(1e6, N, TRUE),
  v3 = sample(round(runif(100, max = 100), 4), N, TRUE),
  v4 = sample(round(runif(100, max = 100), 4), N, TRUE)
)

result_felm <- felm(v2 ~ v3, DT)
tidy(result_felm)
augment(result_felm)

result_felm <- felm(v2 ~ v3 | id + v1, DT)
tidy(result_felm, fe = TRUE)
tidy(result_felm, robust = TRUE)
augment(result_felm)

v1 <- DT$v1
v2 <- DT$v2
v3 <- DT$v3
id <- DT$id
result_felm <- felm(v2 ~ v3 | id + v1)

tidy(result_felm)
augment(result_felm)
glance(result_felm)
**augment.fixest**

**Augment data with information from a(n) fixest 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 all columns used to fit the model are present.

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 'fixest'
augment(x, data = NULL, newdata = NULL, type.predict = "response", ...)
```

**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 `glm.predict`).

...  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

```r
library(fixest)
gavity <- feols(log(Euros) ~ log(dist_km) | Origin + Destination + Product + Year, trade)
tidy(gavity)
glance(gavity)
augment(gavity, trade)

# To get robust or clustered SEs, users can either:
tidy(gavity, conf.int = TRUE, cluster = c("Product", "Year"))
tidy(gavity, conf.int = TRUE, se = "threeway")
# 2) Feed tidy() a summary.fixest object that has already accepted these arguments
gavity_summ <- summary(gavity, cluster = c("Product", "Year"))
tidy(gavity_summ, conf.int = TRUE)
# Approach (1) is preferred.

# The other fixest methods all work similarly. For example:
gavity_pois <- feglm(Euros ~ log(dist_km) | Origin + Destination + Product + Year, trade)
tidy(gavity_pois)
glance(gavity_pois)
augment(gavity_pois, trade)
```
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 all columns used to fit the model are present.

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 '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.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 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:

- .cooks The difference between observed and fitted values.
- .fit Fitted or predicted value.
- .hat Diagonal of the hat matrix.
- .sigma Standard errors of fitted values.
- .std.resid Standardised residuals.

See Also

stats::glm()

Other lm tidiers: augment.lm(), glance.glm(), glance.lm(), glance.svyglm(), tidy.glm(), tidy.lm.beta(), tidy.lm(), tidy.mlm()
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 \texttt{.fitted} column, residuals in
the \texttt{.resid} column, and standard errors for the fitted values in a \texttt{.se.fit} column. New columns
always begin with a \texttt{.} prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the \texttt{data} argument or the \texttt{newdata} argument. If the user
passes data to the \texttt{data} argument, it \textbf{must} be exactly the data that was used to fit the model object.
Pass datasets to \texttt{newdata} to augment data that was not used during model fitting. This still requires
that all columns used to fit the model are present.

Augment will often behave differently depending on whether \texttt{data} or \texttt{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 \texttt{data} arguments, so that \texttt{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 \texttt{tibble::tibble} with the \textbf{same number of rows} as the
passed dataset. This means that the passed data must be coercible to a \texttt{tibble}. At this time, \texttt{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 \texttt{splines::ns()}, \texttt{stats::poly()} and
\texttt{survival::Surv()} objects are not supported in input data. If you encounter errors, try explicitly
passing a \texttt{tibble}, or fitting the original model on data in a \texttt{tibble}.

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

### Usage

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

### Arguments

- \texttt{x} Unused.
- \texttt{...} Unused.
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 all columns used to fit the model are present.

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 '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 Cook's 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.

`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()`
Augment data with information from an htest 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 all columns used to fit the model are present.

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: Missspelled 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.
augment.htest

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

tt <- t.test(rnorm(10))
tidy(tt)
glance(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)

c <- cor.test(mtcars$wt, mtcars$mpg)
tidy(c)

chit <- chisq.test(xtabs(Freq ~ Sex + Class, data = as.data.frame(Titanic)))
tidy(chit)
augment(chit)
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 all columns used to fit the model are present.

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 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.
augment.ivreg

... 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(), AER::ivreg()

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

Examples

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)
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 \texttt{.fitted} column, residuals in the \texttt{.resid} column, and standard errors for the fitted values in a \texttt{.se.fit} column. New columns always begin with a \texttt{.} prefix to avoid overwriting columns in the original dataset.

Users may pass data to augment via either the \texttt{data} argument or the \texttt{newdata} argument. If the user passes data to the \texttt{data} argument, it \textbf{must} be exactly the data that was used to fit the model object. Pass datasets to \texttt{newdata} to augment data that was not used during model fitting. This still requires that all columns used to fit the model are present.

Augment will often behave differently depending on whether \texttt{data} or \texttt{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 \texttt{data} arguments, so that \texttt{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 \texttt{tibble::tibble} with the \textbf{same number of rows} as the passed dataset. This means that the passed data must be coercible to a \texttt{tibble}. At this time, \texttt{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 \texttt{splines::ns()}, \texttt{stats::poly()} and \texttt{survival::Surv()} objects are not supported in input data. If you encounter errors, try explicitly passing a \texttt{tibble}, or fitting the original model on data in a \texttt{tibble}.

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

Usage

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

Arguments

- \texttt{x} A \texttt{kmeans} object created by \texttt{stats::kmeans()}.

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

- \texttt{...} 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 `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.

See Also

`augment()`

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)
```

---

**Description**

`augment.lm` Augment data with information from a(n) lm 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 all columns used to fit the model are present.
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 'lm'
augment(x, data = model.frame(x), newdata = NULL, se.fit = FALSE, ...)
```

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.
- **...**: 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

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

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

Examples

```r
library(ggplot2)
library(dplyr)

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

# coefficient plot
d <- tidy(mod) %>%
  mutate(
    low = estimate - std.error,
    high = estimate + std.error
  )

ggplot(d, aes(estimate, term, xmin = low, xmax = high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0) +
```
```r
geom_errorbarh()

augment(mod)
augment(mod, mtcars)

# predict on new data
ewdata <- mtcars %>%
  head(6) %>%
  mutate(wt = wt + 1)
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)
```

---

**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 all columns used to fit the model are present.

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, ...)
```

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.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::lmRob()

Other robust tidiers: `glance.glmRob()`, `glance.lmRob()`, `tidy.glmRob()`, `tidy.lmRob()`
augment.lmrob

Examples

```r
library(robust)
m <- lmRob(mpg ~ wt, data = mtcars)
tidy(m)
augment(m)
glance(m)
```

### augment.lmrob

**Augment data with information from an 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 all columns used to fit the model are present.

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, 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)

# 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)

--

augment.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.
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.

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.

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(), 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))
```
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 all columns used to fit the model are present.

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 'Mclust'
augment(x, data = NULL, ...)  
```

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

```r
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)
  )
points <- points %>%
  dplyr::select(-num_points, -cluster)
points <- tidyr::unnest(c(x1, x2))

m <- mclust::Mclust(points)
tidy(m)
augment(m, points)
glance(m)
```
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 \texttt{data} argument or the \texttt{newdata} argument. If the user passes data to the \texttt{data} argument, it \textbf{must} be exactly the data that was used to fit the model object. Pass datasets to \texttt{newdata} to augment data that was not used during model fitting. This still requires that all columns used to fit the model are present.

Augment will often behave differently depending on whether \texttt{data} or \texttt{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 \texttt{data} arguments, so that \texttt{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 \texttt{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 \texttt{splines::ns()}, \texttt{stats::poly()} and \texttt{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 \texttt{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,
  ...
)
```

```r
## S3 method for class 'logitmfx'
augment(
  x,
  data = model.frame(x$fit),
```
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:

- **.cooks.d**
  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()
**Examples**

```r
## 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)
```

---

**augment.mjoint**

*Augment data with information from an 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 all columns used to fit the model are present.

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 '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 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.

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

```r
## 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)

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.nls

Augment data with information from a(n) nls 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 all columns used to fit the model are present.

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 '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.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**

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

**Value**

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

<table>
<thead>
<tr>
<th>.fitted</th>
<th>Fitted or predicted value.</th>
</tr>
</thead>
<tbody>
<tr>
<td>.resid</td>
<td>The difference between observed and fitted values.</td>
</tr>
</tbody>
</table>

**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 all columns used to fit the model are present.

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 '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.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.
- `.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 all columns used to fit the model are present.

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
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)

# Augment data with information from a(n) poLCA 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 all columns used to fit the model are present.

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 'poLCA'
augment(x, data = NULL, ...)
```

Arguments

- `x` A `poLCA` object returned from `poLCA::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
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
es2a <- 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)
```

`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 all columns used to fit the model are present.

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 Cook's 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.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

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

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)
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. **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.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 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()
**augment.rlm**

**Augment data with information from a(n) rlm 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 all columns used to fit the model are present.

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.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.
- .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

library(MASS)
r <- rlm(stack.loss ~ ., stackloss)
tidy(r)
augment(r)
glance(r)
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 all columns used to fit the model are present.

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 'rma'
augment(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.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.

Value

A tibble::tibble() with columns:

- `.conf.high` Upper bound on confidence interval for fitted values.
- `.conf.low` Lower bound on confidence interval for fitted values.
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 all columns used to fit the model are present.

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 newdata. If newdata is specified, the data argument will be ignored.
- **...**: Arguments passed on to quantreg::predict.rq

Object of class rq or rqs or rq.process produced by 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**: Converage 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 .conf.low and .conf.high. Does not provide confidence intervals when data is specified via the newdata argument.

Value

A tibble::tibble() with columns:

- .fitted: Fitted or predicted value.
- .resid: The difference between observed and fitted values.
- .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()
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 '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.
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.
... Arguments passed on to quantreg::predict.rq

object object of class rq or rqs or rq.process produced by 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 ma-
trix 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 Sil-
verman'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'.
Augment data with information from a(n) spatialreg 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 all columns used to fit the model are present.

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 'sarlm'
augment(x, data = x$X, ...)```
Arguments

- **x**: An object of 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::tibble()` with columns:

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

See Also

- `augment()`
- Other spatialreg tidiers: `glance.sarlm()`, `tidy.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(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)

augment.smooth.spline  
* Tidy a(n) smooth.spline 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 '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:** Missparsed 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()`, `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)) +
  geom_point() +
  geom_line(aes(y = .fitted))
```

**augment.speedlm**  
*Augment data with information from a(n) speedlm 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 all columns used to fit the model are present.

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 '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

```r
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 all columns used to fit the model are present.

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 '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 all columns used to fit the model are present.

