Package 'broom'

March 29, 2018

Type Package

Title Convert Statistical Analysis Objects into Tidy Data Frames

Version 0.4.4

Date 2018-03-12

Maintainer David Robinson <admiral.david@gmail.com>

Description Convert statistical analysis objects from R into tidy data frames, so that they can more easily be combined, reshaped and otherwise processed with tools like 'dplyr', 'tidyr' and 'ggplot2'. The package provides three S3 generics: tidy, which summarizes a model's statistical findings such as coefficients of a regression; augment, which adds columns to the original data such as predictions, residuals and cluster assignments; and glance, which provides a one-row summary of model-level statistics.

Imports plyr, dplyr, tidyr, psych, stringr, reshape2, nlme, methods

Suggests knitr, boot, survival, gam, glmnet, lfe, Lahman, MASS, rgeos, sp, maps, maptools, multcomp, testthat, lme4, zoo, lmtest, plm, biglm, ggplot2, nnet, geepack, AUC, ergm, network, statnet.common, xergm, btergm, binGroup, Hmisc, bbmle, gamlss, rstan, rstanarm, brms, coda, gmm, Matrix, ks, purrr, orcutt, mgcv, lmodel2, poLCA, mclust, covr, lsmeans, emmeans, betareg, robust, akima, AER, muhaz, speedglm, tibble

URL http://github.com/tidyverse/broom

BugReports http://github.com/tidyverse/broom/issues

VignetteBuilder knitr

License MIT + file LICENSE

RoxygenNote 6.0.1

NeedsCompilation no

Author David Robinson [aut, cre], Matthieu Gomez [ctb], Boris Demeshev [ctb], Dieter Menne [ctb], Benjamin Nutter [ctb],

Luke Johnston [ctb],
Ben Bolker [ctb],
Francois Briatte [ctb],
Jeffrey Arnold [ctb],
Jonah Gabry [ctb],
Luciano Selzer [ctb],
Gavin Simpson [ctb],
Jens Preussner [ctb],
Jay Hesselberth [ctb],
Matthew Lincoln [ctb],
Derek Chiu [ctb],
Hadley Wickham [ctb]

Repository CRAN

Date/Publication 2018-03-29 15:39:33 UTC

R topics documented:

`aareg_tidiers` *Tidiers for aareg objects*

Description

These tidy the coefficients of Aalen additive regression objects.

Usage

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

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

Arguments

`x`	an "aareg" object
`...`	extra arguments (not used)

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy.aareg` returns one row for each coefficient, with the columns

`term`	name of coefficient
`estimate`	estimate of the slope
`statistic`	test statistic for coefficient
`std.error`	standard error of statistic
`robust.se`	robust version of standard error estimate (only when x was called with `dfbeta = TRUE`)
`z`	z score
`p.value`	p-value

`glance` returns a one-row data frame containing

`statistic`	chi-squared statistic
`p.value`	p-value based on chi-squared statistic
`df`	degrees of freedom used by coefficients

Examples

```
if (require("survival", quietly = TRUE)) {
    afit <- aareg(Surv(time, status) ~ age + sex + ph.ecog, data=lung,
                  dfbeta=TRUE)
    summary(afit)
    tidy(afit)
}
```

---

acf_tidiers *Tidying method for the acf function*

Description

Tidy an "acf" object, which is the output of `acf` and the related `pcf` and `ccf` functions.

Usage

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

Arguments

`x`	acf object
`...`	(not used)

Value

data.frame with columns

lag	lag values
acf	calucated correlation

Examples

```
# acf
result <- acf(lh, plot=FALSE)
tidy(result)

# ccf
result <- ccf(mdeaths, fdeaths, plot=FALSE)
tidy(result)

# pcf
result <- pacf(lh, plot=FALSE)
tidy(result)

# lag plot
library(ggplot2)
result <- tidy(acf(lh, plot=FALSE))
p <- ggplot(result, aes(x=lag, y=acf)) +
         geom_bar(stat='identity', width=0.1) +
         theme_bw()
p

# with confidence intervals
conf.level <- 0.95
# from \code{plot.acf} method
len.data <- length(lh) # same as acf$n.used
conf.int <- qnorm((1 + conf.level) / 2) / sqrt(len.data)
p + geom_hline(yintercept = c(-conf.int, conf.int),
               color='blue', linetype='dashed')
```

---

anova_tidiers *Tidying methods for anova and AOV objects*

Description

Tidies the result of an analysis of variance into an ANOVA table. Only a tidy method is provided, not an augment or glance method.

Usage

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

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

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

Arguments

x	An object of class "anova", "aov", or "aovlist"
...	extra arguments (not used)

Details

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

Value

A data.frame with columns

term	Term within the model, or "Residuals"
df	Degrees of freedom used by this term in the model
sumsq	Sum of squares explained by this term
meansq	Mean of sum of squares among degrees of freedom
statistic	F statistic
p.value	P-value from F test

In the case of an "aovlist" object, there is also a stratum column describing the error stratum

Examples

```
a <- anova(lm(mpg ~ wt + qsec + disp, mtcars))
tidy(a)

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

al <- aov(mpg ~ wt + qsec + Error(disp / am), mtcars)
tidy(al)
```

`Arima_tidiers` *Tidying methods for ARIMA modeling of time series*

Description

These methods tidy the coefficients of ARIMA models of univariate time series.

Usage

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

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

Arguments

`x`	An object of class "Arima"
`conf.int`	whether to include a confidence interval
`conf.level`	confidence level of the interval, used only if `conf.int=TRUE`
`...`	extra arguments (not used)

Details

`augment` is not currently implemented, as it is not clear whether ARIMA predictions can or should be merged with the original data frame.

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns one row for each coefficient in the model, with five columns:

`term`	The term in the nonlinear model being estimated and tested
`estimate`	The estimated coefficient
`std.error`	The standard error from the linear model

If `conf.int = TRUE`, also returns

`conf.low`	low end of confidence interval
`conf.high`	high end of confidence interval

`glance` returns one row with the columns

`sigma`	the square root of the estimated residual variance
`logLik`	the data's log-likelihood under the model
`AIC`	the Akaike Information Criterion
`BIC`	the Bayesian Information Criterion

See Also

arima

Examples

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

`auc_tidiers` *Tidiers for objects from the AUC package*

Description

Tidy "roc" objects from the "auc" package. This can be used to, for example, draw ROC curves in ggplot2.

Usage

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

Arguments

`x`	an "roc" object
`...`	Additional arguments, not used

Value

A data frame with three columns:

`cutoff`	The cutoff of the prediction scores used for classification
`tpr`	The resulting true positive rate at that cutoff
`fpr`	The resulting false positive rate at that cutoff

If the labels had names, those are added as an "instance" column.

Examples

```
if (require("AUC", quietly = TRUE)) {
  data(churn)
  r <- roc(churn$predictions,churn$labels)

  td <- tidy(r)
  head(td)
```

```
  library(ggplot2)
  ggplot(td, aes(fpr, tpr)) +
    geom_line()

  # compare the ROC curves for two prediction algorithms
  library(dplyr)
  library(tidyr)

  rocs <- churn %>%
    tidyr::gather(algorithm, value, -labels) %>%
    group_by(algorithm) %>%
    do(tidy(roc(.$value, .$labels)))

  ggplot(rocs, aes(fpr, tpr, color = algorithm)) +
    geom_line()
}
```

---

augment *Augment data according to a tidied model*

Description

Given an R statistical model or other non-tidy object, add columns to the original dataset such as predictions, residuals and cluster assignments.

Usage

```
augment(x, ...)
```

Arguments

x	model or other R object to convert to data frame
...	other arguments passed to methods

Details

Note that by convention the first argument is almost always `data`, which specifies the original data object. This is not part of the S3 signature, partly because it prevents rowwise_df_tidiers from taking a column name as the first argument.

This generic originated in the ggplot2 package, where it was called "fortify."

See Also

augment.lm

`augment_columns` *add fitted values, residuals, and other common outputs to an augment call*

Description

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

Usage

```
augment_columns(x, data, newdata, 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.

`betareg_tidiers` *Tidy betareg objects from the betareg package*

Description

Tidy beta regression objects into summarized coefficients, add their fitted values and residuals, or find their model parameters.

Usage

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

## S3 method for class 'betareg'
augment(x, data = stats::model.frame(x), newdata,
  type.predict, type.residuals, ...)

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

Arguments

x	A "betareg" object
conf.int	whether to include a confidence interval
conf.level	confidence level of the interval, used only if conf.int=TRUE
...	Extra arguments, not used
data	Original data frame the regression was fit on
newdata	New data frame to use for prediction
type.predict	Type of predictions to calculate
type.residuals	Type of residuals to calculate

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

tidy returns a data.frame with one row for each term used to predict the mean, along with at least one term used to predict phi (the inverse of the variance). It starts with the column component containing either "mean" or "precision" to describe which is being modeled, then has the same columns as tidied linear models or glm's (see lm_tidiers).

augment returns the original data, along with new columns describing each observation:

.fitted	Fitted values of model
.resid	Residuals
.cooksd	Cooks distance, cooks.distance

glance returns a one-row data.frame with the columns

pseudo.r.squared	the deviance of the null model
logLik	the data's log-likelihood under the model
AIC	the Akaike Information Criterion
BIC	the Bayesian Information Criterion
df.residual	residual degrees of freedom
df.null	degrees of freedom under the null

Examples

```
if (require("betareg", quietly = TRUE)) {
  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)

  head(augment(mod))

  glance(mod)
}
```

`biglm_tidiers` *Tidiers for biglm and bigglm object*

Description

Tidiers for biglm object from the "biglm" package, which contains a linear model object that is limited in memory usage. Generally the behavior is as similar to the `lm_tidiers` as is possible. Currently no `augment` is defined.

Usage

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

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

Arguments

`x`	a "biglm" object
`conf.int`	whether to include a confidence interval
`conf.level`	confidence level of the interval, used only if `conf.int=TRUE`
`exponentiate`	whether to exponentiate the coefficient estimates and confidence intervals (typical for logistic regression)
`quick`	whether to compute a smaller and faster version, containing only the `term` and `estimate` columns.
`...`	extra arguments (not used)

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy.biglm` returns one row for each coefficient, with columns

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

If `conf.int=TRUE`, it also includes columns for `conf.low` and `conf.high`, computed with `confint`.

`glance.biglm` returns a one-row data frame, with columns

`r.squared`	The percent of variance explained by the model
`AIC`	the Akaike Information Criterion
`deviance`	deviance
`df.residual`	residual degrees of freedom

Examples

```
if (require("biglm", quietly = TRUE)) {
    bfit <- biglm(mpg ~ wt + disp, mtcars)
    tidy(bfit)
    tidy(bfit, conf.int = TRUE)
    tidy(bfit, conf.int = TRUE, conf.level = .9)

    glance(bfit)

    # bigglm: logistic regression
    bgfit <- bigglm(am ~ mpg, mtcars, family = binomial())
    tidy(bgfit)
    tidy(bgfit, exponentiate = TRUE)
    tidy(bgfit, conf.int = TRUE)
    tidy(bgfit, conf.int = TRUE, conf.level = .9)
    tidy(bgfit, conf.int = TRUE, conf.level = .9, exponentiate = TRUE)

    glance(bgfit)
}
```

`binDesign_tidiers` *Tidy a binDesign object*

Description

Tidy a binDesign object from the "binGroup" package, which determines the sample size needed for a particular power.

Usage

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

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

Arguments

`x`	A "binDesign" object
`...`	Extra arguments (not used)

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

The `tidy` method returns a data.frame with one row for each iteration that was performed, with columns

`n`	Number of trials in this iteration
`power`	The power achieved for this n

The `glance` method returns a one-row data.frame with columns

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

Examples

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

    glance(des)
    head(tidy(des))
```

```
    # the ggplot2 equivalent of plot(des)
    library(ggplot2)
    ggplot(tidy(des), aes(n, power)) +
        geom_line()
}
```

`binWidth_tidiers` *Tidy a binWidth object*

Description

Tidy a binWidth object from the "binGroup" package, which calculates the expected width of a confidence interval from a binomial test.

Usage

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

Arguments

`x`	A "binWidth" object
`...`	Extra arguments (not used)

Value

A one-row data.frame with columns:

`ci.width`	Expected width of confidence interval
`alternative`	Alternative hypothesis
`p`	True proportion
`n`	Total sample size

Examples

```
if (require("binGroup", quietly = TRUE)) {
    bw <- binWidth(100, .1)
    bw
    tidy(bw)

    library(dplyr)
    d <- expand.grid(n = seq(100, 800, 100),
                     p = .5,
                     method = c("CP", "Blaker", "Score", "Wald"),
                     stringsAsFactors = FALSE) %>%
        group_by(n, p, method) %>%
```

```
        do(tidy(binWidth(.$n, .$p, method = .$method)))

    library(ggplot2)
    ggplot(d, aes(n, ci.width, color = method)) +
        geom_line() +
        xlab("Total Observations") +
        ylab("Expected CI Width")
}
```

---

bootstrap *Set up bootstrap replicates of a dplyr operation*

Description

Set up bootstrap replicates of a dplyr operation

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 tbl.

Details

This code originates from Hadley Wickham (with a few small corrections) here:

https://github.com/hadley/dplyr/issues/269

Some examples can be found at

https://github.com/dgrtwo/broom/blob/master/vignettes/bootstrapping.Rmd

Examples

```
library(dplyr)
mtcars %>% bootstrap(10) %>% do(tidy(lm(mpg ~ wt, .)))
```

boot_tidiers *Tidying methods for bootstrap computations*

Description

Tidying methods for "boot" objects from the "boot" package.

Usage

```
## S3 method for class 'boot'
tidy(x, conf.int = FALSE, conf.level = 0.95,
  conf.method = "perc", ...)
```

Arguments

x	boot object
conf.int	whether to include a confidence interval
conf.level	confidence level for CI
conf.method	method for computing confidence intervals (see boot.ci)
...	extra arguments (not used)

Value

The tidy method returns a data frame with one row per bootstrapped statistic that was calculated, and the following columns:

term	Name of the computed statistic, if present
statistic	The original values of the statistic
bias	The bias of the original statistic value
std.error	Standard error of the statistic

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.

Examples

```
if (require("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)
}
```

---

brms_tidiers *Tidying methods for a brms model*

Description

These methods tidy the estimates from brmsfit-objects (fitted model objects from the **brms** package) into a summary.