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.survreg

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.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
library(survival)

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

tidy(sr)
augment(sr, ovarian)

# 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)
```

Description

Add fitted values, residuals, and other common outputs to the value returned from `augment`.

Usage

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

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

In the case that a residuals or influence generic is not implemented for the model, fail quietly.

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/hadley/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 to columns corresponding 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 .1
data.frame_tidiers

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

```r
## 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()`
**Examples**

```r
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

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

```r
## 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. **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:

- `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

```r
library(survival)

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

 tidy(afit)
```
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 '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

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

## S3 method for class 'Arima'

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

Value

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

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>Akaike's Information Criterion for the model.</td>
</tr>
<tr>
<td>BIC</td>
<td>Bayesian Information Criterion for the model.</td>
</tr>
<tr>
<td>df.null</td>
<td>Degrees of freedom used by the null model.</td>
</tr>
<tr>
<td>df.residual</td>
<td>Residual degrees of freedom.</td>
</tr>
<tr>
<td>logLik</td>
<td>The log-likelihood of the model.</td>
</tr>
<tr>
<td>nobs</td>
<td>Number of observations used.</td>
</tr>
<tr>
<td>pseudo.r.squared</td>
<td>Like the R squared statistic, for situations when the R squared statistic isn’t defined.</td>
</tr>
</tbody>
</table>

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.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.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
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

## S3 method for class 'biglm'

```r
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:

- `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
```
```r
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)
```

---

**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
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.
## glance.cch

### 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.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:

- `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

### 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
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.clm

**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.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.
- **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
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)

```

**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.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.

- **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
library(ordinal)

fit <- clmm(rating ~ temp + contact + (1 | judge), data = wine)

tidy(fit)
tidy(fit, conf.int = TRUE, conf.level = 0.9)
```
**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 '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.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.
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

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)

c1 <- clogit(case ~ tocc + tocc:education + strata(id), logan2)
tidy(c1)
glance(c1)

library(ggplot2)
```r
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**

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 'cv.glmnet'

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 exactly one row and columns:

- **lambda.1se**  
The value of the penalization parameter lambda that results in the sparsest model while remaining within one standard error of the minimum loss.

- **lambda.min**  
The value of the penalization parameter lambda that achieved minimum loss as estimated by cross validation.

- **nobs**  
Number of observations used.

See Also

- `glance()`, `glmnet::cv.glmnet()`  
Other glmnet tidiers: `glance.glmnet()`, `tidy.cv.glmnet()`, `tidy.glmnet()`

Examples

```r
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)
nobs <- glance(cvfit1)
tidy(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()

# plot of MSE as a function of lambda with confidence ribbon

g <- g + geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)

# plot of MSE as a function of lambda with confidence ribbon and choices

```

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$glmnet.fit)
ggplot(tidied, aes(lambda, estimate, group = term)) +
  scale_x_log10() +
  geom_line() +
  geom_vline(xintercept = glance_cv$lambda.min) +
  geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)

---

glance.drc  Glance at a(n) drc 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 '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

See Also
glance(), drc::drm()

Other drc tidiers: augment.drc(), tidy.drc()

Examples

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

Usage

## S3 method for class 'ergm'
glance(x, deviance = FALSE, mcmc = FALSE, ...)

Arguments

x An ergm object returned from a call to \texttt{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 \texttt{ergm::summary()}. \textbf{Cautionary note}: Mispecified arguments may be silently ignored.

Value

glance.ergm returns a one-row tibble with the columns

\begin{itemize}
  \item \texttt{independence} Whether the model assumed dyadic independence
  \item \texttt{iterations} The number of MCMLE iterations performed before convergence
  \item \texttt{logLik} If applicable, the log-likelihood associated with the model
  \item \texttt{AIC} The Akaike Information Criterion
  \item \texttt{BIC} The Bayesian Information Criterion
\end{itemize}

If \texttt{deviance = TRUE}, and if the model supports it, the tibble will also contain the columns

\begin{itemize}
  \item \texttt{null.deviance} The null deviance of the model
  \item \texttt{df.null} The degrees of freedom of the null deviance
  \item \texttt{residual.deviance} The residual deviance of the model
  \item \texttt{df.residual} The degrees of freedom of the residual deviance
\end{itemize}

See Also

\texttt{glance()}, \texttt{ergm::ergm()}, \texttt{ergm::summary.ergm()}

Other ergm tidiers: \texttt{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 succesful 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.
- `n.obs` 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()

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, 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, 2, 1, 5, 6, 4),
  v5 = c(1, 1, 1, 1, 1, 3, 3, 3, 3, 1, 1, 1, 1, 6, 4, 5),
  v6 = c(1, 1, 2, 1, 3, 3, 3, 4, 3, 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.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.
- `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
library(lfe)

N <- 1e2
DT <- data.frame(
  id = sample(5, N, TRUE),
  v1 = sample(5, N, TRUE),
  v2 = sample(1e6, N, TRUE),
)```
```r
v3 = sample(round(runif(100, max = 100), 4), N, TRUE),
v4 = sample(round(runif(100, max = 100), 4), N, TRUE)
)

result_felm <- felm(v2 ~ v3, DT)
tidy(result_felm)
augment(result_felm)

result_felm <- felm(v2 ~ v3 | id + v1, DT)
tidy(result_felm, fe = TRUE)
tidy(result_felm, robust = TRUE)
augment(result_felm)

v1 <- DT$v1
v2 <- DT$v2
v3 <- DT$v3
id <- DT$id
result_felm <- felm(v2 ~ v3 | id + v1)

tidy(result_felm)
augment(result_felm)
glance(result_felm)
```

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. **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.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.

See Also

- `tidy()`, `MASS::fitdistr()`
- Other `fitdistr` tidiers: `tidy.fitdistr()`

Examples

```r
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 `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 'fixest'
glance(x, ...)
```

Arguments

- `x` A fixest object returned from any of the fixest estimators
- `...` 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 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.
- `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.
- `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.
- `within.r.squared` R squared within fixed-effect groups.

Note

The columns of the result depend on the type of model estimated.
Examples

```r
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:
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)

## The other fixest methods all work similarly. For example:
gravity_pois <- feglm(Euros ~ log(dist_km) | Origin + Destination + Product + Year, trade)
tidy(gravity_pois)
glance(gravity_pois)
augment(gravity_pois, trade)
```

```r

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

## 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.

See Also

glance(), mgcv::gam()

Other mgcv tidiers: tidy.gam()

Examples

g <- mgcv::gam(mpg ~ s(hp) + am + qsec, data = mtcars)
tidy(g)
tidy(g, parametric = 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

```r
## S3 method for class 'garch'
glance(x, test = c("box-ljung-test", "jarque-bera-test"), ...)
```

Arguments

- `x`: A garch object returned by `tseries::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.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.
- **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(x, ...)
```

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.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:

- `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

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.glm 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
## S3 method for class 'glm'

argues

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.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.
- **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.svyglm(), tidy.glm(), tidy.lm.beta(), tidy.lm(), tidy.mlm()

**Examples**

```r
g <- glm(am ~ mpg, mtcars, family = "binomial")
glance(g)
```

---

**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 '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.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:

- `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
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)

---

### 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 '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.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:

- 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
library(robust)

gm <- glmRob(am ~ wt, data = mtcars, family = "binomial")

tidy(gm)
glance(gm)
```

---

`glance.gmm` *Glance at a(n) gmm 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 '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.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.
- `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
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)`
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 'ivreg'
glance(x, diagnostics = FALSE, ...)
```

Arguments

- **x**: An \texttt{ivreg} object created by a call to \texttt{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. \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.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 \texttt{data} argument.

Value

A \texttt{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
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 at an 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.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:

- `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)
```
```r
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
rmsea.conf.high
  95 percent upper bound on RMSEA
srmr  Standardised root mean residual
agfi  Adjusted goodness of fit
cfi  Comparative fit index
tli  Tucker Lewis index
AIC  Akaike information criterion
BIC  Bayesian information criterion
ngroups  Number of groups in model
nobs  Number of observations included
norig  Number of observation in the original dataset
nexcluded  Number of excluded observations
converged  Logical - Did the model converge
estimator  Estimator used
missing_method  Method for eliminating missing data


See Also

\texttt{glance()}, \texttt{lavaan::cfa()}, \texttt{lavaan::sem()}, \texttt{lavaan::fitmeasures()}

Other lavaan tidiers: \texttt{tidy.lavaan()}

Examples

```r
## Not run:
library(lavaan)
cfa.fit <- cfa(
  "F =~ x1 + x2 + x3 + x4 + x5",
  data = HolzingerSwinford1939, group = "school"
)
  glance(cfa.fit)
## End(Not run)
```
glance.lm

Glance at a(n) lm 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 '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()

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

Examples

library(ggplot2)
library(dplyr)
mod <- lm(mpg ~ wt + qsec, data = mtcars)
tidy(mod)
glance(mod)

# coefficient plot
d <- tidy(mod) %>%
  mutate(
    low = estimate - std.error,
    high = estimate + std.error
  )
ggplot(d, aes(estimate, term, xmin = low, xmax = high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0) +
  geom_errorbarh()

augment(mod)
augment(mod, mtcars)

# predict on new data
newdata <- mtcars %>%
  head(6) %>%
  mutate(wt = wt + 1)
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 at an lmodel2 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 '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.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:

- 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

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

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.

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.
- `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

robust::lmRob()

Other robust tidiers: augment.lmRob(), glance.glmRob(), tidy.glmRob(), tidy.lmRob()

Examples

library(robust)

m <- lmRob(mpg ~ wt, data = mtcars)

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

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 'lmrob'

Glance at a(n) lmrob object

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.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 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.Mclust

Glance at a(n) Mclust 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 '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.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:

- **BIC** Bayesian Information Criterion for the model.
- **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.
- **model** A string denoting the model type with optimal BIC
- **G** Number mixture components in optimal model
- **hypvol** If the other model contains a noise component, the value of the hypervolume parameter. Otherwise ‘NA’.
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.mfx

```r
## 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.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 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

## 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.mjoint**

**Glance at a(n) mjoint 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 'mjoint'
glance(x, ...)

---

---

---

---

---
Arguments

x
An mjoint object returned from joineRML::mjoint().

... 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.
logLik       The log-likelihood of the model. [stats::logLik()] may be a useful reference.
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

## 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.muhaz

#### Glance at an 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.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:

- `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
library(muhaz)

data(ovarian, package = "survival")
x <- muhaz::muhaz(ovarian$futime, ovarian$fustat)
tidy(x)
glance(x)
```

---

**glance.multinom**

*Glance at a(n) multinom 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 '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:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIC</td>
<td>Akaike's Information Criterion for the model.</td>
</tr>
<tr>
<td>deviance</td>
<td>Deviance of the model.</td>
</tr>
<tr>
<td>edf</td>
<td>The effective degrees of freedom.</td>
</tr>
<tr>
<td>nobs</td>
<td>Number of observations used.</td>
</tr>
</tbody>
</table>

See Also

- `glance()`, `nnet::multinom()`
- Other multinom tidiers: `tidy.multinom()`

Examples

```r
library(nnet)
library(MASS)

eexample(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

```r
fit.gear <- multinom(gear ~ mpg + factor(am), data = mtcars)
tidy(fit.gear)
glance(fit.gear)
```

---

**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 'nlrq'
glance(x, ...)
```

**Arguments**

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

- `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::nlrq()

Other quantreg tidiers: augment.nlrq(), augment.rqs(), augment.rq(), glance.rq(), tidy.nlrq(), tidy.rqs(), tidy.rq()

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.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.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

```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)
```

---

**glance.orcutt**  
*Glance at a(n) orcutt 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 '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.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.
- `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.
### glace.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()`

**See Also**

- `glance()`, `orcutt::cochrane.orcutt()`
  Other orcutt tidiers: `tidy.orcutt()`

**Examples**

```r
library(orcutt)

reg <- lm(mpg ~ wt + qsec + disp, mtcars)
tidy(reg)

co <- cochrane.orcutt(reg)
tidy(co)
glance(co)
```

```r
# S3 method for class 'pam'
glance(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.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:

```r
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)
```
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` object 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

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)