Usage

```
## S3 method for class 'brmsfit'
tidy(x, parameters = NA, par_type = c("all",
  "non-varying", "varying", "hierarchical"), robust = FALSE,
  intervals = TRUE, prob = 0.9, ...)
```

Arguments

x	Fitted model object from the **brms** package. See brmsfit-class.
parameters	Names of parameters for which a summary should be returned, as given by a character vector or regular expressions. If NA (the default) summarized parameters are specified by the par_type argument.
par_type	One of "all", "non-varying", "varying", or "hierarchical" (can be abbreviated). See the Value section for details.
robust	Whether to use median and median absolute deviation rather than mean and standard deviation.
intervals	If TRUE columns for the lower and upper bounds of posterior uncertainty intervals are included.
prob	Defines the range of the posterior uncertainty intervals, such that 100 * prob% of the parameter's posterior distribution lies within the corresponding interval. Only used if intervals = TRUE.
...	Extra arguments, not used

Value

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

When `parameters = NA`, the `par_type` argument is used to determine which parameters to summarize.

Generally, `tidy.brmsfit` returns one row for each coefficient, with at least three columns:

`term`	The name of the model parameter.
`estimate`	A point estimate of the coefficient (mean or median).
`std.error`	A standard error for the point estimate (sd or mad).

When `par_type = "non-varying"`, only population-level effects are returned.

When `par_type = "varying"`, only group-level effects are returned. In this case, two additional columns are added:

`group`	The name of the grouping factor.
`level`	The name of the level of the grouping factor.

Specifying `par_type = "hierarchical"` selects the standard deviations and correlations of the group-level parameters.

If `intervals = TRUE`, columns for the `lower` and `upper` bounds of the posterior intervals computed.

See Also

brms, brmsfit-class

Examples

```
## Not run:
 library(brms)
 fit <- brm(mpg ~ wt + (1|cyl) + (1+wt|gear), data = mtcars,
            iter = 500, chains = 2)
 tidy(fit)
 tidy(fit, parameters = "^sd_", intervals = FALSE)
 tidy(fit, par_type = "non-varying")
 tidy(fit, par_type = "varying")
 tidy(fit, par_type = "hierarchical", robust = TRUE)

## End(Not run)
```

broom — *Convert Statistical Analysis Objects into Tidy Data Frames*

Description

Convert statistical analysis objects from R into tidy data frames, so that they can more easily be combined, reshaped and otherwise processed with tools like dplyr, tidyr and ggplot2. The package provides three S3 generics: tidy, which summarizes a model's statistical findings such as coefficients of a regression; augment, which adds columns to the original data such as predictions, residuals and cluster assignments; and glance, which provides a one-row summary of model-level statistics.

btergm_tidiers — *Tidying method for a bootstrapped temporal exponential random graph model*

Description

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

```
## S3 method for class 'btergm'
tidy(x, conf.level = 0.95, exponentiate = FALSE,
  quick = FALSE, ...)
```

Arguments

x	a btergm object
conf.level	confidence level of the bootstrapped interval
exponentiate	whether to exponentiate the coefficient estimates and confidence intervals
quick	whether to compute a smaller and faster version, containing only the term and estimate columns.
...	extra arguments (currently not used)

Details

There is no augment or glance method for **ergm** objects.

Value

A `data.frame` without rownames.

`tidy.btergm` returns one row for each coefficient, with four columns:

`term`	The term in the model being estimated and tested
`estimate`	The estimated coefficient
`conf.low`	The lower bound of the confidence interval
`conf.high`	The lower bound of the confidence interval

See Also

`btergm`

Examples

```
if (require("xergm")) {
    # Using the same simulated example as the xergm package
    # 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)
}
```

`cch_tidiers` *tidiers for case-cohort data*

Description

Tidiers for case-cohort analyses: summarize each estimated coefficient, or test the overall model.

Usage

```
## S3 method for class 'cch'
tidy(x, conf.level = 0.95, ...)

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

Arguments

`x`	a "cch" object
`conf.level`	confidence level for CI
`...`	extra arguments (not used)

Details

It is not clear what an `augment` method would look like, so none is provided. Nor is there currently any way to extract the covariance or the residuals.

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns a data.frame with one row for each term

`term`	name of term
`estimate`	estimate of coefficient
`stderror`	standard error
`statistic`	Z statistic
`p.value`	p-value
`conf.low`	low end of confidence interval
`conf.high`	high end of confidence interval

`glance` returns a one-row data.frame with the following columns:

`score`	score
`rscore`	rscore
`p.value`	p-value from Wald test

iter	number of iterations
n	number of predictions
nevent	number of events

See Also

cch

Examples

```
if (require("survival", quietly = TRUE)) {
    # 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)

    # compare between methods
    library(dplyr)
    fits <- data_frame(method = c("Prentice", "SelfPrentice", "LinYing")) %>%
        group_by(method) %>%
        do(tidy(cch(Surv(edrel, rel) ~ stage + histol + age, data = ccoh.data,
                    subcoh = ~subcohort, id= ~seqno, cohort.size = 4028,
                    method = .$method)))

    # coefficient plots comparing methods
    ggplot(fits, aes(x = estimate, y = term, color = method)) + geom_point() +
        geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
        geom_vline(xintercept = 0)
}
```

compact *Remove NULL items in a vector or list*

Description

Remove NULL items in a vector or list

Usage

```
compact(x)
```

Arguments

x	a vector or list

confint.geeglm *Confidence interval for* geeglm *objects*

Description

Generate confidence intervals for GEE analyses

Usage

```
## S3 method for class 'geeglm'
confint(object, parm, level = 0.95, ...)
```

Arguments

object	The 'geeglm' object
parm	The parameter to calculate the confidence interval for. If not specified, the default is to calculate a confidence interval on all parameters (all variables in the model).
level	confidence level of the interval, used only if conf.int=TRUE
...	Additional parameters

Details

This function was taken from http://stackoverflow.com/a/21221995/2632184.

Value

Returns the upper and lower confidence intervals

confint_tidy *Calculate confidence interval as a tidy data frame*

Description

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

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`	Function to use for computing confint
`...`	extra arguments passed on to `confint`

Value

A data frame with two columns: `conf.low` and `conf.high`.

See Also

confint

coxph_tidiers *Tidiers for coxph object*

Description

Tidy the coefficients of a Cox proportional hazards regression model, construct predictions, or summarize the entire model into a single row.

Usage

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

## S3 method for class 'coxph'
augment(x, data = stats::model.frame(x), newdata,
  type.predict = "lp", type.residuals = "martingale", ...)

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

Arguments

x	"coxph" object
exponentiate	whether to report the estimate and confidence intervals on an exponential scale
conf.int	confidence level to be used for CI
...	Extra arguments, not used
data	original data for augment
newdata	new data on which to do predictions
type.predict	type of predicted value (see predict.coxph)
type.residuals	type of residuals (see residuals.coxph)

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

tidy returns a data.frame with one row for each term, with columns

estimate	estimate of slope
std.error	standard error of estimate
statistic	test statistic
p.value	p-value

augment returns the original data.frame with additional columns added:

.fitted	predicted values
.se.fit	standard errors
.resid	residuals (not present if newdata is provided)

glance returns a one-row data.frame with statistics calculated on the cox regression.

See Also

na.action

Examples

```
if (require("survival", quietly = TRUE)) {
    cfit <- coxph(Surv(time, status) ~ age + sex, lung)

    tidy(cfit)
    tidy(cfit, exponentiate = TRUE)

    lp <- augment(cfit, lung)
    risks <- augment(cfit, lung, type.predict = "risk")
    expected <- augment(cfit, lung, type.predict = "expected")

    glance(cfit)

    # also works on clogit models
    resp <- levels(logan$occupation)
    n <- nrow(logan)
    indx <- rep(1:n, length(resp))
    logan2 <- data.frame(logan[indx,],
                         id = indx,
                         tocc = factor(rep(resp, each=n)))
    logan2$case <- (logan2$occupation == logan2$tocc)

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

    library(ggplot2)
    ggplot(lp, aes(age, .fitted, color = sex)) + geom_point()
    ggplot(risks, aes(age, .fitted, color = sex)) + geom_point()
    ggplot(expected, aes(time, .fitted, color = sex)) + geom_point()
}
```

---

cv.glmnet_tidiers *Tidiers for glmnet cross-validation objects*

Description

Tidying methods for cross-validation performed by `glmnet.cv`, summarizing the mean-squared-error across choices of the penalty parameter lambda.

Usage

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

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

Arguments

x	a "cv.glmnet" object
...	extra arguments (not used)

Details

No augment method exists for this class.

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

tidy produces a data.frame with one row per choice of lambda, with columns

lambda	penalty parameter lambda
estimate	estimate (median) of mean-squared error or other criterion
std.error	standard error of criterion
conf.high	high end of confidence interval on criterion
conf.low	low end of confidence interval on criterion
nzero	number of parameters that are zero at this choice of lambda

glance returns a one-row data.frame with the values

nulldev	null deviance
npasses	total passes over the data across all lambda values

Examples

```
if (require("glmnet", quietly = TRUE)) {
    set.seed(2014)

    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)

    head(tidy(cvfit1))
    glance(cvfit1)

    library(ggplot2)
    tidied_cv <- tidy(cvfit1)
    glance_cv <- glance(cvfit1)
```

```
    # plot of MSE as a function of lambda
    g <- ggplot(tidied_cv, aes(lambda, estimate)) + geom_line() + scale_x_log10()
    g

    # plot of MSE as a function of lambda with confidence ribbon
    g <- g + geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)
    g

    # plot of MSE as a function of lambda with confidence ribbon and choices
    # of minimum lambda marked
    g <- g + geom_vline(xintercept = glance_cv$lambda.min) +
        geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
    g

    # plot of number of zeros for each choice of lambda
    ggplot(tidied_cv, aes(lambda, nzero)) + geom_line() + scale_x_log10()

    # coefficient plot with min lambda shown
    tidied <- tidy(cvfit1$glmnet.fit)
    ggplot(tidied, aes(lambda, estimate, group = term)) + scale_x_log10() +
        geom_line() +
        geom_vline(xintercept = glance_cv$lambda.min) +
        geom_vline(xintercept = glance_cv$lambda.1se, lty = 2)
}
```

`data.frame_tidiers` *Tidiers for data.frame objects*

Description

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.

Usage

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

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

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

Arguments

x	A data.frame
...	extra arguments: for tidy, these are passed on to describe from psych package
data	data, not used

Details

The tidy method calls the psych method describe directly to produce its per-columns summary statistics.

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
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

See Also

describe

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()
```

`decompose_tidiers` *Tidying methods for seasonal decompositions*

Description

These tidiers provide an `augment` method for the results of a seasonal decomposition with `decompose` or `stl`.

Usage

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

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

Arguments

`x`	An object of class `"stl"` or `"decomposed.ts"`, resulting from a call to `decompose` or `stl`.
`...`	Extra arguments. Unused.
`weights`	Whether to include the robust weights in the output.

Details

The `augment` method returns the computed seasonal and trend components, as well as the "remainder" term and the seasonally adjusted (or "deseasonalised") series.

Value

The `augment` method returns a tidy data frame with the following columns:

`.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.

Author(s)

Aaron Jacobs

See Also

`decompose`, `stl`

Examples

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

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

# Compare the original series to its decompositions.

cbind(broom::tidy(nottem), broom::augment(d1),
      broom::augment(d2))

# Visually compare seasonal decompositions in tidy data frames.

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

decomps <- tibble(
    # Turn the ts objects into data frames.
    series = list(broom::tidy(nottem), broom::tidy(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(series, augment) %>%
    mutate(index = 1:n()) %>%
    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))
```

`emmeans_tidiers` *Tidy estimated marginal means (least-squares means) objects from the emmeans and lsmeans packages*

Description

Tidiers for estimated marginal means objects, which report the predicted means for factors or factor combinations in a linear model. This covers three classes: `emmGrid`, `lsmobj`, and `ref.grid`. (The first class is from the `emmeans` package, and is the successor to the latter two classes, which have slightly different purposes within the `lsmeans` package but have similar output).

Usage

```
## S3 method for class 'lsmobj'
tidy(x, conf.level = 0.95, ...)

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

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

Arguments

`x`	"emmGrid", lsmobj", or "ref.grid" object
`conf.level`	Level of confidence interval, used only for `emmGrid` and `lsmobj` objects
`...`	Extra arguments, passed on to summary.emmGrid or summary.ref.grid

Details

There are a large number of arguments that can be passed on to summary.emmGrid or summary.ref.grid. By broom convention, we use `conf.level` to pass the `level` argument.

Value

A data frame with one observation for each estimated mean, and one column for each combination of factors, along with the following variables:

`estimate`	Estimated least-squares mean
`std.error`	Standard error of estimate
`df`	Degrees of freedom
`conf.low`	Lower bound of confidence interval
`conf.high`	Upper bound of confidence interval

When the input is a contrast, each row will contain one estimated contrast, along with some of the following columns:

`level1`	One level of the factor being contrasted
`level2`	Second level
`contrast`	In cases where the contrast is not made up of two levels, describes each
`statistic`	T-ratio statistic
`p.value`	P-value

Examples

```
if (require("emmeans", quietly = TRUE)) {
  # 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)
  head(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), 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), aes(price2, estimate, color = day)) +
    geom_line() +
    geom_errorbar(aes(ymin = conf.low, ymax = conf.high))
}
```

`ergm_tidiers` *Tidying methods for an exponential random graph model*

Description

These methods tidy the coefficients of an exponential random graph model estimated with the **ergm** package into a summary, and construct a one-row glance of the model's statistics. 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

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

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

Arguments

`x`	an **ergm** object
`conf.int`	whether to include a confidence interval
`conf.level`	confidence level of the interval, used only if `conf.int=TRUE`
`exponentiate`	whether to exponentiate the coefficient estimates and confidence intervals
`quick`	whether to compute a smaller and faster version, containing only the `term` and `estimate` columns.
`...`	extra arguments passed to `summary.ergm`
`deviance`	whether to report null and residual deviance for the model, along with degrees of freedom; defaults to `FALSE`
`mcmc`	whether to report MCMC interval, burn-in and sample size used to estimate the model; defaults to `FALSE`

Details

There is no `augment` method for **ergm** objects.

Value

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

`tidy.ergm` returns one row for each coefficient, with five 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

If `conf.int=TRUE`, it also includes columns for `conf.low` and `conf.high`.