---

_glance.poLCA_  

Glance at a(n) poLCA 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 'poLCA'
glance(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 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
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)

glance.polr

### Glance at a(n) polr 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 '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.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.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.MASS::polr()`
- Other ordinal tidiers: `augment.clm()`, `augment.polr()`, `glance.clmm()`, `glance.clm()`, `glance.svyolr()`, `tidy.clmm()`, `tidy.clm()`, `tidy.polr()`, `tidy.svyolr()`

Examples

```r
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)
```
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 'pyyears'
glance(x, ...)
```

Arguments

- `x` A `pyyears` object returned from `survival::pyyears()`.
- `...` 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.
- `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
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 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. **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 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	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)

---

**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 'rlm'
glance(x, ...)
```

**Arguments**

- `x`: An `rlm` object returned by `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.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.
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
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.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**

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()`

Usage

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

glance(x, ...)
```

Arguments

- **x**  
  An object of 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.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:

- **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

```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(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)
```
Tidy a smooth.spline 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 'smooth.spline'

glance(x, ...)
```

**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.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:

- `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

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))

---

glance.speedglm Glance at a(n) speedglm 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 'speedglm'
glance(x, ...)
\end{verbatim}

Arguments

\begin{itemize}
\item \textit{x} A speedglm object returned from \texttt{speedglm::speedglm}().
\item \textit{...} 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.lvl = 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.
\end{itemize}
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. \([\text{stats::logLik()}]\) 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
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 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.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.
- `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.
- `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

`speedglm::speedlm()`

Other speedlm tidiers: `augment.speedlm()`, `glance.speedglm()`, `tidy.speedglm()`, `tidy.speedlm()`
Examples

```r
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

```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.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:

- `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

```
library(survival)

s <- survdiff(
  Surv(time, status) ~ pat.karno + strata(inst),
  data = lung
)

tidy(s)

glance(s)
```

```n
---

### 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'

## No examples provided for glance.survexp.
```
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
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)
```
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. 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:

- `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.
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.survep()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survep()`, `tidy.survreg()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
library(survival)
cfit <- coxph(Surv(time, status) ~ age + sex, lung)
sfit <- survfit(cfit)
tidy(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 \textit{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 \textit{NA} of the appropriate type.

\textbf{Usage}

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

\texttt{glance(x, ...)}
```

\textbf{Arguments}

\begin{itemize}
  \item \texttt{x} An \texttt{survreg} object returned from \texttt{survival::survreg()}.
  \item \texttt{...} 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.
\end{itemize}

\textbf{Value}

A \texttt{tibble::tibble()} with exactly one row and columns:

\begin{itemize}
  \item \texttt{AIC} Akaike's Information Criterion for the model.
  \item \texttt{BIC} Bayesian Information Criterion for the model.
  \item \texttt{df} Degrees of freedom used by the model.
  \item \texttt{df.residual} Residual degrees of freedom.
  \item \texttt{iter} Iterations of algorithm/fitting procedure completed.
  \item \texttt{logLik} The log-likelihood of the model. \texttt{[stats::logLik()]} may be a useful reference.
  \item \texttt{nobs} Number of observations used.
  \item \texttt{p.value} P-value corresponding to the test statistic.
  \item \texttt{statistic} Chi-squared statistic.
\end{itemize}

\textbf{See Also}

\texttt{glance()}, \texttt{survival::survreg()}

Other survreg tidiers: \texttt{augment.survreg()}, \texttt{tidy.survreg()}

Other survival tidiers: \texttt{augment.coxph()}, \texttt{augment.survreg()}, \texttt{glance.aareg()}, \texttt{glance.cch()}, \texttt{glance.coxph()}, \texttt{glance.pyears()}, \texttt{glance.survdiff()}, \texttt{glance.survexp()}, \texttt{glance.survfit()}, \texttt{tidy.aareg()}, \texttt{tidy.cch()}, \texttt{tidy.coxph()}, \texttt{tidy.pyears()}, \texttt{tidy.survdiff()}, \texttt{tidy.survexp()}, \texttt{tidy.survfit()}, \texttt{tidy.survreg()}
Examples

```r
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)
```

### 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.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.
- `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()`, `tidy.glm()`, `tidy.lm.beta()`, `tidy.lm()`, `tidy.mlm()`

Examples

```R
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)

---

**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 'svyolr'
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
library(MASS)

fit <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)

tidy(fit, exponentiate = TRUE, conf.int = TRUE)
glance(fit)
```

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

glance_optim(x, ...)
```
Arguments

x  
A list returned from \texttt{stats::optim}.

...  
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:

\begin{itemize}
  \item convergence: Convergence code.
  \item function.count: Number of calls to ‘fn’.
  \item gradient.count: Number of calls to ‘gr’.
  \item value: Minimized or maximized output value.
\end{itemize}

See Also

\texttt{glance(), stats::optim()}

Other list tidiers: \texttt{list_tidiers, tidy_irlba(), tidy_optim(), tidy_svd(), tidy_xyz()}

Examples

\begin{verbatim}
go <- optim(c(1, 1, 1), f)
\end{verbatim}
null_tidiers

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

```r
## S3 method for class `"NULL"
 tidy(x, ...)
```

```r
## S3 method for class `"NULL"
 glance(x, ...)
```

```r
## 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`
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

```r
## 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 a(n) SpatialPolygonsDataFrame 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.

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 Linestring object.
- `region` name of variable used to split up regions
- `...` not used by this method
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 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()`

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 a(n) 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`. 
tidy.acf

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.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
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))
```

### 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 objects, such as those created by \texttt{stats::anova()} or \texttt{car::Anova()}.

... 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 \texttt{data} argument.

Details

The term column of an ANOVA table can come with leading or trailing whitespace, which this tidying method trims.

Value

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

- \texttt{df} Degrees of freedom used by this term in the model.
- \texttt{meansq} Mean sum of squares. Equal to total sum of squares divided by degrees of freedom.
- \texttt{p.value} The two-sided p-value associated with the observed statistic.
- \texttt{statistic} The value of a T-statistic to use in a hypothesis that the regression term is non-zero.
- \texttt{sumsq} Sum of squares explained by this term.
- \texttt{term} The name of the regression term.

See Also

\texttt{tidy()}, \texttt{stats::anova()}, \texttt{car::Anova()}

Other anova tidiers: \texttt{glance.aov()}, \texttt{tidy.TukeyHSD()}, \texttt{tidy.aovlist()}, \texttt{tidy.aov()}, \texttt{tidy.manova()}

Examples

```r
a <- lm(mpg ~ wt + qsec + disp, mtcars)
b <- lm(mpg ~ wt + qsec, mtcars)
tidy(anova(a, b))
```
tidy.aov  Tidy a(n) aov 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 '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.

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)
```
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.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 `term` column of an ANOVA table can come with leading or trailing whitespace, which this tidying method trims.

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)
```

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.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.
- `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.
tidy.betamfx

... 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("mfxarticle") from the mfx package for more details.

Value

A tibble::tibble() with columns:

<table>
<thead>
<tr>
<th>conf.high</th>
<th>Upper bound on the confidence interval for the estimate.</th>
</tr>
</thead>
<tbody>
<tr>
<td>conf.low</td>
<td>Lower bound on the confidence interval for the estimate.</td>
</tr>
<tr>
<td>estimate</td>
<td>The estimated value of the regression term.</td>
</tr>
<tr>
<td>p.value</td>
<td>The two-sided p-value associated with the observed statistic.</td>
</tr>
<tr>
<td>statistic</td>
<td>The value of a T-statistic to use in a hypothesis that the regression term is non-zero.</td>
</tr>
<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>
<tr>
<td>atmean</td>
<td>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.</td>
</tr>
</tbody>
</table>

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)
```r
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)
```

---

### 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.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 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
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 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.
- `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.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.

**Usage**

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

**See Also**

`tidy()`, `biglm::biglm()`, `biglm::bigglm()`

Other `biglm` tidiers: `glance.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)

# 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)
```
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
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
library(binGroup)
l 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
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)
```

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.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:

- `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()

Examples

```r
library(btergm)
set.seed(1)

# Create 10 random networks with 10 actors
networks <- list()
for (i in 1:10) {
  mat <- matrix(rbinom(100, 1, .25), nrow = 10, ncol = 10)
  diag(mat) <- 0
  nw <- network::network(mat)
  networks[[i]] <- nw
}

# Create 10 matrices as covariates
covariates <- list()
for (i in 1:10) {
  ```
mat <- matrix(rnorm(100), nrow = 10, ncol = 10)
covariates[[i]] <- mat
}

# Fit a model where the propensity to form ties depends
# on the edge covariates, controlling for the number of
# in-stars
btfit <- btergm(networks ~ edges + istar(2) + edgecov(covariates), R = 100)

# Show terms, coefficient estimates and errors
tidy(btfit)

# Show coefficients as odds ratios with a 99% CI
tidy(btfit, exponentiate = TRUE, conf.level = 0.99)

---

tidy.cch  

Tidy a(n) cch 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 '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.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:

- `conf.high`  
  Upper bound on the confidence interval for the estimate.
- `conf.low`  
  Lower bound on the confidence interval for the estimate.
tidy.cch

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.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit(), tidy.survreg()*

Examples

```r
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 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.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.
- `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
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))
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.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

## 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 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

library(ordinal)

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

tidy(fit)
tidy(clmm)

```r
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 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
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)
```
tidy.coeftest

Tidy a(n) coeftest 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 '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.
Tidy a `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.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.
- `contrast` Levels being compared.
- `estimate` The estimated value of the regression term.

**Examples**

```r
tidy::lmtest::coeftest()
library(lmtest)
data(Mandible)
fm <- lm(length ~ age, data = Mandible, subset = (age <= 28))

lmtest::coeftest(fm)
tidy(coeftest(fm))
```
### 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, ...)```

See Also

- `tidy()`, `multcomp:::confint.glht()`, `multcomp::glht()`
- Other multcomp tidiers: `tidy.cld()`, `tidy.glht()`, `tidy.summary.glht()`

#### Examples

```r
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))
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)
```
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.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.  
- `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
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)

---

tidy.coxph  
Tidy a(n) coxph 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 '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.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.
- `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
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 == tocc)

c1 <- clogit(case ~ tocc + tocc:education + strata(id), logan2)
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()

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

Arguments
x A cv.glmnet object returned from \texttt{glmnet::cv.glmnet}().

... 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 conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = \texttt{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} with columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>lambda</td>
<td>Value of penalty parameter lambda.</td>
</tr>
<tr>
<td>nzero</td>
<td>Number of non-zero coefficients for the given lambda.</td>
</tr>
<tr>
<td>std.error</td>
<td>The standard error of the regression term.</td>
</tr>
<tr>
<td>conf.low</td>
<td>lower bound on confidence interval for cross-validation estimated loss.</td>
</tr>
<tr>
<td>conf.high</td>
<td>upper bound on confidence interval for cross-validation estimated loss.</td>
</tr>
<tr>
<td>estimate</td>
<td>Median loss across all cross-validation folds for a given lambda</td>
</tr>
</tbody>
</table>
See Also

\texttt{tidy()}, \texttt{glmnet::cv.glmnet()}

Other \texttt{glmnet} tidiers: \texttt{glance.cv.glmnet()}, \texttt{glance.glmnet()}, \texttt{tidy.glmnet()}

Examples

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)
tidy.density

```r
ggplot(tidied, aes(lambda, estimate, group = term)) + scale_x_log10() + geom_line() + geom_vline(xintercept = glance_cv$lambda.min) + geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
```

---

**tidy.density** *(Deprecated) Tidy density objects*

**Description**

(Deprecated) Tidy density objects

**Usage**

```r
## 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

```r
## 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
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)
```

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
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))
```

**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.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 a column for each of the measures of association or tests contained in `massoc` 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
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")
```

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.
- `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 to pass to `ergm::summary()`. **Cautionary note:** Mispecified arguments may be silently ignored.
Value

A `tibble::tibble` with one row for each coefficient in the exponential random graph model, with columns:

- `term` The term in the model being estimated and tested
- `estimate` The estimated coefficient
- `std.error` The standard error
- `mcmc.error` The MCMC error
- `p.value` The two-sided p-value

References


See Also

- `tidy()`, `ergm::ergm()`, `ergm::control.ergm()`, `ergm::summary()`
- Other `ergm` tidiers: `glance.ergm()`

Examples

```r
library(ergm)
# Using the same example as the `ergm` package
# Load the Florentine marriage network data
data(florentine)

gest <- ergm(flomarriage ~ edges + absdiff("wealth"))

# Show terms, coefficient estimates and errors
tidy(gest)

tidy(gest, exponentiate = TRUE, conf.int = TRUE, conf.level = 0.99)

# Take a look at likelihood measures and other
# control parameters used during MCMC estimation
glance(gest)

# Other `ergm` tidiers

# Show coefficients as odds ratios with a 99% CI
```

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()`

Examples

```r
c <- set.seed(123)

# data
ml <- 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, 4, 6, 5),
v3 = c(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, 1, 2, 1, 1, 5, 6, 4),
v5 = c(1, 1, 1, 1, 3, 3, 3, 3, 1, 1, 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, robust = FALSE, ...)
```

**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.
Logical indicating whether or not to include estimates of fixed effects. Defaults to FALSE.

robust Logical indicating robust or clustered standard errors should be used. See lfe::summary.felm for details. 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.

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().lfe::felm()

Other felm tidiers: augment.felm()

Examples

library(lfe)
N <- 1e2
DT <- data.frame(
  id = sample(5, N, TRUE),
  v1 = sample(5, N, TRUE),
  v2 = sample(1e6, N, TRUE),
  v3 = sample(round(runif(100, max = 100), 4), N, TRUE),
  v4 = sample(round(runif(100, max = 100), 4), N, TRUE)
)
result_felm <- felm(v2 ~ v3, DT)
tidy(result_felm)
augment(result_felm)
```r
result_felm <- felm(v2 ~ v3 | id + v1, DT)
tidy(result_felm, fe = TRUE)
tidy(result_felm, robust = TRUE)
augment(result_felm)

v1 <- DT$v1
v2 <- DT$v2
v3 <- DT$v3
id <- DT$id
result_felm <- felm(v2 ~ v3 | id + v1)
tidy(result_felm)
augment(result_felm)
glance(result_felm)
```

**tidy.fitdistr**  
*Tidy a(n) 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, ...)
```

**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 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

\textit{tidy()}, \texttt{MASS::fitdistr()}

Other fitdistr tidiers: \texttt{glance.fitdistr()}

Examples

```r
set.seed(2015)
x <- rnorm(100, 5, 2)

library(MASS)
fit <- fitdistr(x, dnorm, list(mean = 3, sd = 1))
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 'fixest'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

**Arguments**

- `x` A \texttt{fixest} object returned from any of the \texttt{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 \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 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}. 

\textit{tidy.fixest} \hspace{1cm} Tidy a(n) fixest object
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 and 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
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:
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)
```
```r
# Approach (1) is preferred.
## The other fixest methods all work similarly. For example:
gravity_pois <- feglm(Euros ~ log(dist_km) | Origin + Destination + Product + Year, trade)
tidy(gravity_pois)
glance(gravity_pois)
augment(gravity_pois, trade)
```

---

**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 = 0.9`. 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()`
tidy.gam

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 'gam'
tidy(x, parametric = FALSE, conf.int = FALSE, conf.level = 0.95, ...)
```

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.
- `...` 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 `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.
tidy.gamlss

Tidy a(n) gamlss 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 'gamlss'
tidy(x, ...)

Arguments

x           A gamlss object returned from gamlss::gamlss().

...         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
tidy(), mgcv::gam()
Other mgcv tidiers: glance.gam()

Examples

g <- mgcv::gam(mpg ~ s(hp) + am + qsec, data = mtcars)
tidy(g)
tidy(g, parametric = TRUE)

References

tidy.gamlss
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.
- **parameter**: Type of coefficient being estimated: ‘mu’, ‘sigma’, ‘nu’, or ‘tau’.

Examples

```r
library(gamlss)

g <- gamlss(
  y ~ pb(x),
  sigma.fo = ~ pb(x),
  family = BCT,
  data = abdom,
  method = mixed(1, 20)
)
tidy(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

```r
## S3 method for class 'garch'
tidy(x, ...)
```
Arguments

x: A garch object returned by `tseries::garch()`. 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.
- **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
library(tseries)
data(EuStockMarkets)
dax <- diff(log(EuStockMarkets))[, "DAX"]
dax.garch <- garch(dax)
dax.garch

 tidy(dax.garch)
 glance(dax.garch)
```

**tidy.geeglm**

**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.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 `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:

- **regression**: TRUE

See Also

- `tidy()`, `geepack::geeglm()`

Examples

```r
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)

tidy.glht

## 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.
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

## S3 method for class 'glm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)

See Also
tidy(), multcomp::glht()

Other multcomp tidiers: tidy.cld(), tidy.confint.glht(), tidy.summary.glht()
tidy.glmnet

Tidy a(n) glmnet object

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.svyglm(), tidy.lm.beta(), tidy.lm(), tidy.mlm()

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 zero should be included in the results. Defaults to FALSE.
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().glmnet(), tidy.cv.glmnet()`

Examples

```r
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)")
```
```r
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.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.

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

library(robust)

gm <- glmRob(am ~ wt, data = mtcars, family = "binomial")
tidy(gm)
glance(gm)

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

## 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.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:

- `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

```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.gmm  
*Tidy a(n) gmm 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.gmm**

Usage

```r
## 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 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()`
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)
tidy.htest

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)

tidy.htest

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

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

Arguments
x An htest objected, 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.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.
### tidy.ivreg

**Tidy a(n) ivreg 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 'ivreg'
tidy(x, conf.int = FALSE, conf.level = 0.95, instruments = FALSE, ...) # S3 method for class 'ivreg'
tidy(x, conf.int = FALSE, conf.level = 0.95, instruments = FALSE, ...)```

**Arguments**

- `x`: An `ivreg` object.
- `conf.int`: Logical. Should confidence intervals be returned? Default is `FALSE`.
- `conf.level`: The confidence level to use. Default is 0.95 (95%).
- `instruments`: Logical. Should instruments be computed? Default is `FALSE`.

**Value**

A `tibble::tibble` (or other S3 object), depending on the number of components being summarized.

**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)
glance(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)

c <- cor.test(mtcars$wt, mtcars$mpg)
tidy(c)

c <- chisq.test(xtabs(Freq ~ Sex + Class, data = as.data.frame(Titanic)))
tidy(c)
augment(c)
```
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.

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

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

## 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
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.kde

_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

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() +
tidy.Kendall

```r
theme_void()

# also works with 3 dimensions
dat3 <- replicate(3, rnorm(100))
k3 <- kde(dat3)

td3 <- tidy(k3)
td3
```

---

**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

library(Kendall)

A <- c(2.5, 2.5, 2.5, 2.5, 6.5, 6.5, 10, 10, 10, 10, 14, 14, 14, 16, 17)
B <- c(1, 1, 1, 1, 2, 1, 1, 2, 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.kmeans

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

## S3 method for class 'kmeans'
tidy(x, col.names = colnames(x$centers), ...)

Arguments

x

da 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.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.lavaan

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**: Standard error
- **statistic**: The z value returned by `lavaan::parameterEstimates()`
- **p.value**: p-value
- **conf.low**: Lower confidence limit
- **conf.high**: Upper confidence limit
- **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
## 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)
```

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.svyglm(), tidy.glm(), tidy.lm.beta(), tidy.mlm()

Examples

```r
library(ggplot2)
library(dplyr)

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

 tidy(mod)
 glance(mod)

# coefficient plot
d <- tidy(mod) %>%
  mutate(
    low = estimate - std.error,
    high = estimate + std.error
  )

  ggplot(d, aes(estimate, term, xmin = low, xmax = high, height = 0)) +
  geom_point() +
  geom_vline(xintercept = 0) +
  geom_errorbarh()

 augment(mod)
 augment(mod, mtcars)

# predict on new data
newdata <- mtcars %>%
  head(6) %>%
  mutate(wt = wt + 1)
 augment(mod, newdata = newdata)
```
```r
tidy.lm.beta

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)
```

---

**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**

```r
## 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.
- `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`.

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.svyglm()`, `tidy.glm()`, `tidy.lm()`, `tidy.mlm()`

Examples

```r
library(lm.beta)

mod <- stats::lm(speed ~ ., data = cars)
std <- lm.beta(mod)
tidy(std, conf.int = TRUE)

ctl <- c(4.17, 5.58, 5.18, 6.11, 4.50, 4.61, 5.17, 4.53, 5.33, 5.14)
trt <- c(4.81, 4.17, 4.41, 3.59, 5.87, 3.83, 6.03, 4.89, 4.32, 4.69)
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)
```
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 '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 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.
tidy.lmRob

See Also
tidy(), lmodel2::lmodel2()
Other lmodel2 tidiers: glance.lmodel2()

Examples

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

## 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.
tidy.lmrob

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

library(robust)
m <- lmRob(mpg ~ wt, data = mtcars)
tidy(m)
augment(m)
glance(m)

Tidy a(n) lmrob 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 '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

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

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_rgl1 <- ref_grid(oranges_lm1)
tidy.manova

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

## 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

Object An object of class "manova" or an aov object with multiple responses.

- **intercept** logical. If TRUE, the intercept term is included in the table.
- **tol** tolerance to be used in deciding if the residuals are rank-deficient: see `qr`.

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.

See Also

- `tidy()`, `stats::summary.manova()`

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

Examples

```r
npk2 <- within(npk, foo <- rnorm(24))
m <- manova(cbind(yield, foo) ~ block + N * P * K, npk2)
tidy(m)
```
tidy.map

Tidy a(n) map 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 'map'
tidy(x, ...)
```

Arguments

- `x` A map object returned from `maps::map()`.
- `...` 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:

- `term` The name of the regression term.
- `long` Longitude.
- `lat` 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

```r
library(maps)
library(ggplot2)
```
tidy.Mclust

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 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

## 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.
tidy.mediate

See Also

tidy(), mclust::Mclust()

Other mclust tidiers: augment.Mclust()

Examples

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)

#### 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

```r
## 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 group, 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
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.mfx

Tidy a(n) mfx 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 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

```r
## 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 betamfx 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.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("mfxarticle") 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()`, `mfx::logitmfx()`, `mfx::negbinmfx()`, `mfx::poissonmfx()`, `mfx::probitmfx()`

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

Examples

```r
## 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)

 tidy.mjoint  

 **Tidy a(n) mjoint 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 '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.
tidy.mjoint

Optional a bootSE object from joineRML::bootSE(). If specified, calculates
certainty intervals via the bootstrap. Defaults to NULL, in which case standard
eors 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.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(), joineRML::mjoint(), joineRML::bootSE()

Other mjoint tidiers: glance.mjoint()

Examples

## 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,
```r
data = hvd,
init = 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

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 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

```r
## 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.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

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.
tidy.muhaz

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.svyglm(), tidy.glm(), tidy.lm.beta(), tidy.lm()

Examples

mod <- lm(cbind(mpg, disp) ~ wt, mtcars)
tidy(mod, conf.int = TRUE)

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

Arguments

x  A muhaz object returned by muhaz::muhaz().
...

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.

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

library(muhaz)

data(ovarian, package = "survival")
x <- muhaz::muhaz(ovarian$futime, ovarian$fustat)
tidy(x)
glance(x)

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 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.
tidy.nlrq

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
library(nnet)
library(MASS)

example(birthwt)
bwt.mu <- multinom(low ~ ., bwt)
tidy(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)

```

display

---

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.