`glance.ergm` returns a one-row data.frame with the columns

`independence`	Whether the model assumed dyadic independence
`iterations`	The number of iterations performed before convergence
`logLik`	If applicable, the log-likelihood associated with the model
`AIC`	The Akaike Information Criterion
`BIC`	The Bayesian Information Criterion

If `deviance=TRUE`, and if the model supports it, the data frame will also contain the columns

`null.deviance`	The null deviance of the model
`df.null`	The degrees of freedom of the null deviance
`residual.deviance`	The residual deviance of the model
`df.residual`	The degrees of freedom of the residual deviance

Last, if `mcmc=TRUE`, the data frame will also contain the columns

`MCMC.interval`	The interval used during MCMC estimation
`MCMC.burnin`	The burn-in period of the MCMC estimation
`MCMC.samplesize`	The sample size used during MCMC estimation

References

Hunter DR, Handcock MS, Butts CT, Goodreau SM, Morris M (2008b). **ergm**: A Package to Fit, Simulate and Diagnose Exponential-Family Models for Networks. *Journal of Statistical Software*, 24(3). http://www.jstatsoft.org/v24/i03/.

See Also

`ergm`, `control.ergm`, `summary.ergm`

Examples

```
if (require("ergm")) {
    # Using the same example as the ergm package
    # Load the Florentine marriage network data
    data(florentine)

    # Fit a model where the propensity to form ties between
    # families depends on the absolute difference in wealth
    gest <- ergm(flomarriage ~ edges + absdiff("wealth"))

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

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

    # Take a look at likelihood measures and other
    # control parameters used during MCMC estimation
    glance(gest)
    glance(gest, deviance = TRUE)
    glance(gest, mcmc = TRUE)
}
```

`felm_tidiers` *Tidying methods for models with multiple group fixed effects*

Description

These methods tidy the coefficients of a linear model with multiple group fixed effects

Usage

```
## S3 method for class 'felm'
tidy(x, conf.int = FALSE, conf.level = 0.95, fe = FALSE,
  fe.error = fe, ...)

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

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

Arguments

`x`	felm object
`conf.int`	whether to include a confidence interval
`conf.level`	confidence level of the interval, used only if `conf.int=TRUE`

fe	whether to include estimates of fixed effects
fe.error	whether to include standard error of fixed effects
...	extra arguments (not used)
data	Original data, defaults to extracting it from the model

Details

If conf.int=TRUE, the confidence interval is computed

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

tidy.felm returns one row for each coefficient. If fe=TRUE, it also includes rows for for fixed effects estimates. There are five columns:

term	The term in the linear model being estimated and tested
estimate	The estimated coefficient
std.error	The standard error from the linear model
statistic	t-statistic
p.value	two-sided p-value

If cont.int=TRUE, it also includes columns for conf.low and conf.high, computed with confint.

augment.felm returns one row for each observation, with multiple columns added to the original data:

.fitted	Fitted values of model
.resid	Residuals

If fixed effect are present,

.comp	Connected component
.fe_	Fixed effects (as many columns as factors)

glance.lm returns a one-row data.frame with the columns

r.squared	The percent of variance explained by the model
adj.r.squared	r.squared adjusted based on the degrees of freedom
sigma	The square root of the estimated residual variance
statistic	F-statistic
p.value	p-value from the F test
df	Degrees of freedom used by the coefficients
df.residual	residual degrees of freedom

Examples

```
if (require("lfe", quietly = TRUE)) {
    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)
    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)
}
```

---

finish_glance *Add logLik, AIC, BIC, and other common measurements to a glance of a prediction*

Description

A helper function for several functions in the glance generic. Methods such as logLik, AIC, and BIC are defined for many prediction objects, such as lm, glm, and nls. This is a helper function that adds them to a glance data.frame can be performed. If any of them cannot be computed, it fails quietly.

Usage

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

Arguments

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

Details

In one special case, deviance for objects of the `lmerMod` class from lme4 is computed with `deviance(x, REML=FALSE)`.

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

Each of these are produced by the corresponding generics

`fitdistr_tidiers` *Tidying methods for fitdistr objects from the MASS package*

Description

These methods tidies the parameter estimates resulting from an estimation of a univariate distribution's parameters.

Usage

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

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

Arguments

`x`	An object of class "fitdistr"
`...`	extra arguments (not used)

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy.fitdistr` returns one row for each parameter that was estimated, with columns:

`term`	The term that was estimated
`estimate`	Estimated value
`std.error`	Standard error of estimate

glance.fitdistr returns a one-row data.frame with the columns

n	Number of observations used in estimation
logLik	log-likelihood of estimated data
AIC	Akaike Information Criterion
BIC	Bayesian Information Criterion

Examples

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

library(MASS)
fit <- fitdistr(x, dnorm, list(mean = 3, sd = 1))

tidy(fit)
glance(fit)
```

---

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

Description

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

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

`gamlss_tidiers` *Tidying methods for gamlss objects*

Description

Tidying methods for "gamlss" objects from the gamlss package.

Usage

```
## S3 method for class 'gamlss'
tidy(x, quick = FALSE, ...)
```

Arguments

`x`	A "gamlss" object
`quick`	Whether to perform a fast version, and return only the coefficients
`...`	Extra arguments (not used)

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

A data.frame with one row for each coefficient, containing columns

`parameter`	Type of coefficient being estimated: `mu`, `sigma`, `nu`, or `tau`
`term`	The term in the model being estimated and tested
`estimate`	The estimated coefficient
`std.error`	The standard error from the linear model
`statistic`	t-statistic
`p.value`	two-sided p-value

if (requireNamespace("gamlss", quietly = TRUE)) data(abdom) mod<-gamlss(y~pb(x),sigma.fo=~pb(x),family=BCT, data=abdom, method=mixed(1,20))

tidy(mod)

gam_tidiers *Tidying methods for a generalized additive model (gam)*

Description

These methods tidy the coefficients of a "gam" object (generalized additive model) into a summary, augment the original data with information on the fitted values and residuals, and construct a one-row glance of the model's statistics.

Usage

```
## S3 method for class 'gam'
tidy(x, parametric = FALSE, ...)

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

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

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

Arguments

x	gam or Gam object
parametric	logical. Return parametric coefficients (TRUE) or information about smooth terms (FALSE)?
...	extra arguments (not used)

Details

The "augment" method is handled by lm_tidiers.

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy.gam` called on an object from the gam package, or an object from the mgcv package with `parametric = FALSE`, returns the tidied output of the parametric ANOVA with one row for each term in the formula. The columns match those in anova_tidiers. `tidy.gam` called on a gam object from the mgcv package with `parametric = TRUE` returns the fixed coefficients.

`glance.gam` returns a one-row data.frame with the columns

df	Degrees of freedom used by the coefficients
logLik	the data's log-likelihood under the model

AIC	the Akaike Information Criterion
BIC	the Bayesian Information Criterion
deviance	deviance
df.residual	residual degrees of freedom

See Also

lm_tidiers, anova_tidiers

Examples

```
if (require("gam", quietly = TRUE)) {
    data(kyphosis)
    g <- gam(Kyphosis ~ s(Age,4) + Number, family = binomial, data = kyphosis)
    tidy(g)
    augment(g)
    glance(g)
}
```

---

geeglm_tidiers *Tidying methods for generalized estimating equations models*

Description

These methods tidy the coefficients of generalized estimating equations models of the geeglm class from functions of the geepack package.

Usage

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

Arguments

x	An object of class geeglm, such as from geeglm
conf.int	whether to include a confidence interval
conf.level	confidence level of the interval, used only if conf.int=TRUE
exponentiate	whether to exponentiate the coefficient estimates and confidence intervals (typical for log distributions)
quick	whether to compute a smaller and faster version, containing only the term and estimate columns.
...	Additional arguments to be passed to other methods. Currently not used.

Details

If conf.int=TRUE, the confidence interval is computed with the confint.geeglm function.

While tidy is supported for "geeglm" objects, augment and glance are not.

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

All tidying methods return a data.frame without rownames. The structure depends on the method chosen.

tidy.geeglm returns one row for each coefficient, with five columns:

term	The term in the linear model being estimated and tested
estimate	The estimated coefficient
std.error	The standard error from the GEE model
statistic	Wald statistic
p.value	two-sided p-value

If conf.int=TRUE, it also includes columns for conf.low and conf.high, computed with confint.geeglm (included as part of broom).

Examples

```
if (require('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, quick = TRUE)
    tidy(geefit, conf.int = TRUE)
}
```

---

glance *Construct a single row summary "glance" of a model, fit, or other object*

Description

glance methods always return either a one-row data frame (except on NULL, which returns an empty data frame)

Usage

```
glance(x, ...)
```

Arguments

x	model or other R object to convert to single-row data frame
...	other arguments passed to methods

---

glmnet_tidiers *Tidiers for LASSO or elasticnet regularized fits*

Description

Tidying methods for regularized fits produced by `glmnet`, summarizing the estimates across values of the penalty parameter lambda.

Usage

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

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

Arguments

x	a "glmnet" object
...	extra arguments (not used)

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 extremely 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

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` produces a data.frame with one row per combination of coefficient (including the intercept) and value of lambda for which the estimate is nonzero, with the columns:

`term`	coefficient name (V1...VN by default, along with "(Intercept)")
`step`	which step of lambda choices was used

estimate	estimate of coefficient
lambda	value of penalty parameter lambda
dev.ratio	fraction of null deviance explained at each value of lambda

glance returns a one-row data.frame with the values

nulldev	null deviance
npasses	total passes over the data across all lambda values

Examples

```
if (require("glmnet", quietly = TRUE)) {
    set.seed(2014)
    x <- matrix(rnorm(100*20),100,20)
    y <- rnorm(100)
    fit1 <- glmnet(x,y)

    head(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")
    head(tidy(fit2))
}
```

---

glm_tidiers *Tidying methods for a glm object*

Description

Tidy a glm object. The tidy and augment methods are handled by lm_tidiers.

Usage

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

Arguments

x	glm object
...	extra arguments, not used

Value

tidy and augment return the same values as do tidy.lm and augment.lm.

glance returns a one-row data.frame with the columns

null.deviance	the deviance of the null model
df.null	the residual degrees of freedom for the null model
logLik	the data's log-likelihood under the model
AIC	the Akaike Information Criterion
BIC	the Bayesian Information Criterion
deviance	deviance
df.residual	residual degrees of freedom

See Also

tidy.lm and augment.lm. Also glm, which computes the values reported by the glance method.

Examples

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

---

gmm_tidiers *Tidying methods for generalized method of moments "gmm" objects*

Description

These methods tidy the coefficients of "gmm" objects from the gmm package, or glance at the model-wide statistics (especially the J-test).

Usage

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

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

Arguments

x	gmm object
conf.int	whether to include a confidence interval
conf.level	confidence level of the interval, used only if conf.int=TRUE
exponentiate	whether to exponentiate the coefficient estimates and confidence intervals (typical for logistic regression)
quick	whether to compute a smaller and faster version, containing only the term and estimate columns (and confidence interval if requested, which may be slower)
...	extra arguments (not used)

Details

If conf.int=TRUE, the confidence interval is computed with the confint function.

Note that though the "gmm" object contains residuals and fitted values, there is not yet an augment method implemented. This is because the input to gmm is not tidy (it's a "wide" matrix), so it is not immediately clear what the augmented results should look like.

Value

All tidying methods return a data.frame without rownames. The structure depends on the method chosen.

tidy.gmm returns one row for each coefficient, with six columns:

term	The term in the model being estimated
estimate	The estimated coefficient
std.error	The standard error from the linear model
statistic	t-statistic
p.value	two-sided p-value

If all the the terms have _ in them (e.g. WMK_(Intercept)), they are split into variable and term.

If conf.int=TRUE, it also includes columns for conf.low and conf.high, computed with confint.

glance.gmm returns a one-row data.frame with the columns

df	Degrees of freedom
statistic	Statistic from J-test for E(g)=0
p.value	P-value from J-test
df.residual	Residual degrees of freedom, if included in "gmm" object

Examples

```
if (require("gmm", quietly = TRUE)) {
  # 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(variable, estimate)) %>%
    ggplot(aes(estimate, variable)) +
    geom_point() +
    geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
    facet_wrap(~ term) +
    geom_vline(xintercept = 0, color = "red", lty = 2)

  # from a function instead of a matrix
  g <- function(theta, x) {
   e <- x[,2:11] - theta[1] - (x[,1] - theta[1]) %*% matrix(theta[2:11], 1, 10)
   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(variable, estimate)) %>%
    ggplot(aes(estimate, variable)) +
    geom_point() +
    geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
    facet_wrap(~ term) +
    geom_vline(xintercept = 0, color = "red", lty = 2)
}
```

`htest_tidiers` *Tidying methods for an htest object*

Description

Tidies hypothesis test objects, such as those from `cor.test`, `t.test`, and `wilcox.test`, into a one-row data frame.

Usage

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

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

Arguments

`x`	An object of class `"htest"`
`...`	extra arguments (not used)

Details

No `augment` method is provided for `"htest"`, since there is no sense in which a hypothesis test generates one value for each observation.

Value

Both `tidy` and `glance` return the same output, a one-row data frame with one or more of the following columns:

`estimate`	Estimate of the effect size
`statistic`	Test statistic used to compute the p-value
`p.value`	P-value

parameter	Parameter field in the htest, typically degrees of freedom
conf.low	Lower bound on a confidence interval
conf.high	Upper bound on a confidence interval
estimate1	Sometimes two estimates are computed, such as in a two-sample t-test
estimate2	Sometimes two estimates are computed, such as in a two-sample t-test
method	Method used to compute the statistic as a string
alternative	Alternative hypothesis as a string

Which columns are included depends on the hypothesis test used.

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)
tidy(wt)

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

---

inflate *Expand a dataset to include all factorial combinations of one or more variables*

Description

This function is deprecated: use `tidyr::crossing` instead

Usage

```
inflate(df, ..., stringsAsFactors = FALSE)
```

Arguments

df	a tbl
...	arguments
stringsAsFactors	logical specifying if character vectors are converted to factors.

Value

A tbl

`insert_NAs` *insert a row of NAs into a data frame wherever another data frame has NAs*

Description

insert a row of NAs into a data frame wherever another data frame has NAs

Usage

```
insert_NAs(x, original)
```

Arguments

`x`	data frame that has one row for each non-NA row in original
`original`	data frame with NAs

`ivreg_tidiers` *Tidiers for ivreg models*

Description

Tidiers for ivreg models

Usage

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

## S3 method for class 'ivreg'
augment(x, data = as.data.frame(stats::model.frame(x)),
  newdata, ...)

## S3 method for class 'ivreg'
glance(x, diagnostics = FALSE, ...)
```

Arguments

`x`	An "ivreg" object
`conf.int`	Whether to include a confidence interval
`conf.level`	Confidence level of the interval, used only if `conf.int=TRUE`
`exponentiate`	Whether to exponentiate the coefficient estimates and confidence intervals
`...`	extra arguments, not used

data	Original dataset
newdata	New data to make predictions from (optional)
diagnostics	Logical. Return results of diagnostic tests.

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

tidy.ivreg returns a data frame with one row per coefficient, of the same form as tidy.lm.

augment returns a data frame with one row for each initial observation, adding the columns:

.fitted	predicted (fitted) values

and if newdata is NULL:

.resid	residuals

glance returns a one-row data frame with columns

r.squared	The percent of variance explained by the model
adj.r.squared	r.squared adjusted based on the degrees of freedom
statistic	Wald test statistic
p.value	p-value from the Wald test
df	Degrees of freedom used by the coefficients
sigma	The square root of the estimated residual variance
df.residual	residual degrees of freedom

If diagnostics is TRUE, glance also returns:

p.value.Sargan	P value of Sargan test
p.value.Wu.Hausman	P value of Wu-Hausman test
p.value.weakinst	P value of weak instruments test

See Also

lm_tidiers

Examples

```
if (require("AER", quietly = TRUE)) {
    data("CigarettesSW", package = "AER")
    CigarettesSW$rprice <- with(CigarettesSW, price/cpi)
    CigarettesSW$rincome <- with(CigarettesSW, income/population/cpi)
    CigarettesSW$tdiff <- with(CigarettesSW, (taxs - tax)/cpi)
   ivr <- ivreg(log(packs) ~ log(rprice) + log(rincome) | log(rincome) + tdiff + I(tax/cpi),
          data = CigarettesSW, subset = year == "1995")
```

```
    summary(ivr)

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

    head(augment(ivr))

    glance(ivr)
}
```

---

kappa_tidiers *Tidy a kappa object from a Cohen's kappa calculation*

Description

Tidy a "kappa" object, from the cohen.kappa function in the psych package. This represents the agreement of two raters when using nominal scores.

Usage

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

Arguments

x	An object of class "kappa"
...	extra arguments (not used)

Details

Note that the alpha of the confidence interval is determined when the cohen.kappa function is originally run.

Value

A data.frame with columns

type	Either "weighted" or "unweighted"
estimate	The estimated value of kappa with this method
conf.low	Lower bound of confidence interval
conf.high	Upper bound of confidence interval

See Also

cohen.kappa

Examples

```
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))
```

---

kde_tidiers *Tidy a kernel density estimate object from the ks package*

Description

Tidy a kernel density estimate object, into a table with one row for each point in the estimated grid, and one column for each dimension (along with an `estimate` column with the estimated density).

Usage

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

Arguments

`x`	A "ks" object from the kde package
`...`	Extra arguments, not used

Value

A data frame with one row for each point in the estimated grid. The result contains one column (named `x1`, `x2`, etc) for each dimension, and an `estimate` column containing the estimated density.

Examples

```
if (require("ks", quietly = TRUE)) {
  dat <- replicate(2, rnorm(100))
  k <- kde(dat)

  td <- tidy(k)
  head(td)
```

```
  library(ggplot2)
  ggplot(td, aes(x1, x2, fill = estimate)) +
    geom_tile() +
    theme_void()

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

  td3 <- tidy(k3)
  head(td3)
}
```

`kmeans_tidiers` *Tidying methods for kmeans objects*

Description

These methods summarize the results of k-means clustering into three tidy forms. `tidy` describes the center and size of each cluster, `augment` adds the cluster assignments to the original data, and `glance` summarizes the total within and between sum of squares of the clustering.

Usage

```
## S3 method for class 'kmeans'
tidy(x, col.names = paste0("x", 1:ncol(x$centers)), ...)

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

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

Arguments

`x`	kmeans object
`col.names`	The names to call each dimension of the data in `tidy`. Defaults to `x1, x2...`
`...`	extra arguments, not used
`data`	Original data (required for `augment`)

Value

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

`tidy` returns one row per cluster, with one column for each dimension in the data describing the center, followed by

size	The size of each cluster
withinss	The within-cluster sum of squares
cluster	A factor describing the cluster from 1:k

augment returns the original data with one extra column:

.cluster	The cluster assigned by the k-means algorithm

glance returns a one-row data.frame with the columns

totss	The total sum of squares
tot.withinss	The total within-cluster sum of squares
betweenss	The total between-cluster sum of squares
iter	The numbr of (outer) iterations

See Also

kmeans

Examples

```
library(dplyr)
library(ggplot2)

set.seed(2014)
centers <- data.frame(cluster=factor(1:3), size=c(100, 150, 50),
                      x1=c(5, 0, -3), x2=c(-1, 1, -2))
points <- centers %>% group_by(cluster) %>%
 do(data.frame(x1=rnorm(.$size[1], .$x1[1]),
               x2=rnorm(.$size[1], .$x2[1])))

k <- kmeans(points %>% dplyr::select(x1, x2), 3)
tidy(k)
head(augment(k, points))
glance(k)

ggplot(augment(k, points), aes(x1, x2)) +
    geom_point(aes(color = .cluster)) +
    geom_text(aes(label = cluster), data = tidy(k), size = 10)
```

list_tidiers *Tidiers for return values from functions that aren't S3 objects*

Description

This method handles the return values of functions that return lists rather than S3 objects, such as `optim`, `svd`, or `interp`, and therefore cannot be handled by S3 dispatch.

Usage

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

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

Arguments

`x`	list object
`...`	extra arguments, passed to the tidying function

Details

Those tiders themselves are implemented as functions of the form tidy_<function> or glance_<function> that are not exported.

See Also

optim_tidiers, xyz_tidiers, svd_tidiers, orcutt_tidiers

lme4_tidiers *Tidying methods for mixed effects models*

Description

These methods tidy the coefficients of mixed effects models, particularly responses of the `merMod` class

Usage

```
## S3 method for class 'merMod'
tidy(x, effects = c("ran_pars", "fixed"), scales = NULL,
  ran_prefix = NULL, conf.int = FALSE, conf.level = 0.95,
  conf.method = "Wald", ...)

## S3 method for class 'merMod'
augment(x, data = stats::model.frame(x), newdata, ...)

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

Arguments

x	An object of class merMod, such as those from lmer, glmer, or nlmer
effects	A character vector including one or more of "fixed" (fixed-effect parameters), "ran_pars" (variances and covariances or standard deviations and correlations of random effect terms) or "ran_modes" (conditional modes/BLUPs/latent variable estimates)
scales	scales on which to report the variables: for random effects, the choices are '"sdcor"' (standard deviations and correlations: the default if scales is NULL) or '"vcov"' (variances and covariances). NA means no transformation, appropriate e.g. for fixed effects; inverse-link transformations (exponentiation or logistic) are not yet implemented, but may be in the future.
ran_prefix	a length-2 character vector specifying the strings to use as prefixes for self- (variance/standard deviation) and cross- (covariance/correlation) random effects terms
conf.int	whether to include a confidence interval
conf.level	confidence level for CI
conf.method	method for computing confidence intervals (see lme4::confint.merMod)
...	extra arguments (not used)
data	original data this was fitted on; if not given this will attempt to be reconstructed
newdata	new data to be used for prediction; optional

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

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

`tidy` returns one row for each estimated effect, either with groups depending on the `effects` parameter. It contains the columns

`group`	the group within which the random effect is being estimated: `"fixed"` for fixed effects
`level`	level within group (`NA` except for modes)
`term`	term being estimated
`estimate`	estimated coefficient
`std.error`	standard error
`statistic`	t- or Z-statistic (`NA` for modes)
`p.value`	P-value computed from t-statistic (may be missing/NA)

`augment` returns one row for each original observation, with columns (each prepended by a .) added. Included are the columns

`.fitted`	predicted values
`.resid`	residuals
`.fixed`	predicted values with no random effects

Also added for "merMod" objects, but not for "mer" objects, are values from the response object within the model (of type `lmResp`, `glmResp`, `nlsResp`, etc). These include `".mu"`,`".offset"`, `".sqrtXwt"`, `".sqrtrwt"`,

`glance` returns one row with the columns

`sigma`	the square root of the estimated residual variance
`logLik`	the data's log-likelihood under the model
`AIC`	the Akaike Information Criterion
`BIC`	the Bayesian Information Criterion
`deviance`	deviance

See Also

na.action

Examples

```
if (require("lme4")) {
    # example regressions are from lme4 documentation
    lmm1 <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
    tidy(lmm1)
    tidy(lmm1, effects = "fixed")
    tidy(lmm1, effects = "fixed", conf.int=TRUE)
    tidy(lmm1, effects = "fixed", conf.int=TRUE, conf.method="profile")
    tidy(lmm1, effects = "ran_modes", conf.int=TRUE)
```

```
    head(augment(lmm1, sleepstudy))
    glance(lmm1)

    glmm1 <- glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
                   data = cbpp, family = binomial)
    tidy(glmm1)
    tidy(glmm1, effects = "fixed")
    head(augment(glmm1, cbpp))
    glance(glmm1)

    startvec <- c(Asym = 200, xmid = 725, scal = 350)
    nm1 <- nlmer(circumference ~ SSlogis(age, Asym, xmid, scal) ~ Asym|Tree,
                   Orange, start = startvec)
    tidy(nm1)
    tidy(nm1, effects = "fixed")
    head(augment(nm1, Orange))
    glance(nm1)
}
```

`lmodel2_tidiers` *Tidiers for linear model II objects from the lmodel2 package*

Description

Tidy or glance an lmodel2 object. An lmodel2 represents model II simple linear regression, where both variables in the regression equation are random.

Usage

```
## S3 method for class 'lmodel2'
tidy(x, ...)

## S3 method for class 'lmodel2'
glance(x, ...)
```

Arguments

`x`	lmodel2 object
`...`	Extra arguments, not used

Details

Note that unlike linear regression, there are always only two terms in an lmodel2: Intercept and Slope. Furthermore, these are computed by four methods: OLS (ordinary least squares), MA (major axis), SMA (standard major axis), and RMA (ranged major axis). See the lmodel2 documentation for more.

Note that there is no `augment` method for lmodel2 objects because lmodel2 does not provide a `predict` or `residuals` method (and since when both observations are random, fitted values and residuals have a less clear meaning).

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns a data frame with one row for each combination of method (OLS/MA/SMA/RMA) and term (always Intercept/Slope). Its columns are:

method Either OLS/MA/SMA/RMA

term Either "Intercept" or "Slope"

estimate Estimated coefficient

conf.low Lower bound of 95% confidence interval

conf.high Upper bound of 95% confidence interval

`glance` returns a one-row data frame with columns

r.squared OLS R-squared

p.value OLS parametric p-value

theta Angle between OLS lines `lm(y ~ x)` and `lm(x ~ y)`

H H statistic for computing confidence interval of major axis slope

Examples

```
if (require("lmodel2", quietly = TRUE)) {
  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))
}
```

`lm_tidiers` *Tidying methods for a linear model*

Description

These methods tidy the coefficients of a linear model into a summary, augment the original data with information on the fitted values and residuals, and construct a one-row glance of the model's statistics.

Usage

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

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

## S3 method for class 'lm'
augment(x, data = stats::model.frame(x), newdata, type.predict,
  type.residuals, ...)

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

## S3 method for class 'summary.lm'
glance(x, ...)
```

Arguments

x	lm object
conf.int	whether to include a confidence interval
conf.level	confidence level of the interval, used only if conf.int=TRUE
exponentiate	whether to exponentiate the coefficient estimates and confidence intervals (typical for logistic regression)
quick	whether to compute a smaller and faster version, containing only the term and estimate columns.
...	extra arguments (not used)
data	Original data, defaults to the extracting it from the model
newdata	If provided, performs predictions on the new data
type.predict	Type of prediction to compute for a GLM; passed on to predict.glm
type.residuals	Type of residuals to compute for a GLM; passed on to residuals.glm

Details

If you have missing values in your model data, you may need to refit the model with na.action = na.exclude.

If conf.int=TRUE, the confidence interval is computed with the confint function.

While tidy is supported for "mlm" objects, augment and glance are not.

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.

Code and documentation for augment.lm originated in the ggplot2 package, where it was called fortify.lm

Value

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

`tidy.lm` returns one row for each coefficient, with five columns:

`term`	The term in the linear model being estimated and tested
`estimate`	The estimated coefficient
`std.error`	The standard error from the linear model
`statistic`	t-statistic
`p.value`	two-sided p-value

If the linear model is an "mlm" object (multiple linear model), there is an additional column:

`response`	Which response column the coefficients correspond to (typically Y1, Y2, etc)

If `conf.int=TRUE`, it also includes columns for `conf.low` and `conf.high`, computed with `confint`.

When `newdata` is not supplied `augment.lm` returns one row for each observation, with seven columns added to the original data:

`.hat`	Diagonal of the hat matrix
`.sigma`	Estimate of residual standard deviation when corresponding observation is dropped from model
`.cooksd`	Cooks distance, `cooks.distance`
`.fitted`	Fitted values of model
`.se.fit`	Standard errors of fitted values
`.resid`	Residuals
`.std.resid`	Standardised residuals

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

When `newdata` is supplied, `augment.lm` returns one row for each observation, with three columns added to the new data:

`.fitted`	Fitted values of model
`.se.fit`	Standard errors of fitted values
`.resid`	Residuals of fitted values on the new data

`glance.lm` returns a one-row data.frame with the columns

`r.squared`	The percent of variance explained by the model
`adj.r.squared`	r.squared adjusted based on the degrees of freedom
`sigma`	The square root of the estimated residual variance
`statistic`	F-statistic
`p.value`	p-value from the F test, describing whether the full regression is significant
`df`	Degrees of freedom used by the coefficients

logLik	the data's log-likelihood under the model
AIC	the Akaike Information Criterion
BIC	the Bayesian Information Criterion
deviance	deviance
df.residual	residual degrees of freedom

See Also

summary.lm

na.action

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()

head(augment(mod))
head(augment(mod, mtcars))

# predict on new data
newdata <- mtcars %>% head(6) %>% mutate(wt = wt + 1)
augment(mod, newdata = newdata)

au <- augment(mod, data = mtcars)

plot(mod, which = 1)
qplot(.fitted, .resid, data = au) +
  geom_hline(yintercept = 0) +
  geom_smooth(se = FALSE)
qplot(.fitted, .std.resid, data = au) +
  geom_hline(yintercept = 0) +
  geom_smooth(se = FALSE)
qplot(.fitted, .std.resid, data = au,
  colour = factor(cyl))
qplot(mpg, .std.resid, data = au, colour = factor(cyl))

plot(mod, which = 2)
```

```
qplot(sample =.std.resid, data = au, stat = "qq") +
    geom_abline()

plot(mod, which = 3)
qplot(.fitted, sqrt(abs(.std.resid)), data = au) + geom_smooth(se = FALSE)

plot(mod, which = 4)
qplot(seq_along(.cooksd), .cooksd, data = au)

plot(mod, which = 5)
qplot(.hat, .std.resid, data = au) + geom_smooth(se = FALSE)
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)

qplot(.hat, .std.resid, data = au, size = .cooksd) +
  geom_smooth(se = FALSE, size = 0.5)

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()
qplot(.hat, .cooksd, size = .cooksd / .hat, data = au) + scale_size_area()

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

`loess_tidiers` *Augmenting methods for loess models*

Description

This method augments the original data with information on the fitted values and residuals, and optionally the standard errors.

Usage