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

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)

## tidier.numeric

Tidy atomic vectors

Description

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

Usage

## 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)
Tidy 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.orcutt  
_Tidy a(n) orcutt object_

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.orcut()`. 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.
- **term**: The name of the regression term.

See Also

- `orcutt::cochrane.orcutt()`

Other orcutt tidiers: `glance.orcutt()`

Examples

```r
library(orcutt)

reg <- lm(mpg ~ wt + qsec + disp, mtcars)
tidy(reg)

co <- cochrane.orcutt(reg)
co
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.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 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

```
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.wilcox.test(compounds, class))
```
### tidy.pam

*Tidy a(n) pam 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 '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.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 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.
tidy.plm

Tidy a(n) plm 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 '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.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(), plm::plm(), tidy.lm()

Other plm tidiers: augment.plm(), glance.plm()

Examples

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)
Tidy a(n) poLCA 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 '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
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.polr**

Tidy a(n) polr 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 '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 [https://r.789695.n4.nabble.com/p-values-of-plor-td4668100.html](https://r.789695.n4.nabble.com/p-values-of-plor-td4668100.html). This approach is computationally intensive, so that p.values are only returned if requested explicitly. Ad-
ditionally, 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
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(fit2, 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.
tidy.power.htest

Usage

## 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.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:

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

ptt <- power.t.test(n = 2:30, delta = 1)
tidy(ptt)

library(ggplot2)

ggplot(tidy(ptt), 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

## 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.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 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:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</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:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</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</td>
</tr>
<tr>
<td>cumulative</td>
<td>Cumulative fraction of variation explained by principle components up to this component.</td>
</tr>
</tbody>
</table>

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)

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.pyears

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.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**

- `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.
- **pyyears**
  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
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.rcorr**

*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.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

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

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
tidy.ref.grid

```r
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

```
## S3 method for class 'ref.grid'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

### 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
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")
  ))
```
tidy.regsubsets

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.regsubsets

Tidy an 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, ...)

Arguments

x  
A regsubsets object created by \texttt{leaps::regsubsets()}.  

...  
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.\texttt{level} = 0.9}, all computation will proceed using \texttt{conf.\texttt{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 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
all_fits <- leaps::regsubsets(hp ~ ., mtcars)
tidy(all_fits)
```

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, ...)
```

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
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

## S3 method for class 'rlm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

x An rlm object returned by MASS::rlm().

See Also

tidy(), MASS::lm.ridge()

Other ridgelm tidiers: glance.ridgelm()
### 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.

- `...`: 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()`

---

```r
tidy.rma
tidy.rma
```
tidy.rma

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.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 individual study

type  The estimate type (summary vs individual study)

Examples

library(metafor)

df <- escalc(
    measure = "RR",
    a1 = tpos,
    b1 = tneg,
    c1 = cpos,
    d1 = cneg,
    data = dat.bcg
)
tidy.roc

```r
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.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:

- `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
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(tidyr)
rocs <- 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 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

Tidy a(n) rqs 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 '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.
tidy.sarlm

Tidying methods for spatially autoregressive models

Description

These methods tidy the coefficients of spatial autoregression models generated by functions of the spatialreg package.

Usage

## S3 method for class 'sarlm'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)

Arguments

- **x**
  An object of 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. **Cautio-**

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-

- **std.error**
  The standard error of the regression term.

- **term**
  The name of the regression term.
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

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

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(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)
```
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

## S3 method for class 'speedglm'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)
tidy.speedglm

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.
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

library(speedglm)

clotting <- data.frame(
u = c(5, 10, 15, 20, 30, 40, 60, 80, 100),
lot1 = c(118, 58, 42, 35, 27, 21, 19, 18)
)

```r
fit <- speedglm(lot1 ~ log(u), data = clotting, family = Gamma(log))
tidy(fit)
glance(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

```r
## 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.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::speedlm(), tidy.lm()
```

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

Examples

```r
mod <- speedglm::speedlm(mpg ~ wt + qsec, data = mtcars, fitted = TRUE)
tidy(mod)
 glance(mod)
augment(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

```
## 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.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.
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

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))
  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_emm  Tidy a(n) summary_emm 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_emm'
tidy(x, null.value = NULL, ...)
```

Arguments

- `x`: A `summary_emm` object.
- `null.value`: Value to which estimate is compared.
- `...`: 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.
- `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
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 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`  
  A `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 `survdiff()` tidiers:  
  - `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.surveexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.surveexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```r
library(survival)

s <- survdiff(
  Surv(time, status) ~ pat.karno + strata(inst),
  data = lung
)

tidy(s)
glance(s)
```
**tidy.survexp**

**Tidy a(n) survexp 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 '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

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)

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 '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.
**tidy.survfit**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conf.low</td>
<td>Lower bound on the confidence interval for the estimate.</td>
</tr>
<tr>
<td>n.censor</td>
<td>Number of censored events.</td>
</tr>
<tr>
<td>n.event</td>
<td>Number of events at time $t$.</td>
</tr>
<tr>
<td>n.risk</td>
<td>Number of individuals at risk at time zero.</td>
</tr>
<tr>
<td>std.error</td>
<td>The standard error of the regression term.</td>
</tr>
<tr>
<td>time</td>
<td>Point in time.</td>
</tr>
<tr>
<td>estimate</td>
<td>estimate of survival or cumulative incidence rate when multistate state</td>
</tr>
<tr>
<td>state</td>
<td>state if multistate survfit object input</td>
</tr>
<tr>
<td>strata</td>
<td>strata if stratified survfit object input</td>
</tr>
</tbody>
</table>

**See Also**

`tidy()`, `survival::survfit()`

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

**Examples**

```r
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

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.
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'
tidy(x, conf.int = FALSE, conf.level = 0.95, exponentiate = FALSE, ...)

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.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit()
tidy.svyolr

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()`

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 'svyolr'
tidy(
x,
conf.int = FALSE,
conf.level = 0.95,
exponentiate = FALSE,
p.values = FALSE,
...
)
```
**Arguments**

- **x**: A `svyolr` object returned from `survey::svyolr()`.
- **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 [https://r.789695.n4.nabble.com/p-values-of-plor-td4668100.html](https://r.789695.n4.nabble.com/p-values-of-plor-td4668100.html). 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.
Tidy a(n) systemfit 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 '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.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.

Examples

library(MASS)

fit <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)

tidy(fit, exponentiate = TRUE, conf.int = TRUE)

glance(fit)
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
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.
tidy.ts

Usage

## 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.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

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.ts 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

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

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.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

- 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.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:

- `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))
```
Tidy a(n) zoo 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 '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.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:

- `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()`.

Examples

```r
library(zoo)
library(ggplot2)
set.seed(1071)
```
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)

ggplot(tidy(Zrolled), aes(index, value, color = series)) +
  geom_line()

---

tidy_irlba

Tidy a(n) irlba 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.

Usage

 tidy_irlba(x, ...)

Arguments

x

A list returned from irlba::irlba().

... 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**

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 principal 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.
- `cumulative` Cumulative fraction of variation explained by principle components up to this component.

**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
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_optim

```r
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 a(n) 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()
tidy_svd

Arguments

- **x**: A list with components `u`, `d`, `v` returned by `base::svd()`.  
  - `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.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 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 principal 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
- **cumulative**: Cumulative fraction of variation explained by principle components up to this component.
See Also

base::svd()

Other svd tidiers: augment.prcomp(), tidy.prcomp(), tidy_irlba()

Other list tidiers: glance_optim(), list_tidders, tidy_irlba(), tidy_optim(), tidy_xyz()

Examples

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")

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!).

tidy_xyz Tidy a(n) xyz object masquerading as list
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**