```
## S3 method for class 'loess'
augment(x, data = stats::model.frame(x), newdata, ...)
```

Arguments

`x`	A "loess" object
`data`	Original data, defaults to the extracting it from the model

newdata	If provided, performs predictions on the new data
...	extra arguments

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

When `newdata` is not supplied `augment.loess` returns one row for each observation with three columns added to the original data:

.fitted	Fitted values of model
.se.fit	Standard errors of the fitted values
.resid	Residuals of the fitted values

When `newdata` is supplied `augment.loess` returns one row for each observation with one additional column:

.fitted	Fitted values of model
.se.fit	Standard errors of the fitted values

See Also

na.action

Examples

```
lo <- loess(mpg ~ wt, mtcars)
augment(lo)

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

# with a new dataset
augment(lo, newdata = head(mtcars))
```

matrix_tidiers *Tidiers for matrix objects*

Description

These perform tidying operations on matrix objects. `tidy` turns the matrix into a data.frame while bringing rownames, if they exist, in as a column called `.rownames` (since results of tidying operations never contain rownames). `glance` simply reports the number of rows and columns. Note that no augment method exists for matrices.

Usage

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

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

Arguments

`x`	A matrix
`...`	extra arguments, not used

Value

`tidy.matrix` returns the original matrix converted into a data.frame, except that it incorporates rownames (if they exist) into a column called `.rownames`.

`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

Examples

```
mat <- as.matrix(mtcars)
tidy(mat)
glance(mat)
```

`mclust_tidiers` *Tidying methods for Mclust objects*

Description

These methods summarize the results of Mclust clustering into three tidy forms. `tidy` describes the size, mixing probability, mean and variabilty of each class, `augment` adds the class assignments and their probabilities to the original data, and `glance` summarizes the model parameters of the clustering.

Usage

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

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

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

Arguments

`x`	Mclust object
`...`	extra arguments, not used
`data`	Original data (required for `augment`)

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns one row per component, with

`component`	A factor describing the cluster from 1:k (or 0:k in presence of a noise term in x)
`size`	The size of each component
`proportion`	The mixing proportion of each component
`variance`	In case of one-dimensional and spherical models, the variance for each component, omitted otherwise. NA for noise component
`mean`	The mean for each component. In case of two- or more dimensional models, a column with the mean is added for each dimension. NA for noise component

`augment` returns the original data with two extra columns:

`.class`	The class assigned by the Mclust algorithm
`.uncertainty`	The uncertainty associated with the classification

glance returns a one-row data.frame with the columns

model	A character string denoting the model at which the optimal BIC occurs
n	The number of observations in the data
G	The optimal number of mixture components
BIC	The optimal BIC value
logLik	The log-likelihood corresponding to the optimal BIC
df	The number of estimated parameters
hypvol	The hypervolume parameter for the noise component if required, otherwise set to NA

See Also

Mclust

Examples

```
library(dplyr)
library(ggplot2)
library(mclust)

set.seed(2016)
centers <- data.frame(cluster=factor(1:3), size=c(100, 150, 50),
                      x1=c(5, 0, -3), x2=c(-1, 1, -2))
points <- centers %>% group_by(cluster) %>%
 do(data.frame(x1=rnorm(.$size[1], .$x1[1]),
               x2=rnorm(.$size[1], .$x2[1]))) %>%
 ungroup()

m = Mclust(points %>% dplyr::select(x1, x2))

tidy(m)
head(augment(m, points))
glance(m)
```

---

mcmc_tidiers *Tidying methods for MCMC (Stan, JAGS, etc.) fits*

Description

Tidying methods for MCMC (Stan, JAGS, etc.) fits

Usage

```
tidyMCMC(x, pars, estimate.method = "mean", conf.int = FALSE,
  conf.level = 0.95, conf.method = "quantile", droppars = "lp__",
  rhat = FALSE, ess = FALSE, ...)

## S3 method for class 'rjags'
tidy(x, pars, estimate.method = "mean", conf.int = FALSE,
  conf.level = 0.95, conf.method = "quantile", ...)

## S3 method for class 'stanfit'
tidy(x, pars, estimate.method = "mean", conf.int = FALSE,
  conf.level = 0.95, conf.method = "quantile", droppars = "lp__",
  rhat = FALSE, ess = FALSE, ...)
```

Arguments

x	an object of class ‘"stanfit"’
pars	(character) specification of which parameters to include
estimate.method	method for computing point estimate ("mean" or median")
conf.int	(logical) include confidence interval?
conf.level	probability level for CI
conf.method	method for computing confidence intervals ("quantile" or "HPDinterval")
droppars	Parameters not to include in the output (such as log-probability information)
rhat, ess	(logical) include Rhat and/or effective sample size estimates?
...	unused

Examples

```
## Not run:

# Using example from "RStan Getting Started"
# https://github.com/stan-dev/rstan/wiki/RStan-Getting-Started

model_file <- system.file("extdata", "8schools.stan", package = "broom")

schools_dat <- list(J = 8,
                    y = c(28,  8, -3,  7, -1,  1, 18, 12),
                    sigma = c(15, 10, 16, 11,  9, 11, 10, 18))

if (requireNamespace("rstan", quietly = TRUE)) {
  set.seed(2015)
  rstan_example <- stan(file = model_file, data = schools_dat,
                        iter = 100, chains = 2)
}
```

```
## End(Not run)

if (requireNamespace("rstan", quietly = TRUE)) {
  # the object from the above code was saved as rstan_example.rda
  infile <- system.file("extdata", "rstan_example.rda", package = "broom")
  load(infile)

  tidy(rstan_example)
  tidy(rstan_example, conf.int = TRUE, pars = "theta")

  td_mean <- tidy(rstan_example, conf.int = TRUE)
  td_median <- tidy(rstan_example, conf.int = TRUE, estimate.method = "median")

  library(dplyr)
  library(ggplot2)
  tds <- rbind(mutate(td_mean, method = "mean"),
               mutate(td_median, method = "median"))

  ggplot(tds, aes(estimate, term)) +
    geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
    geom_point(aes(color = method))
}
```

`mle2_tidiers` *Tidy mle2 maximum likelihood objects*

Description

Tidy mle2 objects from the bbmle package.

Usage

```
## S3 method for class 'mle2'
tidy(x, conf.int = FALSE, conf.level = 0.95, ...)
```

Arguments

`x`	An "mle2" object
`conf.int`	Whether to add `conf.low` and `conf.high` columns
`conf.level`	Confidence level to use for interval
`...`	Extra arguments, not used

Examples

```
if (require("bbmle", quietly = TRUE)) {
  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)
}
```

`muhaz_tidiers` *Tidying methods for kernel based hazard rate estimates*

Description

These methods tidy the output of `muhaz` objects as returned by the `muhaz` function, which provides kernel based non-parametric hazard rate estimators.

Usage

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

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

Arguments

`x`	`muhaz` object
`...`	extra arguments (not used)

Details

The "augment" method is not useful and therefore not available for `muhaz` objects.

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy.muhaz` returns a tibble containing two columns: `time` at which the hazard rate was estimated and `estimate`.

`glance.muhaz` returns a one-row data.frame with the columns

`nobs`	Number of observations used for estimation

min.time	The minimum observed event or censoring time
max.time	The maximum observed event or censoring time
min.harzard	Minimal estimated hazard
max.hazard	Maximal estimated hazard

Examples

```
if (require("muhaz", quietly = TRUE)) {
  data(ovarian, package="survival")
  x <- muhaz(ovarian$futime, ovarian$fustat)
  tidy(x)
  glance(x)
}
```

---

multcomp_tidiers *tidying methods for objects produced by* **multcomp**

Description

These methods originated in ggplot2, as "fortify." In broom, they were renamed "tidy" because they summarize terms and tests, rather than adding columns to a dataset.

Usage

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

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

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

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

Arguments

x	an object of class glht, confint.glht, summary.glht or cld
...	extra arguments (not used)

Examples

```
if (require("multcomp") && require("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)
}
```

`multinom_tidiers` *Tidying methods for multinomial logistic regression models*

Description

These methods tidy the coefficients of multinomial logistic regression models generated by `multinom` of the `nnet` package.

Usage

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

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

Arguments

`x`	A model object of class `multinom`
`conf.int`	whether to include a confidence interval
`conf.level`	confidence level of the interval, used only if `conf.int=TRUE`
`exponentiate`	whether to exponentiate the coefficient estimates and confidence intervals (typical for multinomial logistic regression)
`...`	extra arguments, not used

Details

If `conf.int=TRUE`, the confidence interval is computed with the `confint` function.

While `tidy` and `glance` are supported for "multinom" objects, `augment` is not.

Value

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

`tidy.multinom` returns one row for each coefficient at each level of the response variable, with six columns:

`y.value`	The response level
`term`	The term in the model being estimated and tested
`estimate`	The estimated coefficient
`std.error`	The standard error from the linear model
`statistic`	Wald z-statistic
`p.value`	two-sided p-value

If `conf.int=TRUE`, it also includes columns for `conf.low` and `conf.high`, computed with `confint`.

`glance.multinom` returns a

`glance.multinom` returns a one-row data.frame with the columns

`edf`	The effective degrees of freedom
`deviance`	deviance
`AIC`	the Akaike Information Criterion

Examples

```
if (require(nnet) & require(MASS)){
  example(birthwt)
  bwt.mu <- multinom(low ~ ., bwt)
  tidy(bwt.mu)
  glance(bwt.mu)

  #* This model is a truly terrible model
  #* but it should show you what the output looks
  #* like in a multinomial logistic regression

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

`nlme_tidiers` *Tidying methods for mixed effects models*

Description

These methods tidy the coefficients of mixed effects models of the `lme` class from functions of the `nlme` package.

Usage

```
## S3 method for class 'lme'
tidy(x, effects = "random", ...)

## S3 method for class 'lme'
augment(x, data = x$data, newdata, ...)

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

Arguments

`x`	An object of class `lme`, such as those from `lme` or `nlme`
`effects`	Either "random" (default) or "fixed"
`...`	extra arguments (not used)
`data`	original data this was fitted on; if not given this will attempt to be reconstructed
`newdata`	new data to be used for prediction; optional

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

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

`tidy` returns one row for each estimated effect, either random or fixed depending on the `effects` parameter. If `effects = "random"`, it contains the columns

`group`	the group within which the random effect is being estimated
`level`	level within group
`term`	term being estimated

estimate	estimated coefficient

If effects="fixed", tidy returns the columns

term	fixed term being estimated
estimate	estimate of fixed effect
std.error	standard error
statistic	t-statistic
p.value	P-value computed from t-statistic

augment returns one row for each original observation, with columns (each prepended by a .) added. Included are the columns

.fitted	predicted values
.resid	residuals
.fixed	predicted values with no random effects

glance returns one row with the columns

sigma	the square root of the estimated residual variance
logLik	the data's log-likelihood under the model
AIC	the Akaike Information Criterion
BIC	the Bayesian Information Criterion
deviance	returned as NA. To quote Brian Ripley on R-help: McCullagh & Nelder (1989) would be the authoritative reference, but the 1982 first edition manages to use 'deviance' in three separate senses on one page.

See Also

na.action

Examples

```
if (require("nlme") & require("lme4")) {
    # example regressions are from lme4 documentation, but used for nlme
    lmm1 <- lme(Reaction ~ Days, random=~ Days|Subject, sleepstudy)
    tidy(lmm1)
    tidy(lmm1, effects = "fixed")
    head(augment(lmm1, sleepstudy))
    glance(lmm1)


    startvec <- c(Asym = 200, xmid = 725, scal = 350)
    nm1 <- nlme(circumference ~ SSlogis(age, Asym, xmid, scal),
                 data = Orange,
                 fixed = Asym + xmid + scal ~1,
                 random = Asym ~1,
                 start = startvec)
```

```
    tidy(nm1)
    tidy(nm1, effects = "fixed")
    head(augment(nm1, Orange))
    glance(nm1)
}
```

`nls_tidiers` *Tidying methods for a nonlinear model*

Description

These methods tidy the coefficients of a nonlinear model into a summary, augment the original data with information on the fitted values and residuals, and construct a one-row glance of the model's statistics.

Usage

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

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

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

Arguments

`x`	An object of class "nls"
`conf.int`	whether to include a confidence interval
`conf.level`	confidence level of the interval, used only if `conf.int=TRUE`
`quick`	whether to compute a smaller and faster version, containing only the `term` and `estimate` columns.
`...`	extra arguments (not used)
`data`	original data this was fitted on; if not given this will attempt to be reconstructed from nls (may not be successful)
`newdata`	new data frame to use for predictions

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

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

`tidy` returns one row for each coefficient in the model, with five columns:

`term`	The term in the nonlinear model being estimated and tested
`estimate`	The estimated coefficient
`std.error`	The standard error from the linear model
`statistic`	t-statistic
`p.value`	two-sided p-value

`augment` returns one row for each original observation, with two columns added:

`.fitted`	Fitted values of model
`.resid`	Residuals

If `newdata` is provided, these are computed on based on predictions of the new data.

`glance` returns one row with the columns

`sigma`	the square root of the estimated residual variance
`isConv`	whether the fit successfully converged
`finTol`	the achieved convergence tolerance
`logLik`	the data's log-likelihood under the model
`AIC`	the Akaike Information Criterion
`BIC`	the Bayesian Information Criterion
`deviance`	deviance
`df.residual`	residual degrees of freedom

See Also

na.action

`nls` and `summary.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))

# augment on new data
```

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

`optim_tidiers` *Tidiers for lists returned from optim*

Description

Tidies objects returned by the `optim` function for general-purpose minimization and maximization.