```r
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, 93
  tidy.Arima, 194
* aareg tidiers
  glance.aareg, 90
  tidy.aareg, 188
* anova tidiers
  glance.aov, 92
  tidy.aov, 190
  tidy.aov, 192
  tidy.aovlist, 193
  tidy.manova, 270
  tidy.TukeyHSD, 336
* betareg tidiers
  tidy.betareg, 197
* biglm tidiers
  glance.biglm, 97
  tidy.biglm, 199
* bingroup tidiers
  glance.binDesign, 99
  tidy.binDesign, 200
  tidy.binWidth, 201
* car tidiers
  durbinWatsonTest_tidiers, 88
* cch tidiers
  glance.cch, 100
  glance.survfit, 176
  tidy.cch, 206
* coxph tidiers
  augment.coxph, 16
  glance.coxph, 105
  tidy.coxph, 217
* decompose tidiers
  augment.decomposed.ts, 19
  augment.stl, 80
* deprecated
  bootstrap, 84
  confint_tidy, 85
  data.frame_tidiers, 86
  finish_glance, 89
  fix_data_frame, 90
  summary_frame, 187
  tidy.density, 221
  tidy.dist, 222
  tidy.ftable, 236
  tidy.numeric, 288
* drc tidiers
  augment.drc, 21
  glance.drc, 109
  tidy.drc, 223
* emmeans tidiers
  tidy.emmGrid, 224
  tidy.lsmobj, 268
  tidy.ref.grid, 305
  tidy.summary_emm, 322
* epiR tidiers
  tidy.epi.2by2, 226
* ergm tidiers
  glance.ergm, 110
  tidy.ergm, 228
* factanal tidiers
  augment.factanal, 23
  glance.factanal, 112
  tidy.factanal, 230
* felm tidiers
  augment.felm, 25
  tidy.felm, 231
* fitdistr tidiers
  glance.fitdistr, 115
  tidy.fitdistr, 233
* fixest tidiers
  augment.fixest, 27
  tidy.fixest, 234
* garch tidiers
  glance.garch, 119
  tidy.garch, 239
* geepack tidiers
  glance.geeglm, 121
* glmnet tidiers
  glance.cv.glmnet, 107
  glance.glmnet, 123
  tidy.cv.glmnet, 219
  tidy.glmnet, 244
* gmm tidiers
  glance.gmm, 126
  tidy.gmm, 248
* htest tidiers
  augment.htest, 34
  tidy.htest, 251
  tidy.pairwise.htest, 290
  tidy.power.htest, 298
* ivreg tidiers
  augment.ivreg, 36
  glance.ivreg, 128
  tidy.ivreg, 252
* kmeans tidiers
  augment.kmeans, 38
  glance.kmeans, 130
  tidy.kmeans, 258
* lavaan tidiers
  glance.lavaan, 132
  tidy.lavaan, 259
* list tidiers
  glance_optim, 182
  list_tidiers, 183
  tidy_irlba, 339
  tidy_optim, 341
  tidy_svd, 342
  tidy_xyz, 344
* lm tidiers
  augment.glm, 29
  augment.lm, 39
  glance.glm, 122
  glance.lm, 134
  glance.svyglm, 179
  tidy.glm, 243
  tidy.lm, 261
  tidy.lm.beta, 263
  tidy.lm, 282
* lmodel2 tidiers
  glance.lmodel2, 136
  tidy.lmodel2, 265
* mclust tidiers
  augment.Mclust, 48
  tidy.Mclust, 273
* mediate tidiers
  tidy.mediate, 274
* mfx tidiers
  augment.betamfx, 10
  augment.mfx, 50
  glance.betamfx, 94
  glance.mfx, 142
  tidy.betamfx, 195
  tidy.mfx, 276
* mgcv tidiers
  glance.gam, 118
  tidy.gam, 237
* mjoint tidiers
  glance.mjoint, 144
  tidy.mjoint, 278
* muhaz tidiers
  glance.muhaz, 146
  tidy.muhaz, 283
* multcomp tidiers
  tidy.cld, 208
  tidy.confint.glht, 214
  tidy.glht, 242
  tidy.summary.glht, 321
* multinom tidiers
  glance.multinom, 147
  tidy.multinom, 284
* nls tidiers
  augment.nls, 57
  glance.nls, 150
  tidy.nls, 287
* orcutt tidiers
  glance.orcutt, 151
  tidy.orcutt, 289
* ordinal tidiers
  augment.clm, 14
  augment.polr, 64
  glance.clm, 102
  glance.clmm, 103
  glance.polr, 158
  glance.svyolr, 181
  tidy.clm, 209
  tidy.clmm, 211
  tidy.polr, 296
  tidy.svyolr, 331
* pam tidiers
  augment.pam, 58
  glance.pam, 153
  tidy.pam, 292
* plm tidiers
augment.plm, 60
  glance.plm, 155
  tidy.plm, 293
* poLCA tidiers
  augment.poLCA, 62
  glance.poLCA, 156
  tidy.poLCA, 295
* pyears tidiers
  glance.pyears, 160
  tidy.pyears, 302
* quantreg tidiers
  augment.nlrq, 55
  augment.rq, 71
  augment.rqs, 73
  glance.nlrq, 149
  glance.rq, 166
  tidy.nlrq, 285
  tidy.rq, 313
  tidy.rqs, 315
* ridgelm tidiers
  glance.ridgelm, 161
  tidy.ridgelm, 308
* rlm tidiers
  augment.rlm, 68
  glance.rlm, 163
  tidy.rlm, 309
* robust tidiers
  augment.lmRob, 42
  glance.lmRob, 125
  glance.lmRob, 138
  tidy.lmRob, 246
  tidy.lmRob, 266
* robustbase tidiers
  augment.lmrob, 32
  augment.lmrob, 44
  glance.lmrob, 139
  tidy.lmrob, 247
  tidy.lmrob, 267
* smoothing spline tidiers
  augment.smooth.spline, 77
  glance.smooth.spline, 169
* spatialreg tidiers
  augment.sarlm, 75
  glance.sarlm, 167
  tidy.sarlm, 316
* speedlm tidiers
  augment.speedlm, 78
  glance.speedglm, 170
  glance.speedlm, 171
  tidy.speedglm, 318
  tidy.speedlm, 320
* survdiff tidiers
  glance.survdiff, 173
  tidy.survdiff, 324
* survexp tidiers
  glance.survexp, 174
  tidy.survexp, 326
* survey tidiers
  tidy.svyglm, 330
* survfit tidiers
  tidy.survfit, 327
* survival tidiers
  augment.coxph, 16
  augment.survreg, 81
  glance.aareg, 90
  glance.cch, 100
  glance.coxph, 105
  glance.pyears, 160
  glance.survdiff, 173
  glance.survexp, 174
  glance.survfit, 176
  glance.survreg, 177
  tidy.aareg, 188
  tidy.cch, 206
  tidy.coxph, 217
  tidy.pyears, 302
  tidy.survdiff, 324
  tidy.survexp, 326
  tidy.survfit, 327
  tidy.survreg, 329
* survreg tidiers
  augment.survreg, 81
  glance.survreg, 177
  tidy.survreg, 329
* svd tidiers
  augment.prcomp, 66
  tidy.prcomp, 300
  tidy_irlba, 339
  tidy_svd, 342
* systemfit tidiers
  tidy.systemfit, 333
* time series tidiers
  tidy.acf, 189
  tidy.spec, 317
  tidy.ts, 335
  tidy.zoo, 338
INDEX

aareg_tidiers (tidy.aareg), 188
AER::ivreg(), 36, 37, 129, 130, 253
aer_tidiers (tidy.ivreg), 252
akima::interp(), 182, 183, 339, 341, 342, 344, 345
Arima_tidiers (tidy.Arima), 194
AUC::roc(), 312
auc_tidiers (tidy.roc), 312
augment(), 73, 75
augment.betamfx, 10, 52, 95, 143, 196, 277
augment.betareg, 12
augment.betareg(), 11
augment.clm, 14, 66, 103, 104, 159, 182, 210, 212, 298, 333
augment.coxph, 16, 83, 91, 101, 106, 161, 174, 175, 177, 178, 189, 207, 218, 303, 325, 326, 328, 330
augment.data.frame (data.frame_tidiers), 86
augment.decomposed.ts, 19, 81
augment.drc, 21, 110, 224
augment.factanal, 23, 113, 230
augment.felm, 25, 232
augment.fixest, 27, 235
augment.glm, 29, 41, 123, 135, 180, 244, 262, 264, 283
augment.glm(), 52
augment.glmRob, 31
augment.glmrob, 32, 45, 140, 248, 268
augment.hptest, 34, 252, 291, 299
augment.ivreg, 36, 130, 253
augment.kmeans, 38, 131, 259
augment.lm, 30, 39, 123, 135, 180, 244, 262, 264, 283
augment.lmRob, 42, 126, 139, 246, 267
augment.lmrob, 33, 44, 140, 248, 268
augment.loess, 46
augment.logitmfx (augment.mfx), 50
augment.Mclust, 48, 274
augment.mfx, 11, 50, 95, 143, 196, 277
augment.mjjoint, 53
augment.negbinmfx (augment.mfx), 50
augment.nlrq, 55, 73, 75, 150, 167, 286, 314, 316
augment.nls, 57, 151, 288
augment.NULL (null_tidiers), 184
augment.pam, 58, 154, 293
augment.plm, 60, 156, 294
augment.poissonmfx (augment.mfx), 50
augment.poLCA, 62, 157, 295
augment.polr, 16, 64, 103, 104, 159, 182, 210, 212, 298, 333
augment.pcrcomp, 66, 301, 340, 344
augment.probitmfx (augment.mfx), 50
augment.rlm, 68, 164, 310
augment.rma, 69
augment.rq, 56, 71, 75, 150, 167, 286, 314, 316
augment.rqs, 56, 73, 73, 150, 167, 286, 314, 316
augment.sarlm, 75, 168, 317
augment.smooth.spline, 77, 169
augment.speedlm, 78, 171, 172, 319, 321
augment.stl, 20, 80
augment.survreg, 18, 81, 91, 101, 106, 161, 174, 175, 177, 178, 189, 207, 218, 303, 325, 326, 328, 330
augment.columns, 83
base::as.data.frame.table(), 335
base::data.frame, 11, 13, 15, 17, 22, 24, 25, 27, 30, 33, 36, 38, 40, 43, 45, 46, 48, 51, 54, 56, 57, 59, 61, 63, 65, 67, 68, 72, 74, 77, 79, 82
**base::data.frame(), 11, 13, 15, 17, 22, 27, 30, 33, 36, 40, 43, 45, 46, 52, 56, 57, 65, 67, 68, 72, 74, 79, 82**

**base::svd(), 183, 343, 344**

**base::table, 335**

**bbmle::mle2(), 280, 281**

**bbmle_tidiers (tidy.mle2), 280**

**betareg::betareg(), 13, 14, 96, 97, 197, 198**

**betareg::predict.betareg(), 11**

**betareg::residuals.betareg(), 11**

**betareg_tidiers (tidy.betareg), 197**

**biglm::bigglm(), 98, 199, 200**

**biglm::biglm(), 98, 199, 200**

**bindesign_tidiers (tidy.binDesign), 200**

**binGroup::binDesign, 99**

**binGroup::binDesign(), 100, 201**

**binGroup::binWidth(), 202**

**binwidth_tidiers (tidy.binWidth), 201**

**boot::boot(), 203, 204**

**boot::boot.ci(), 203, 204**

**boot::tsboot(), 204**

**boot_tidiers (tidy.boot), 203**

**bootstrap, 84, 85, 87, 89, 90, 187, 221, 222, 236, 289**

**btergm::btergm(), 205**

**btergm_tidiers (tidy.btergm), 204**

**car::Anova(), 191**

**car::durbinWatsonTest(), 88**

**caret::confusionMatrix(), 216**

**caret_tidiers (tidy.confusionMatrix), 215**

**cch_tidiers (tidy.cch), 206**

**cfa_tidiers (tidy.lavaan), 259**

**cluster::pam(), 59, 60, 153, 154, 292, 293**

**coeftest_tidiers (tidy.coeftest), 213**

**confint(), 85**

**confint_tidy, 84, 85, 87, 89, 90, 187, 221, 222, 236, 289**

**confusionMatrix_tidiers (tidy.confusionMatrix), 215**

**coxph_tidiers (tidy.coxph), 217**

**data.frame_tidiers, 84, 85, 86, 89, 90, 187, 221, 222, 236, 289**

**decompose_tidiers**

**(augment.decomposed.ts), 19**

**drc::drm(), 22, 23, 109, 110, 223, 224**

**drc_tidiers (tidy.drc), 223**

**durbinWatsonTest_tidiers, 88**

**emmeans::contrast(), 225, 269, 306, 323**

**emmeans::emmeans(), 225, 269, 306, 323**

**emmeans::ref_grid(), 225, 269, 305, 306, 323**

**emmeans::summary.emmGrid(), 225, 269, 305, 323**

**emmeans_tidiers (tidy.lsmobj), 268**

**epiR::epi.2by2(), 227**

**epiR_tidiers (tidy.epi.2by2), 226**

**ergm::control.ergm(), 229**

**ergm::ergm(), 111, 228, 229**

**ergm::summary(), 111, 228, 229**

**ergm::summary.ergm(), 111**

**ergm_tidiers (tidy.ergm), 228**

**factanal_tidiers (tidy.factanal), 230**

**felm_tidiers (tidy.felm), 231**

**finish.glance, 84, 85, 87, 89, 90, 187, 221, 222, 236, 289**

**fitdistr_tidiers (tidy.fitdistr), 233**

**fix_data_frame, 84, 85, 87, 89, 90, 187, 221, 222, 236, 289**

**fixest::fepois(), 222**

**fixest_tidiers, 222**

**fixes::feols(), 28, 235**

**fixes::feols(), 28, 235**

**gam_tidiers (tidy.gam), 237**

**gamlss::gamlss(), 238**

**garch_tidiers (tidy.garch), 239**

**geeglm_tidiers (tidy.geeglm), 240**

**geepack::geepack(), 121, 122, 241**

**geepack_tidiers (tidy.geeglm), 240**


**glance.aareg, 18, 83, 90, 101, 106, 161, 174, 175, 177, 178, 189, 207, 218, 303, 325, 326, 328, 330**

**glance.aov, 92, 191, 192, 194, 271, 337**

**glance.Arima, 93, 195**

**glance.betamfx, 11, 52, 94, 143, 196, 277**
INDEX

glance.betareg, 96
  glance.betareg(), 95
  glance.biglm, 97, 200
  glance.binDesign, 99, 201, 202
  glance.cch, 18, 83, 91, 100, 106, 161, 174,
  175, 177, 178, 189, 207, 218, 303, 325, 326, 328, 330
  glance.clm, 16, 66, 102, 104, 159, 182, 210,
  212, 298, 333
  glance.clmm, 16, 66, 103, 103, 159, 182, 210,
  212, 298, 333
  glance.coxph, 18, 83, 91, 101, 105, 161, 174,
  175, 177, 178, 189, 207, 218, 303, 325, 326, 328, 330
  glance.cv.glmnet, 107, 124, 220, 245
  glance.data.frame(data.frame_tidiers), 86
  glance.drc, 23, 109, 224
  glance.durbinWatsonTest
    (durbinWatsonTest_tidiers), 88
  glance.ergm, 110, 229
  glance.factanal, 25, 112, 230
  glance.felm, 113
  glance.fitdistr, 115, 234
  glance.fixest, 116
  glance.gam, 118, 238
  glance.garch, 119, 240
  glance.geeglm, 121
  glance.glm, 30, 41, 122, 135, 180, 244, 262,
  264, 283
  glance.glm(), 143
  glance.glmmnet, 108, 123, 220, 245
  glance.glmmRob, 43, 125, 139, 246, 267
  glance.gmm, 126, 249
  glance.htest(tidy.htest), 251
  glance.ivreg, 37, 128, 253
  glance.kmeans, 39, 130, 259
  glance.lavaan, 132, 260
  glance.list(list_tidiers), 183
  glance.lm, 30, 41, 123, 134, 180, 244, 262,
  264, 283
  glance.lme02, 136, 266
  glance.lmRob, 43, 126, 138, 246, 267
  glance.lmrob, 33, 45, 139, 248, 268
  glance.logitmfx(glance.mfx), 142
  glance.Mclust, 140
  glance.mfx, 11, 52, 95, 142, 196, 277
  glance.mjoint, 144, 279
  glance.muhaz, 146, 284
  glance.multinom, 147, 285
  glance.negbinmfx(glance.mfx), 142
  glance.nlrq, 56, 73, 75, 149, 167, 286, 314,
  316
  glance.nls, 58, 150, 288
  glance.NULL(null_tidiers), 184
  glance.optim(glance_optim), 182
  glance.orcutt, 151, 290
  glance.pam, 60, 153, 293
  glance.plm, 62, 155, 294
  glance.poissonmfx(glance.mfx), 142
  glance.polCA, 63, 156, 295
  glance.polr, 16, 66, 103, 104, 158, 182, 210,
  212, 298, 333
  glance.probitmfx(glance.mfx), 142
  glance.pyears, 18, 83, 91, 101, 106, 160,
  174, 175, 177, 178, 189, 207, 218,
  303, 325, 326, 328, 330
  glance.ridgelm, 161, 309
  glance.rlm, 69, 163, 310
  glance.rma, 164
  glance.rq, 56, 73, 75, 150, 166, 286, 314, 316
  glance.sarlm, 76, 167, 317
  glance.smooth.spline, 78, 169
  glance.speedglm, 79, 170, 172, 319, 321
  glance.speedglm, 79, 171, 171, 319, 321
  glance.summaryDefault
    (summary_tidiers), 187
  glance.survdiff, 18, 83, 91, 101, 106, 161,
  173, 175, 177, 178, 189, 207, 218,
  303, 325, 326, 328, 330
  glance.survexp, 18, 83, 91, 101, 106, 161,
  174, 174, 177, 178, 189, 207, 218,
  303, 325, 326, 328, 330
  glance.survfit, 18, 83, 91, 101, 106, 161,
  174, 175, 176, 178, 189, 207, 218,
  303, 325, 326, 328, 330
  glance.survreg, 18, 83, 91, 101, 106, 161,
  174, 175, 177, 177, 189, 207, 218,
  303, 325, 326, 328, 330
  glance.sv glm, 30, 41, 123, 135, 179, 244,
  262, 264, 283
  glance.svyolr, 16, 66, 103, 104, 159, 181,
  210, 212, 298, 333
  glance_optim, 182, 184, 340, 342, 344, 345
  glmnet::cv.glmnet(), 107, 108, 219, 220
  glmnet::glmnet(), 124, 244, 245
glmnet_tidiers (tidy.glmnet), 244
  gmm::gmm(), 127, 249
  gmm_tidiers (tidy.gmm), 248
  graphics::image(), 345
  graphics::persp(), 345

Hmisc::rcorr(), 304
Hmisc_tidiers (tidy.rcorr), 303
htest_tidiers (tidy.htest), 251

irlba::irlba(), 339, 340
irlba_tidiers (tidy.irlba), 339
ivreg_tidiers (tidy.ivreg), 252

joineRML::bootSE(), 279
joineRML::fitted.mjoint(), 54
joineRML::mjoint(), 54, 145, 278, 279
joineRML::residuals.mjoint(), 54
joineRML_tidiers (tidy.mjoint), 278

kappa_tidiers (tidy.kappa), 254
kde_tidiers (tidy.kde), 255
Kendall::Kendall(), 257, 258
Kendall::MannKendall(), 257, 258
Kendall::SeasonalMannKendall(), 257, 258
Kendall_tidiers (tidy.Kendall), 257
kendall_tidiers (tidy.Kendall), 257
kmeans_tidiers (tidy.kmeans), 258
ks::kde(), 256
ks_tidiers (tidy.kde), 255

lavaan::cfa(), 132, 133, 260
lavaan::fitmeasures(), 133
lavaan::parameterEstimates(), 260
lavaan::sem(), 132, 133, 260
lavaan_tidiers (tidy.lavaan), 259
leaps::regsubsets(), 307, 308
leaps_tidiers (tidy.regsubsets), 307
lfe::felm(), 25, 26, 114, 231, 232
lfe_tidiers (tidy.felm), 231
list_tidiers, 183, 183, 340, 342, 344, 345
lm.beta::lm.beta, 263
lm_tidiers (tidy.lm), 261
lmodel2::lmodel2(), 136, 137, 265, 266
lmodel2_tidiers (tidy.lmodel2), 265
lmtest::coeftest(), 213, 214
lmtest_tidiers (tidy.coeftest), 213
loess_tidiers (augment.loess), 46

lsmeans::summary.ref.grid(), 225, 269, 305, 323

maps::map(), 272
maps_tidiers (tidy.map), 272
MASS::dropterm(), 297, 332
MASS::fitdistr(), 116, 233, 234
MASS::lm.ridge(), 162, 308, 309
MASS::polr(), 65, 66, 159, 297, 298
MASS::rlm(), 68, 69, 163, 164, 309, 310
MASS::select.ridgelm(), 162
mclust::Mclust(), 145
mclust_tidiers (tidy.Mclust), 273
mean, 86
mediate_tidiers (tidy.mediate), 274
mediation::mediate(), 275
metafor::escalc(), 311
metafor::rma(), 70, 165, 310
metafor::rma.glm(), 70, 165, 310
metafor::rma.mh(), 70, 165, 310
metafor::rma.mv(), 70, 165, 310
metafor::rma.peto(), 70, 165, 310
metafor::rma.uni(), 70, 165, 310
mfx::betamfx(), 11, 95, 196
mfx::logitmfx(), 52, 143, 277
mfx::negbinmfx(), 52, 143, 277
mfx::poissonmfx(), 52, 143, 277
mfx::probitmfx(), 52, 143, 277
mset::betamfx(), 11, 95, 196
mset::logitmfx(), 52, 143, 277
mset::negbinmfx(), 52, 143, 277
mset::poissonmfx(), 52, 143, 277
mset::probitmfx(), 52, 143, 277
mset::rma(), 70, 165, 310
mset::rma.glm(), 70, 165, 310
mset::rma.mh(), 70, 165, 310
mset::rma.mv(), 70, 165, 310
mset::rma.peto(), 70, 165, 310
mset::rma.uni(), 70, 165, 310
mset::summary.ref.grid(), 225, 269, 305, 323

multcomp::confint.glht(), 208, 214, 215
multcomp::glht(), 208, 214, 215, 242, 243, 321, 322
multcomp::summary.glht(), 208, 321, 322
multcomp_tidiers (tidy.glht), 242
multinom_tidiers (tidy.multinom), 284

nnet::multinom(), 148, 284, 285
nnet_tidiers (tidy.multinom), 284
null_tidiers, 184

optim_tidiers (tidy.optim), 341
INDEX

orcutt::cochrane.orcutt(), 152, 153, 289, 290
orcutt_tidiers (tidy.orcutt), 289
ordinal::clm(), 15, 102, 103, 209, 210
ordinal::clmm(), 104, 211, 212
ordinal::confint.clm(), 210, 212
ordinal::predict.clm(), 15
ordinal_tidiers (tidy.clm), 209

pam_tidiers (tidy.pam), 292
plm::plm(), 61, 62, 155, 156, 293, 294
plm_tidiers (tidy.plm), 293
polCA::polCA(), 63, 157, 295
polCA_tidiers (tidy.polCA), 295
polr_tidiers (tidy.polr), 296
prcomp_tidiers (tidy.prcomp), 300
predict.fixest(), 28
psych::cohen.kappa(), 254, 255
psych_tidiers (tidy.kappa), 254
purrr::map(), 166
purrr::map_df(), 184
pyears_tidiers (tidy.pyears), 302

qr(), 271
quantreg::nlrq(), 56, 149, 150, 286
quantreg::predict.rq(), 72, 74
quantreg::predict.rq(), 73
quantreg::predict.rqs(), 75
quantreg::rq(), 72–75, 166, 167, 314–316
quantreg::summary.rq(), 314, 315
quantreg::summary.rqs(), 315
quantreg_tidiers (tidy.rq), 313

rcorr_tidiers (tidy.rcorr), 303
ridgelm_tidiers (tidy.ridgelm), 308
rlm_tidiers (glance.rlm), 163
robust::glmRob(), 125, 126, 246
robust::lmRob(), 43, 138, 139, 266, 267
robust_tidiers (tidy.lmRob), 266
robustbase::glmrob(), 32, 33, 247, 248
robustbase::lmrob(), 45, 139, 140, 267, 268
robustbase_tidiers (tidy.lmrob), 267
roc_tidiers (tidy.roc), 312
rq_tidiers (tidy.rq), 313
rqs_tidiers (tidy.rqs), 315
rsample::bootstraps(), 204