Usage

```
tidy_optim(x, ...)

glance_optim(x, ...)
```

Arguments

`x`	list returned from `optim`
`...`	extra arguments

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns a data frame with one row per parameter that was estimated, with columns

`parameter`	name of the parameter, or `parameter1`, `parameter2`... if the input vector is not named
`value`	parameter value that minimizes or maximizes the output

`glance` returns a one-row data frame with the columns

`value`	minimized or maximized output value
`function.count`	number of calls to `fn`
`gradient.count`	number of calls to `gr`
`convergence`	convergence code representing the error state

See Also

`optim`

Examples

```
func <- function(x) {
    (x[1] - 2)^2 + (x[2] - 3)^2 + (x[3] - 8)^2
}

o <- optim(c(1, 1, 1), func)

tidy(o)
glance(o)
```

`orcutt_tidiers` *Tidiers for Cochrane Orcutt object*

Description

Tidies a Cochrane Orcutt object, which estimates autocorrelation and beta coefficients in a linear fit.

Usage

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

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

Arguments

`x`	An "orcutt" object returned by `cochrane.orcutt`
`...`	Extra arguments passed on to `tidy.lm`

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns the same information as `tidy.lm`, though without confidence interval options.

`glance`

`glance` returns a one-row data frame with the following columns:

`r.squared`	R-squared
`adj.r.squared`	Adjusted R-squared
`rho`	Spearman's rho autocorrelation
`number.interaction`	Number of interactions

dw.original	Durbin-Watson statistic of original fit
p.value.original	P-value of original Durbin-Watson statistic
dw.transformed	Durbin-Watson statistic of transformed fit
p.value.transformed	P-value of autocorrelation after transformation

Examples

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

if (require("orcutt", quietly = TRUE)) {
  co <- cochrane.orcutt(reg)
  co

  tidy(co)
  glance(co)
}
```

---

plm_tidiers *Tidiers for panel regression linear models*

Description

Tidiers for panel regression linear models

Usage

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

## S3 method for class 'plm'
augment(x, data = as.data.frame(stats::model.frame(x)), ...)

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

Arguments

x	a "plm" object representing a panel object
conf.int	whether to include a confidence interval
conf.level	confidence level of the interval, used only if conf.int=TRUE
exponentiate	whether to exponentiate the coefficient estimates and confidence intervals
...	extra arguments, not used
data	original dataset

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy.plm` returns a data frame with one row per coefficient, of the same form as `tidy.lm`.

`augment` returns a data frame with one row for each initial observation, adding the columns

`.fitted`	predicted (fitted) values
`.resid`	residuals

`glance` returns a one-row data frame with columns

`r.squared`	The percent of variance explained by the model
`adj.r.squared`	r.squared adjusted based on the degrees of freedom
`statistic`	F-statistic
`p.value`	p-value from the F test, describing whether the full regression is significant
`deviance`	deviance
`df.residual`	residual degrees of freedom

See Also

`lm_tidiers`

Examples

```
if (require("plm", quietly = TRUE)) {
    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 = .9)

    head(augment(zz))

    glance(zz)
}
```

poLCA_tidiers *Tidiers for poLCA objects*

Description

Tidiers for poLCA latent class regression models. Summarize the probabilities of each outcome for each variable within each class with `tidy`, add predictions to the data with `augment`, or find the log-likelihood/AIC/BIC with `glance`.

Usage

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

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

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

Arguments

`x`	A poLCA object
`...`	Extra arguments, not used
`data`	For `augment`, the original dataset used to fit the latent class model. If not given, uses manifest variables in `x$y` and, if applicable, covariates in `x$x`

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns a data frame with one row per variable-class-outcome combination, with columns:

variable Manifest variable

class Latent class ID, an integer

outcome Outcome of manifest variable

estimate Estimated class-conditional response probability

std.error Standard error of estimated probability

`augment` returns a data frame with one row for each original observation, augmented with the following columns:

.class Predicted class, using modal assignment

.probability Posterior probability of predicted class

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, since it would result in potentially many additional columns, which augment tends to avoid.

`glance` returns a one-row data frame with the following columns:

logLik the data's log-likelihood under the model

AIC the Akaike Information Criterion

BIC the Bayesian Information Criterion

g.squared The likelihood ratio/deviance statistic

chi.squared The Pearson Chi-Square goodness of fit statistic for multiway tables

df Number of parameters estimated, and therefore degrees of freedom used

df.residual Number of residual degrees of freedom left

Examples

```
if (require("poLCA", quietly = TRUE)) {
  library(poLCA)
  library(dplyr)

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

  M1
  tidy(M1)
  head(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)

  set.seed(2016)
  # compare multiple
  mods <- data_frame(nclass = 1:3) %>%
    group_by(nclass) %>%
    do(mod = poLCA(f, values, nclass = .$nclass, verbose = FALSE))

  # compare log-likelihood and/or AIC, BIC
  mods %>%
    glance(mod)

  ## 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)
  head(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)
  head(au)
  au %>%
    count(.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)
  head(au2)
  dim(au2)
}
```

`prcomp_tidiers` *Tidying methods for principal components analysis via* `prcomp`

Description

These tidiers operate on the results of a principal components analysis computed using `prcomp`. The `tidy` method returns a data frame with either the eigenvectors representing each row or each column.

Usage

```
## S3 method for class 'prcomp'
tidy(x, matrix = "u", ...)

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

Arguments

`x`	an object of class `"prcomp"` resulting from a call to `prcomp`

matrix	character; Indicates which sets of eigenvectors are returned in tidy form. "v", "rotation", or "variables" will return information about each variable, while "u", "x", or "samples" (default) returns the loadings for each original row. "d" or "pcs" returns information about each principal component.
...	Extra arguments, not used
data	the original data on which principal components analysis was performed. This cannot be recovered from x. If newdata is supplied, data is ignored. If both data and newdata are missing, only the fitted locations on the principal components are returned.
newdata	data frame; new observations for which locations on principal components are sought.

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

If matrix is "u", "samples", or "x", the tidy method returns

row The sample labels (rownames) 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 "v", "variables", or "rotation", the tidy method returns

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" or "pcs", the tidy method returns

PC An integer vector indicating the principal component

std.dev Standard deviation explained by this PC

percent Percentage of variation explained

cumulative Cumulative percentage of variation explained

The augment.prcomp method returns a data frame containing fitted locations on the principal components for the observed data plus either the original data or the new data if supplied via data or newdata respectively.

Author(s)

Gavin L. Simpson

See Also

prcomp, svd_tidiers

Examples

```
pc <- prcomp(USArrests, scale = TRUE)

# information about rotation
head(tidy(pc))

# information about samples (states)
head(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)
head(au)

ggplot(au, aes(.fittedPC1, .fittedPC2)) +
  geom_point() +
  geom_text(aes(label = .rownames), vjust = 1, hjust = 1)
```

---

process_ergm *helper function to process a tidied ergm object*

Description

Optionally exponentiates the coefficients, and optionally adds a confidence interval, to a tidied ergm object.

Usage

```
process_ergm(ret, x, conf.int = FALSE, conf.level = 0.95,
  exponentiate = FALSE)
```

Arguments

ret	data frame with a tidied version of a coefficient matrix
x	an "ergm" object
conf.int	whether to include a confidence interval
conf.level	confidence level of the interval, used only if conf.int=TRUE
exponentiate	whether to exponentiate the coefficient estimates and confidence intervals (typical for logistic regression)

---

process_geeglm *helper function to process a tidied geeglm object*

Description

Adds a confidence interval, and possibly exponentiates, a tidied object.

Usage

```
process_geeglm(ret, x, conf.int = FALSE, conf.level = 0.95,
  exponentiate = FALSE)
```

Arguments

ret	data frame with a tidied version of a coefficient matrix
x	a "geeglm" object
conf.int	whether to include a confidence interval
conf.level	confidence level of the interval, used only if conf.int=TRUE
exponentiate	whether to exponentiate the coefficient estimates and confidence intervals (typical for log distributions)

---

process_lm *helper function to process a tidied lm object*

Description

Adds a confidence interval, and possibly exponentiates, a tidied object. Useful for operations shared between lm and biglm.

Usage

```
process_lm(ret, x, conf.int = FALSE, conf.level = 0.95,
  exponentiate = FALSE)
```

Arguments

ret	data frame with a tidied version of a coefficient matrix
x	an "lm", "glm", "biglm", or "bigglm" object
conf.int	whether to include a confidence interval
conf.level	confidence level of the interval, used only if conf.int=TRUE
exponentiate	whether to exponentiate the coefficient estimates and confidence intervals (typical for logistic regression)

---

process_rq *Helper function for tidy.rq and tidy.rqs*

Description

See documentation for summary.rq for complete description of the options for se.type, conf.int, etc.

Usage

```
process_rq(rq_obj, se.type = "rank", conf.int = TRUE, conf.level = 0.95,
  ...)
```

Arguments

rq_obj	an object returned by summary.rq or summary.rqs
se.type	type of standard errors used in summary.rq or summary.rqs
conf.int	whether to include a confidence interval
conf.level	confidence level for confidence interval
...	currently unused

---

pyears_tidiers *Tidy person-year summaries*

Description

These tidy the output of pyears, a calculation of the person-years of follow-up time contributed by a cohort of subject. Since the output of pyears$data is already tidy (if the data.frame = TRUE argument is given), this does only a little work and should rarely be necessary.

Usage

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

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

Arguments

x	a "pyears" object
...	extra arguments (not used)

Value

tidy returns a data.frame with the columns

pyears	person-years of exposure
n	number of subjects contributing time
event	observed number of events
expected	expected number of events (present only if a ratetable term is present)

If the data.frame = TRUE argument is supplied to pyears, this is simply the contents of x$data.

glance returns a one-row data frame with

total	total number of person-years tabulated
offtable	total number of person-years off table

This contains the values printed by summary.pyears.

See Also

pyears

Examples

```
if (require("survival", quietly = TRUE)) {
    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)
    head(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)
    head(tidy(pfit2))
    glance(pfit2)
}
```

rcorr_tidiers *Tidying methods for rcorr objects*

Description

Tidies a correlation matrix from the `rcorr` function in the "Hmisc" package, including correlation estimates, p-values, and the number of observations in each pairwise correlation. Note that it returns these in "long", or "melted", format, with one row for each pair of columns being compared.

Usage

```
## S3 method for class 'rcorr'
tidy(x, diagonal = FALSE, ...)
```

Arguments

`x`	An object of class "rcorr"
`diagonal`	Whether to include diagonal elements (where `estimate` is 1 and `p.value` is NA), default FALSE
`...`	extra arguments (not used)

Details

Only half the symmetric matrix is shown.

Value

A data.frame with one row for each pairing in the correlation matrix. Columns are:

`column1`	Name or index of the first column being described
`column2`	Name or index of the second column being described
`estimate`	Estimate of Pearson's r or Spearman's rho
`n`	Number of observations used to compute the correlation
`p.value`	P-value of correlation

Examples

```
if (require("Hmisc", quietly = TRUE)) {
    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)
    head(td)

    library(ggplot2)
    ggplot(td, aes(p.value)) +
        geom_histogram(binwidth = .1)

    ggplot(td, aes(estimate, p.value)) +
        geom_point() +
        scale_y_log10()
}
```

`ridgelm_tidiers` *Tidying methods for ridgelm objects from the MASS package*

Description

These methods tidies the coefficients of a ridge regression model chosen at each value of lambda into a data frame, or constructs a one-row glance of the model's choices of lambda (the ridge constant)

Usage

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

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

Arguments

`x`	An object of class "ridgelm"
`...`	extra arguments (not used)

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy.ridgelm` returns one row for each combination of choice of lambda and term in the formula, with columns:

`lambda`	choice of lambda
`GCV`	generalized cross validation value for this lambda
`term`	the term in the ridge regression model being estimated
`estimate`	estimate of scaled coefficient using this lambda
`scale`	Scaling factor of estimated coefficient

`glance.ridgelm` returns a one-row data.frame with the 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

This is similar to the output of `select.ridgelm`, but it is returned rather than printed.

Examples

```
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)
```

---

rlm_tidiers *Tidying methods for an rlm (robust linear model) object*

Description

This method provides a glance of an "rlm" object. The `tidy` and `augment` methods are handled by lm_tidiers.

Usage

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

Arguments

`x`	rlm object
`...`	extra arguments (not used)

Value

`glance.rlm` returns a one-row data.frame with the columns

`sigma`	The square root of the estimated residual variance
`converged`	whether the IWLS converged
`logLik`	the data's log-likelihood under the model
`AIC`	the Akaike Information Criterion
`BIC`	the Bayesian Information Criterion
`deviance`	deviance

See Also

lm_tidiers

Examples

```
library(MASS)

r <- rlm(stack.loss ~ ., stackloss)
tidy(r)
augment(r)
glance(r)
```

---

robust_tidiers *Tidiers for lmRob and glmRob objects*

Description

Tidying robust regression objects from the robust package. The tidy and augment methods simply pass it on to the linear model tidiers.

Usage

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

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

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

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

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

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

Arguments

x	An lmRob or glmRob object with a robust regression
...	Extra arguments, not used

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` and `augment` return the same results as `lm_tidiers`.

On an `lmRob` `glance` returns a one-row data frame with the following columns:

r.squared	R-squared
deviance	Robust deviance
sigma	Residual scale estimate
df.residual	Number of residual degrees of freedom

On an `glmRob` `glance` returns a one-row data frame with the following columns:

deviance	Robust deviance
null.deviance	Deviance under the null model
df.residual	Number of residual degrees of freedom

See Also

`lm_tidiers`, `lmRob`, `glmRob`

Examples

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

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

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

`rowwise_df_tidiers` *Tidying methods for rowwise_dfs from dplyr, for tidying each row and recombining the results*

Description

These `tidy`, `augment` and `glance` methods are for performing tidying on each row of a rowwise data frame created by dplyr's `group_by` and `do` operations. They first group a rowwise data frame based on all columns that are not lists, then perform the tidying operation on the specified column. This greatly shortens a common idiom of extracting tidy/augment/glance outputs after a do statement.

Usage

```
## S3 method for class 'rowwise_df'
tidy(x, object, ...)

## S3 method for class 'rowwise_df'
tidy_(x, object, ...)

## S3 method for class 'rowwise_df'
augment(x, object, ...)

## S3 method for class 'rowwise_df'
augment_(x, object, ...)

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

## S3 method for class 'rowwise_df'
glance_(x, object, ...)

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

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

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

Arguments

`x`	a rowwise_df
`object`	the column name of the column containing the models to be tidied. For tidy, augment, and glance it should be the bare name; for _ methods it should be quoted.
`...`	additional arguments to pass on to the respective tidying method

Details

Note that this functionality is not currently implemented for data.tables, since the result of the do operation is difficult to distinguish from a regular data.table.

Value

A "grouped_df", where the non-list columns of the original are used as grouping columns alongside the tidied outputs.

Examples

```
library(dplyr)
regressions <- mtcars %>%
    group_by(cyl) %>%
    do(mod = lm(mpg ~ wt, .))

regressions

regressions %>% tidy(mod)
regressions %>% augment(mod)
regressions %>% glance(mod)

# we can provide additional arguments to the tidying function
regressions %>% tidy(mod, conf.int = TRUE)

# we can also include the original dataset as a "data" argument
# to augment:
regressions <- mtcars %>%
    group_by(cyl) %>%
    do(mod = lm(mpg ~ wt, .), original = (.))