sem_tidiers (tidy.lavaan), 259
sexpfit_tidiers (tidy.survexp), 326
smooth.spline_tidiers
(augment.smooth.spline), 77
sp_tidiers, 186
sparse_tidiers, 185
spatialreg::errorsarlm(), 76, 167, 168, 316, 317
spatialreg::lagsarlm(), 76, 167, 168, 316, 317
spatialreg::sacsarlm(), 168, 317
spatialreg_tidiers (tidy.sarlm), 316
speedglm::speedglm(), 170, 319
speedglm::speedlm(), 79, 171, 172, 320, 321
speedglm_tidiers (tidy.speedglm), 318
speedlm_tidiers (tidy.speedlm), 320
splines::ns(), 10, 13, 15, 16, 19, 22, 24, 25, 27, 29, 31, 32, 34, 36, 38, 40, 43, 44, 48, 50, 53, 57, 59, 61, 62, 65, 67, 68, 70, 72, 74, 75, 78, 80, 81
stats::acf(), 190
stats::anova(), 191
stats::aov(), 92, 192–194
stats::arima(), 93, 94, 194, 195
stats::ccf(), 190
stats::chisq.test(), 34, 35, 251, 252
stats::cooks.distance(), 14
stats::cor.test(), 34, 251, 252
stats::decompose(), 20
stats::density(), 221
stats::dist(), 222
stats::factanal(), 24, 25, 112, 113, 230
stats::ftable(), 236
stats::glm(), 29, 30, 122, 123, 180, 244, 331
stats::kmeans(), 38, 39, 131, 258, 259
stats::lm(), 24, 40, 134, 261, 282
stats::loess(), 46, 47
stats::manova(), 271
stats::na.action, 18, 41, 47
stats::nls(), 57, 58, 150, 151, 287, 288
stats::optim(), 182, 183, 339, 341, 342, 344
stats::pacf(), 190
stats::pairwise.t.test(), 291
stats::pairwise.wilcox.test(), 291
stats::poly(), 10, 13, 15, 16, 19, 22, 24, 25, 27, 29, 31, 32, 34, 36, 38, 40, 43, 44, 48, 50, 53, 57, 59, 61, 62, 65, 67, 68, 70, 72, 74, 75, 78, 80, 81
INDEX

stats::power.t.test(), 299
stats::prcomp(), 67, 300, 301
stats::predict(), 13, 17, 33, 82
stats::predict.glm(), 30, 52
stats::predict.lm(), 41
stats::predict.loess(), 47
stats::predict.nls(), 58
stats::predict.smooth.spline(), 78
stats::residuals(), 13, 17, 33, 82
stats::residuals.glm(), 30, 52
stats::rstandard.glm(), 30, 52
stats::smooth.spline(), 77, 78, 169
stats::spectrum(), 318
stats::stl(), 80, 81
stats::summary.lm(), 262
stats::summary.manova, 271
stats::summary.manova(), 271
stats::summary.nls(), 288
stats::t.test(), 34, 251, 252
stats::ts(), 336
stats::TukeyHSD(), 337
stats::wilcox.test(), 34, 251, 252
summary(), 187
summary.fixest, 28, 117, 234
summary_tidiers, 84, 85, 87, 89, 90, 187, 221, 222, 236, 289
survdiff_tidiers(tidy.survdiff), 324
survexp_tidiers(tidy.survexp), 326
survey::anova.svyglm, 180
survey::svyglm(), 180, 331
survey::svyolr(), 181, 182, 332, 333
survfit_tidiers(tidy.survfit), 327
survival::aareg(), 91, 188, 189
survival::cch(), 101, 206, 207
survival::coxph(), 17, 18, 105, 106, 217, 218
survival::pyears(), 160, 161, 302, 303
survival::Surv(), 10, 13, 15, 16, 19, 22, 24, 25, 27, 29, 31, 32, 34, 36, 38, 40, 43, 44, 48, 50, 53, 57, 59, 61, 63, 65, 67, 68, 70, 72, 74, 75, 78, 80, 81
survival::survdiff(), 173, 174, 325
survival::survexp(), 175, 326
survival::survfit(), 176, 177, 327, 328
survival::survreg(), 82, 83, 178, 329, 330
survreg_tidiers(tidy.survreg), 329
svd(), 182, 339, 341, 342, 344
svd_tidiers, 67, 301
svd_tidyrs(tidy_svd), 342
svyolr_tidiers(tidy.svyolr), 331
systemfit::systemfit(), 333, 334
systemfit_tidiers(tidy.systemfit), 333
tibble::as_tibble(), 236, 334, 335
tibble::as_tibble.table(), 335
tibble::tibble(), 10, 13, 15, 16, 19, 20, 22, 24, 25, 27, 29, 31, 32, 34, 36, 38, 40, 43, 44, 48, 50, 53, 57, 59, 61, 62, 65, 67, 68, 70, 72, 73, 75, 78, 80, 81, 122, 134, 172, 221, 222, 229, 236, 300, 335, 340, 343, 345
tidy(), 15, 58, 103, 104, 151, 159, 182, 210, 212, 288, 298, 333
tidy.aareg, 18, 83, 91, 101, 106, 161, 174, 175, 177, 178, 188, 207, 218, 303, 325, 326, 328, 330
tidy.acf, 189, 318, 336, 338
tidy.anova, 93, 190, 192, 194, 271, 337
tidy.aov, 93, 191, 192, 194, 271, 337
tidy.aovlist, 93, 191, 192, 193, 271, 337
 tidy.Arima, 94, 194
 tidy.betamfx, 11, 52, 95, 143, 195, 277
 tidy.betareg, 197
 tidy.betareg(), 196
 tidy.biglm, 98, 199
 tidy.binDesign, 100, 200, 202
 tidy.binWidth, 100, 201, 201
 tidy.boot, 203
 tidy.btergm, 204
 tidy.cchs, 18, 83, 91, 101, 106, 161, 174, 175, 177, 178, 189, 206, 218, 303, 325, 326, 328, 330
 tidy.character (tidy.numeric), 288
 tidy.clad, 208, 215, 243, 322
 tidy.clm, 16, 66, 103, 104, 159, 182, 209, 212, 298, 333
 tidy.clmm, 16, 66, 103, 104, 159, 182, 210, 211, 298, 333
 tidy.coefest, 213
 tidy.confint.glht, 208, 214, 243, 322
 tidy.confusionMatrix, 215
 tidy.coxph, 18, 83, 91, 101, 106, 161, 174, 175, 177, 178, 189, 207, 217, 303, 325, 326, 328, 330
 tidy.cv.glmnet, 108, 124, 219, 245
 tidy.data.frame (data.frame_tidiers), 86
 tidy.density, 84, 85, 87, 89, 90, 187, 221, 222, 236, 289
 tidy.dgCMatrix (sparse_tidiers), 185
 tidy.dgTMatrix (sparse_tidiers), 185
 tidy.dist, 84, 85, 87, 89, 90, 187, 221, 222, 236, 289
 tidy.drc, 23, 110, 223
 tidy.durbinWatsonTest
 (durbinWatsonTest_tidiers), 88
 tidy.emmGrid, 224, 269, 306, 323
 tidy.epi.2by2, 226
 tidy.ergm, 111, 228
 tidy.factanal, 25, 113, 230
 tidy.felm, 26, 231
 tidy.fitdistr, 116, 233
 tidy.fixest, 28, 234
 tidy.ftable, 84, 85, 87, 89, 90, 187, 221, 222, 236, 289
 tidy.gam, 119, 237
 tidy.gamlss, 238
 tidy.garch, 120, 239
 tidy.geeglm, 240
 tidy.glht, 208, 215, 242, 322
 tidy.glm, 30, 41, 123, 135, 180, 243, 262, 264, 283
 tidy.glmnet, 108, 124, 220, 244
 tidyglmRob, 43, 126, 139, 246, 267
 tidy.glmrob, 33, 45, 140, 247, 268
 tidy.gmm, 127, 248
 tidy.htest, 35, 251, 291, 299
 tidy.irlba (tidy.irlba), 339
 tidy.ivreg, 37, 130, 252
 tidy.kappa, 254
 tidy.kde, 255
 tidy.Kendall, 257
 tidy.kmeans, 39, 131, 258
 tidy.lavaan, 133, 259
 tidy.Line (sp_tidiers), 186
 tidy.Lines (sp_tidiers), 186
 tidy.list (list_tidiers), 183
 tidy.lm, 30, 41, 123, 135, 180, 244, 261, 264, 283
 tidy.lm(), 294, 321
 tidy.lm.beta, 30, 41, 123, 135, 180, 244, 262, 263, 283
 tidy.lmodel2, 137, 265
 tidy.lmRob, 43, 126, 139, 246, 266
 tidy.lmrob, 33, 45, 140, 248, 267
 tidy.logical (tidy.numeric), 288
 tidy.logitmfx (tidy.mfx), 276
 tidy.lsmobj, 225, 268, 306, 323
 tidy.manova, 93, 191, 192, 194, 270, 337
 tidy.map, 272
 tidy.Mclust, 49, 273
 tidy.mediate, 274
 tidy.mfx, 11, 52, 95, 143, 196, 276
 tidy.mjtest, 145, 278
 tidy.mle2, 280
 tidy.mlm, 30, 41, 123, 135, 180, 244, 262, 264, 282
 tidy.mlm(), 261
 tidy.nearlyzerosum, 147, 283
 tidy.multinom, 148, 284
 tidy.negbinmfx (tidy.mfx), 276
 tidy.nlrq, 56, 73, 75, 150, 167, 285, 314, 316
 tidy.nls, 58, 151, 287
 tidy.NULL (null_tidiers), 184
 tidy.numeric, 84, 85, 87, 89, 90, 187, 221, 222, 236, 288
tidy.optim (tidy_optim), 341
tidy.orcutt, 153, 289
tidy.pairwise.htest, 35, 252, 290, 299
tidy.pam, 60, 154, 292
tidy.plm, 62, 156, 293
tidy.poissonmfx (tidy.mfx), 276
tidy.polr, 16, 66, 103, 104, 159, 182, 210, 212, 296, 323
tidy.Polygon (sp_tidiers), 186
tidy.Polygons (sp_tidiers), 186
tidy.power.htest, 35, 252, 291, 298
tidy.pcomp, 67, 300, 340, 344
tidy.probitmfx (tidy.mfx), 276
tidy.pyears, 18, 83, 91, 101, 106, 161, 174, 175, 177, 178, 189, 207, 218, 303, 325, 326, 327, 330
tidy survfit, 18, 83, 91, 101, 106, 161, 174, 175, 177, 178, 189, 207, 218, 303, 325, 326, 327, 330
tidy survreg, 18, 83, 91, 101, 106, 161, 174, 175, 177, 178, 189, 207, 218, 303, 325, 326, 328, 329
tidy survfit, 18, 83, 91, 101, 106, 161, 174, 175, 177, 178, 189, 207, 218, 303, 325, 326, 328, 329
tidy svyglm, 330
tidy svyolr, 16, 66, 103, 104, 159, 182, 210, 212, 298, 331
tidy systemfit, 333
tidy table, 334
tidy ts, 190, 318, 335, 338
tidy TukeyHSD, 93, 191, 192, 194, 271, 336
tidy zoo, 190, 318, 336, 338
tidy irlba, 67, 183, 184, 301, 339, 342, 344, 345
tidy optim, 183, 184, 340, 341, 344, 345
tidy optim(), 281
tidy svd, 67, 183, 184, 301, 340, 342, 342, 345
tidy svd(), 340
tidy xyz, 183, 184, 340, 342, 344, 344
tidyr pivot_longer(), 335
tseries garch(), 120, 240
xyz_tidiers (tidy.xyz), 344
zoo::zoo(), 338
zoo_tidiers (tidy.zoo), 338