# this allows all the original columns to be included:
regressions %>% augment(mod)  # doesn't include all original
regressions %>% augment(mod, data = original)  # includes all original
```

---

rq_tidiers *Tidying methods for quantile regression models*

Description

These methods tidy the coefficients of a quantile regression model into a summary, augment the original data with information on the fitted values and residuals, and construct a glance of the model's statistics.

Usage

```
## S3 method for class 'rq'
tidy(x, se.type = "rank", conf.int = TRUE, conf.level = 0.95,
  alpha = 1 - conf.level, ...)

## S3 method for class 'rqs'
tidy(x, se.type = "rank", conf.int = TRUE,
  conf.level = 0.95, alpha = 1 - conf.level, ...)

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

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

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

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

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

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

Arguments

x	model object returned by rq or nlrq
se.type	Type of standard errors to calculate; see summary.rq
conf.int	boolean; should confidence intervals be calculated, ignored if se.type = "rank"
conf.level	confidence level for intervals
alpha	confidence level when se.type = "rank"; defaults to the same as conf.level although the specification is inverted
...	other arguments passed on to summary.rq
data	Original data, defaults to extracting it from the model
newdata	If provided, new data frame to use for predictions

Details

If se.type != "rank" and conf.int = TRUE confidence intervals are calculated by summary.rq. Otherwise they are standard t based intervals.

This simply calls augment.nls on the "nlrq" object.

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy.rq` returns a data frame with one row for each coefficient. The columns depend upon the confidence interval method selected.

`tidy.rqs` returns a data frame with one row for each coefficient at each quantile that was estimated. The columns depend upon the confidence interval method selected.

`tidy.nlrq` returns one row for each coefficient in the model, with five columns:

`term`	The term in the nonlinear model being estimated and tested
`estimate`	The estimated coefficient
`std.error`	The standard error from the linear model
`statistic`	t-statistic
`p.value`	two-sided p-value

`glance.rq` returns one row for each quantile (tau) with the columns:

`tau`	quantile estimated
`logLik`	the data's log-likelihood under the model
`AIC`	the Akaike Information Criterion
`BIC`	the Bayesian Information Criterion
`df.residual`	residual degrees of freedom

`glance.rq` returns one row for each quantile (tau) with the columns:

`tau`	quantile estimated
`logLik`	the data's log-likelihood under the model
`AIC`	the Akaike Information Criterion
`BIC`	the Bayesian Information Criterion
`df.residual`	residual degrees of freedom

`augment.rq` returns a row for each original observation with the following columns added:

`.resid`	Residuals
`.fitted`	Fitted quantiles of the model
`.tau`	Quantile estimated

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`	Lower confidence interval value
`.conf.high`	Upper confidence interval value

See `predict.rq` for details on additional arguments to specify confidence intervals. `predict.rq` does not provide confidence intervals when `newdata` is provided.

`augment.rqs` returns a row for each original observation and each estimated quantile (`tau`) with the following columns added:

.resid	Residuals
.fitted	Fitted quantiles of the model
.tau	Quantile estimated

predict.rqs does not return confidence interval estimates.

augment.rqs returns a row for each original observation with the following columns added:

.resid	Residuals
.fitted	Fitted quantiles of the model

---

rstanarm_tidiers *Tidying methods for an rstanarm model*

Description

These methods tidy the estimates from stanreg-objects (fitted model objects from the **rstanarm** package) into a summary.

Usage

```
## S3 method for class 'stanreg'
tidy(x, parameters = "non-varying", intervals = FALSE,
  prob = 0.9, ...)

## S3 method for class 'stanreg'
glance(x, looic = FALSE, ...)
```

Arguments

x	Fitted model object from the **rstanarm** package. See stanreg-objects.
parameters	One or more of "non-varying", "varying", "hierarchical", "auxiliary" (can be abbreviated). See the Value section for details.
intervals	If TRUE columns for the lower and upper bounds of the 100*prob% posterior uncertainty intervals are included. See posterior_interval for details.
prob	See posterior_interval.
...	For glance, if looic=TRUE, optional arguments to loo.stanreg.
looic	Should the LOO Information Criterion (and related info) be included? See loo.stanreg for details. Note: for models fit to very large datasets this can be a slow computation.

Value

All tidying methods return a `data.frame` without rownames. The structure depends on the method chosen.

When `parameters="non-varying"` (the default), `tidy.stanreg` returns one row for each coefficient, with three columns:

`term`	The name of the corresponding term in the model.
`estimate`	A point estimate of the coefficient (posterior median).
`std.error`	A standard error for the point estimate based on `mad`. See the *Uncertainty estimates* section in `print.stanreg` for more details.

For models with group-specific parameters (e.g., models fit with `stan_glmer`), setting `parameters="varying"` selects the group-level parameters instead of the non-varying regression coefficients. Addtional columns are added indicating the `level` and `group`. Specifying `parameters="hierarchical"` selects the standard deviations and (for certain models) correlations of the group-level parameters.

Setting `parameters="auxiliary"` will select parameters other than those included by the other options. The particular parameters depend on which **rstanarm** modeling function was used to fit the model. For example, for models fit using `stan_glm.nb` the overdispersion parameter is included if `parameters="aux"`, for `stan_lm` the auxiliary parameters include the residual SD, R^2, and log(fit_ratio), etc.

If `intervals=TRUE`, columns for the `lower` and `upper` values of the posterior intervals computed with `posterior_interval` are also included.

`glance` returns one row with the columns

`algorithm`	The algorithm used to fit the model.
`pss`	The posterior sample size (except for models fit using optimization).
`nobs`	The number of observations used to fit the model.
`sigma`	The square root of the estimated residual variance, if applicable. If not applicable (e.g., for binomial GLMs), `sigma` will be given the value `1` in the returned object.

If `looic=TRUE`, then the following additional columns are also included:

`looic`	The LOO Information Criterion.
`elpd_loo`	The expected log predictive density (`elpd_loo = -2 * looic`).
`p_loo`	The effective number of parameters.

See Also

`summary.stanreg`

Examples

```
## Not run:
fit <- stan_glmer(mpg ~ wt + (1|cyl) + (1+wt|gear), data = mtcars,
                  iter = 300, chains = 2)
```

```
# non-varying ("population") parameters
tidy(fit, intervals = TRUE, prob = 0.5)

# hierarchical sd & correlation parameters
tidy(fit, parameters = "hierarchical")

# group-specific deviations from "population" parameters
tidy(fit, parameters = "varying")

# glance method
glance(fit)
glance(fit, looic = TRUE, cores = 1)

## End(Not run)
```

`sexpfit_tidiers` *Tidy an expected survival curve*

Description

This constructs a summary across time points or overall of an expected survival curve. Note that this contains less information than most survfit objects.

Usage

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

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

Arguments

`x`	"survexp" object
`...`	extra arguments (not used)

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns a one row for each time point, with columns

`time`	time point
`estimate`	estimated survival
`n.risk`	number of individuals at risk

`glance` returns a one-row data.frame with the columns:

n.max	maximum number of subjects at risk
n.start	starting number of subjects at risk
timepoints	number of timepoints

Examples

```
if (require("survival", quietly = TRUE)) {
    sexpfit <- survexp(futime ~ 1, rmap=list(sex="male", year=accept.dt,
                                             age=(accept.dt-birth.dt)),
                       method='conditional', data=jasa)

    tidy(sexpfit)
    glance(sexpfit)
}
```

---

smooth.spline_tidiers *tidying methods for smooth.spline objects*

Description

This combines the original data given to smooth.spline with the fit and residuals

Usage

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

## S3 method for class 'smooth.spline'
glance(x, ...)
```

Arguments

x	a smooth.spline object
data	defaults to data used to fit model
...	not used in this method

Details

No tidy method is provided for smooth.spline objects.

Value

augment returns the original data with extra columns:

.fitted	Fitted values of model
.resid	Residuals

glance returns one row with columns

spar	smoothing parameter
lambda	choice of lambda corresponding to spar
df	equivalent degrees of freedom
crit	minimized criterion
pen.crit	penalized criterion
cv.crit	cross-validation score

Examples

```
spl <- smooth.spline(mtcars$wt, mtcars$mpg, df = 4)
head(augment(spl, mtcars))
head(augment(spl))  # calls original columns x and y

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

---

sparse_tidiers *Tidy a sparseMatrix object from the Matrix package*

Description

Tidy a sparseMatrix object from the Matrix package into a three-column data frame, row, column, and value (with zeros missing). If there are row names or column names, use those, otherwise use indices

Usage

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

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

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

Arguments

x	A Matrix object
...	Extra arguments, not used

`speedlm_tidiers` *Tidying methods for a speedlm model*

Description

These methods tidy the coefficients of a "speedlm" object into a summary, augment the original data with information on the fitted values and residuals, and construct a one-row glance of the model's statistics.

Usage

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

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

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

Arguments

x	speedlm object
conf.int	whether to include a confidence interval
conf.level	confidence level of the interval, used only if `conf.int=TRUE`
exponentiate	whether to exponentiate the coefficient estimates and confidence intervals (typical for logistic regression)
quick	whether to compute a smaller and faster version, containing only the `term` and `estimate` columns.
...	extra arguments (not used)
data	data frame to augment
newdata	new data to use for predictions, optional

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy.speedlm` returns the tidied output of the lm with one row for each term in the formula. The columns match those in lm_tidiers.

`glance.speedlm` returns a one-row data.frame with the columns

`r.squared`	The percent of variance explained by the model
`adj.r.squared`	r.squared adjusted based on the degrees of freedom
`statistic`	F-statistic
`p.value`	p-value from the F test, describing whether the full regression is significant
`df`	Degrees of freedom used by the coefficients
`logLik`	the data's log-likelihood under the model
`AIC`	the Akaike Information Criterion
`BIC`	the Bayesian Information Criterion
`deviance`	deviance
`df.residual`	residual degrees of freedom

`augment.speedlm` returns one row for each observation, with just one column added:

`.fitted`	Fitted values of model

See Also

lm_tidiers, biglm_tidiers

Examples

```
if (require("speedglm", quietly = TRUE)) {
    mod <- speedglm::speedlm(mpg ~ wt + qsec, data = mtcars)
    tidy(mod)
    glance(mod)
    augment(mod)
}
```

sp_tidiers *tidying methods for classes from the sp package.*

Description

Tidy classes from the sp package to allow them to be plotted using ggplot2. To figure out the correct variable name for region, inspect `as.data.frame(x)`.

Usage

```
## 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	SpatialPolygonsDataFrame to convert into a dataframe.
region	name of variable used to split up regions
...	not used by this method

Details

These functions originated in the ggplot2 package as "fortify" functions.

Examples

```
if (require("maptools")) {
    sids <- system.file("shapes/sids.shp", package="maptools")
    nc1 <- readShapePoly(sids,
    proj4string = CRS("+proj=longlat +datum=NAD27"))
```

```
    nc1_df <- tidy(nc1)
}
```

---

summary_tidiers *Tidiers for summaryDefault objects*

Description

Tidy a summary of a vector.

Usage

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

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

Arguments

x	summaryDefault object
...	extra arguments, not used

Value

Both tidy and glance return the same object: a one-row data frame with columns

minimum	smallest value in original vector
q1	value at the first quartile
median	median of original vector
mean	mean of original vector
q3	value at the third quartile
maximum	largest value in original vector
NAs	number of NA values (if any)

See Also

summary

Examples

```
v <- rnorm(1000)
s <- summary(v)
s

tidy(s)
glance(s)

v2 <- c(v,NA)
tidy(summary(v2))
```

`survdiff_tidiers` *Tidiers for Tests of Differences between Survival Curves*

Description

Tidiers for Tests of Differences between Survival Curves

Usage

```
## S3 method for class 'survdiff'
tidy(x, strata = FALSE, ...)

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

Arguments

`x`	a "survdiff" object
`strata`	logical, whether to include strata in the output
`...`	other arguments passed to/from other methods, currently ignored

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` on "survdiff" objects returns a data frame with the following columns:

`...`	initial column(s) correspond to grouping factors (right-hand side of the formula)
`obs`	weighted observed number of events in each group
`exp`	weighted expected number of events in each group
`N`	number of subjects in each group

`glance` on "survdiff" objects returns a data frame with the following columns:

statistic	value of the test statistic
df	degrees of freedom
p.value	p-value

See Also

survdiff

Examples

```
if( require("survival") ) {
    s <- survdiff( Surv(time, status) ~ pat.karno + strata(inst), data=lung)
    tidy(s)
    glance(s)
}
```

---

survfit_tidiers *tidy survival curve fits*

Description

Construct tidied data frames showing survival curves over time.

Usage

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

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

Arguments

x	"survfit" object
...	extra arguments, not used

Details

glance does not work on multi-state survival curves, since the values glance outputs would be calculated for each state. tidy does work for multi-state survival objects, and includes a state column to distinguish between them.

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns a row for each time point, with columns

`time`	timepoint
`n.risk`	number of subjects at risk at time t0
`n.event`	number of events at time t
`n.censor`	number of censored events
`estimate`	estimate of survival or cumulative incidence rate when multistate
`std.error`	standard error of estimate
`conf.high`	upper end of confidence interval
`conf.low`	lower end of confidence interval
`state`	state if multistate survfit object inputted
`strata`	strata if stratified survfit object inputted

`glance` returns one-row data.frame with the columns displayed by `print.survfit`

`records`	number of observations
`n.max`	n.max
`n.start`	n.start
`events`	number of events
`rmean`	Restricted mean (see print.survfit)
`rmean.std.error`	Restricted mean standard error
`median`	median survival
`conf.low`	lower end of confidence interval on median
`conf.high`	upper end of confidence interval on median

Examples

```
if (require("survival", quietly = TRUE)) {
    cfit <- coxph(Surv(time, status) ~ age + sex, lung)
    sfit <- survfit(cfit)

    head(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)
    head(td_multi)
    tail(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)

    # perform simple bootstrapping
    library(dplyr)
    bootstraps <- lung %>% bootstrap(100) %>%
        do(tidy(survfit(coxph(Surv(time, status) ~ age + sex, .))))

    ggplot(bootstraps, aes(time, estimate, group = replicate)) +
        geom_line(alpha = .25)

    bootstraps_bytime <- bootstraps %>% group_by(time) %>%
        summarize(median = median(estimate),
                  low = quantile(estimate, .025),
                  high = quantile(estimate, .975))

    ggplot(bootstraps_bytime, aes(x = time, y = median)) + geom_line() +
        geom_ribbon(aes(ymin = low, ymax = high), alpha = .25)

    # bootstrap for median survival
    glances <- lung %>%
        bootstrap(100) %>%
        do(glance(survfit(coxph(Surv(time, status) ~ age + sex, .))))

    glances

    qplot(glances$median, binwidth = 15)
    quantile(glances$median, c(.025, .975))
}
```

`survreg_tidiers` *Tidiers for a parametric regression survival model*

Description

Tidies the coefficients of a parametric survival regression model, from the "survreg" function, adds fitted values and residuals, or summarizes the model statistics.

Usage

```
## S3 method for class 'survreg'
tidy(x, conf.level = 0.95, ...)

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

```
augment(x, data = stats::model.frame(x), newdata,
  type.predict = "response", type.residuals = "response", ...)

## S3 method for class 'survreg'
glance(x, conf.level = 0.95, ...)
```

Arguments

x	a "survreg" model
conf.level	confidence level for CI
...	extra arguments (not used)
data	original data; if it is not provided, it is reconstructed as best as possible with model.frame
newdata	New data to use for prediction; optional
type.predict	type of prediction, default "response"
type.residuals	type of residuals to calculate, default "response"

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

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns a data.frame with one row for each term

term	name of term
estimate	estimate of coefficient
stderror	standard error
statistic	Z statistic
p.value	p-value
conf.low	low end of confidence interval
conf.high	high end of confidence interval

`augment` returns the original data.frame with the following additional columns:

.fitted	Fitted values of model
.se.fit	Standard errors of fitted values
.resid	Residuals

glance returns a one-row data.frame with the columns:

iter	number of iterations
df	degrees of freedom
statistic	chi-squared statistic
p.value	p-value from chi-squared test
logLik	log likelihood
AIC	Akaike information criterion
BIC	Bayesian information criterion
df.residual	residual degrees of freedom

See Also

na.action

Examples

```
if (require("survival", quietly = TRUE)) {
    sr <- survreg(Surv(futime, fustat) ~ ecog.ps + rx, ovarian,
           dist="exponential")

    td <- tidy(sr)
    augment(sr, ovarian)
    augment(sr)
    glance(sr)

    # coefficient plot
    library(ggplot2)
    ggplot(td, aes(estimate, term)) + geom_point() +
        geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) +
        geom_vline(xintercept = 0)
}
```

---

svd_tidiers *Tidying methods for singular value decomposition*

Description

These methods tidy the U, D, and V matrices returned by the svd function into a tidy format. Because svd returns a list without a class, this function has to be called by tidy.list when it recognizes a list as an SVD object.

Usage

```
tidy_svd(x, matrix = "u", ...)
```

Arguments

x	list containing d, u, v components, returned from svd
matrix	which of the u, d or v matrix to tidy
...	Extra arguments (not used)

Value

An SVD object contains a decomposition into u, d, and v matrices, such that `u %\*% diag(d) %\*% t(v)` gives the original matrix. This tidier gives a choice of which matrix to tidy.

When `matrix = "u"`, each observation represents one pair of row and principal component, with variables:

row	Number of the row in the original data being described
PC	Principal component
loading	Loading of this principal component for this row

When `matrix = "d"`, each observation represents one principal component, with variables:

PC	Principal component
d	Value in the d vector
percent	Percent of variance explained by this PC, which is proportional to d^2

When `matrix = "v"`, each observation represents a pair of a principal component and a column of the original matrix, with variables:

column	Column of original matrix described
PC	Principal component
value	Value of this PC for this column

See Also

svd, tidy.list

Examples

```
mat <- as.matrix(iris[, 1:4])
s <- svd(mat)

tidy_u <- tidy(s, matrix = "u")
head(tidy_u)

tidy_d <- tidy(s, matrix = "d")
tidy_d

tidy_v <- tidy(s, matrix = "v")
head(tidy_v)

library(ggplot2)
```

```
library(dplyr)

ggplot(tidy_d, aes(PC, percent)) +
    geom_point() +
    ylab("% of variance explained")

tidy_u %>%
    mutate(Species = iris$Species[row]) %>%
    ggplot(aes(Species, loading)) +
    geom_boxplot() +
    facet_wrap(~ PC, scale = "free_y")
```

`tidy` *Tidy the result of a test into a summary data.frame*

Description

The output of tidy is always a data.frame with disposable row names. It is therefore suited for further manipulation by packages like dplyr, reshape2, ggplot2 and ggvis.

Usage

```
tidy(x, ...)
```

Arguments

`x`	An object to be converted into a tidy data.frame
`...`	extra arguments

Value

a data.frame

`tidy.coeftest` *Tidying methods for coeftest objects*

Description

This tidies the result of a coefficient test, from the `coeftest` function in the `lmtest` package.

Usage

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

Arguments

x	coeftest object
...	extra arguments (not used)

Value

A data.frame with one row for each coefficient, with five columns:

term	The term in the linear model being estimated and tested
estimate	The estimated coefficient
std.error	The standard error
statistic	test statistic
p.value	p-value

Examples

```
if (require("lmtest", quietly = TRUE)) {
    data(Mandible)
    fm <- lm(length ~ age, data=Mandible, subset=(age <= 28))

    coeftest(fm)
    tidy(coeftest(fm))
}
```

---

tidy.default *Default tidying method*

Description

By default, tidy uses as.data.frame to convert its output. This is dangerous, as it may fail with an uninformative error message. Generally tidy is intended to be used on structured model objects such as lm or htest for which a specific S3 object exists.

Usage

```
## Default S3 method:
tidy(x, ...)
```

Arguments

x	an object to be tidied
...	extra arguments (not used)

Details

If you know that you want to use `as.data.frame` on your untidy object, just use it directly.

Value

A data frame, from `as.data.frame` applied to the input x.

`tidy.density` *tidy a density objet*

Description

Given a "density" object, returns a tidy data frame with two columns: points x where the density is estimated, points y for the estimate

Usage

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

Arguments

`x`	an object of class "density"
`...`	extra arguments (not used)

Value

a data frame with "x" and "y" columns

d <- density(faithful$eruptions, bw = "sj") head(tidy(d))

library(ggplot2) ggplot(tidy(d), aes(x, y)) + geom_line()

See Also

`density`

`tidy.dist` *Tidy a distance matrix*

Description

Tidy a distance matrix, such as that computed by the dist function, into a one-row-per-pair table. If the distance matrix does not include an upper triangle and/or diagonal, this will not either.

Usage

```
## S3 method for class 'dist'
tidy(x, diag = attr(x, "Diag"), upper = attr(x, "Upper"),
  ...)
```

Arguments

`x`	A "dist" object
`diag`	Whether to include the diagonal of the distance matrix. Defaults to whether the distance matrix includes it
`upper`	Whether to include the upper right triangle of the distance matrix. Defaults to whether the distance matrix includes it
`...`	Extra arguments, not used

Value

A data frame with one row for each pair of item distances, with columns:

item1 First item

item2 Second item

distance Distance between items

Examples

```
iris_dist <- dist(t(iris[, 1:4]))
iris_dist

tidy(iris_dist)
tidy(iris_dist, upper = TRUE)
tidy(iris_dist, diag = TRUE)
```

tidy.ftable *tidy an ftable object*

Description

An ftable contains a "flat" contingency table. This melts it into a data.frame with one column for each variable, then a `Freq` column. It directly uses the `stats:::as.data.frame.ftable` function

Usage

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

Arguments

`x`	An object of class "ftable"
`...`	Extra arguments (not used)

See Also

`ftable`

Examples

```
tidy(ftable(Titanic, row.vars = 1:3))
```

tidy.manova *tidy a MANOVA object*

Description

Constructs a data frame with one row for each of the terms in the model, containing the information from summary.manova.

Usage

```
## S3 method for class 'manova'
tidy(x, test = "Pillai", ...)
```

Arguments

x	object of class "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"
...	additional arguments passed on to summary.manova

Value

A data.frame with the columns

term	Term in design
statistic	Approximate F statistic
num.df	Degrees of freedom
p.value	P-value

Depending on which test statistic is specified, one of the following columns is also included:

pillai	Pillai's trace
wilks	Wilk's lambda
hl	Hotelling-Lawley trace
roy	Roy's greatest root

See Also

summary.manova

Examples

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

---

tidy.map *Tidy method for map objects.*

Description

This function turns a map into a data frame.

Usage

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

Arguments

x	map object
...	not used by this method

Details

This code and documentation originated in ggplot2, but was called "fortify." In broom, "fortify" became "augment", which is reserved for functions that *add* columns to existing data (based on a model fit, for example) so these functions were renamed as "tidy."

Examples

```
if (require("maps") && require("ggplot2")) {
    ca <- map("county", "ca", plot = FALSE, fill = TRUE)
    head(tidy(ca))
    qplot(long, lat, data = ca, geom = "polygon", group = group)

    tx <- map("county", "texas", plot = FALSE, fill = TRUE)
    head(tidy(tx))
    qplot(long, lat, data = tx, geom = "polygon", group = group,
          colour = I("white"))
}
```

---

tidy.NULL *tidy on a NULL input*

Description

tidy on a NULL input returns an empty data frame, which means it can be combined with other data frames (treated as "empty")

Usage

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

Arguments

x	A value NULL
...	extra arguments (not used)

Value

An empty data.frame

tidy.numeric *Tidy atomic vectors*

Description

Turn atomic vectors into data frames, where the names of the vector (if they exist) are a column and the values of the vector are a column.

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)

Examples

```
x <- 1:5
names(x) <- letters[1:5]
tidy(x)
```

tidy.pairwise.htest *tidy a pairwise hypothesis test*

Description

Tidy a pairwise.htest object, containing (adjusted) p-values for multiple pairwise hypothesis tests.

Usage

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

Arguments

x	a "pairwise.htest" object
...	extra arguments (not used)

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 data frame with one row per group/group comparison, with columns

group1	First group being compared
group2	Second group being compared
p.value	(Adjusted) p-value of comparison

See Also

pairwise.t.test, pairwise.wilcox.test

Examples

```
attach(airquality)
Month <- factor(Month, labels = month.abb[5:9])
ptt <- pairwise.t.test(Ozone, Month)
tidy(ptt)

attach(iris)
ptt2 <- pairwise.t.test(Petal.Length, Species)
tidy(ptt2)

tidy(pairwise.t.test(Petal.Length, Species, alternative = "greater"))
tidy(pairwise.t.test(Petal.Length, Species, alternative = "less"))

tidy(pairwise.wilcox.test(Petal.Length, Species))
```

tidy.power.htest *tidy a power.htest*

Description

tidy a power.htest

Usage

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

Arguments

x	a power.htest object
...	extra arguments, not used

Value

A data frame with one row per parameter passed in, with columns `n`, `delta`, `sd`, `sig.level`, and `power` (from the `power.htest` object).

See Also

power.t.test

Examples

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

library(ggplot2)
ggplot(tidy(ptt), aes(n, power)) + geom_line()
```

tidy.spec *tidy a spec objet*

Description

Given a "spec" object, which shows a spectrum across a range of frequencies, returns a tidy data frame with two columns: "freq" and "spec"

Usage

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

Arguments

x	an object of class "spec"
...	extra arguments (not used)

Value

a data frame with "freq" and "spec" columns

Examples

```
spc <- spectrum(lh)
tidy(spc)

library(ggplot2)
ggplot(tidy(spc), aes(freq, spec)) + geom_line()
```

---

tidy.table *tidy a table object*

Description

A table, typically created by the table function, contains a contingency table of frequencies across multiple vectors. This directly calls the `as.data.frame.table` method, which melts it into a data frame with one column for each variable and a `Freq` column.

Usage

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

Arguments

x	An object of class "table"
...	Extra arguments (not used)

See Also

`as.data.frame.table`

Examples

```
tab <- with(airquality, table(cut(Temp, quantile(Temp)), Month))
tidy(tab)
```

---

tidy.ts *tidy a ts timeseries object*

Description

Turn a ts object into a tidy data frame. Right now simply uses `as.data.frame.ts`.

Usage

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

Arguments

`x`	a "ts" object
`...`	extra arguments (not used)

Value

a tidy data frame

See Also

as.data.frame.ts

---

tidy.TukeyHSD *tidy a TukeyHSD object*

Description

Returns a data.frame with one row for each pairwise comparison

Usage

```
## S3 method for class 'TukeyHSD'
tidy(x, separate.levels = FALSE, ...)
```

Arguments

x	object of class "TukeyHSD"
separate.levels	Whether to separate comparison into level1 and level2 columns
...	additional arguments (not used)

Value

A data.frame with one row per comparison, containing columns

term	Term for which levels are being compared
comparison	Levels being compared, separated by -
estimate	Estimate of difference
conf.low	Low end of confidence interval of difference
conf.high	High end of confidence interval of difference
adj.p.value	P-value adjusted for multiple comparisons

If separate.levels = TRUE, the comparison column will be split up into level1 and level2.

See Also

TukeyHSD

Examples

```
fm1 <- aov(breaks ~ wool + tension, data = warpbreaks)
thsd <- TukeyHSD(fm1, "tension", ordered = TRUE)
tidy(thsd)
tidy(thsd, separate.levels = TRUE)

# may include comparisons on multiple terms
fm2 <- aov(mpg ~ as.factor(gear) * as.factor(cyl), data = mtcars)
tidy(TukeyHSD(fm2))
```

---

unrowname *strip rownames from an object*

Description

strip rownames from an object

Usage

```
unrowname(x)
```

Arguments

x	a data frame

---

xyz_tidiers *Tidiers for x, y, z lists suitable for persp, image, etc.*

Description

Tidies lists with components x, y (vector of coordinates) and z (matrix of values) which are typically used by functions such as `persp` or `image` and returned by interpolation functions such as `interp`.

Usage

```
tidy_xyz(x, ...)
```

Arguments

x	list with components x, y and z
...	extra arguments

Value

All tidying methods return a data.frame without rownames, whose structure depends on the method chosen.

`tidy` returns a data frame with columns x, y and z and one row per value in matrix z.

Examples

```
A <- list(x=1:5, y=1:3, z=matrix(runif(5*3), nrow=5))
image(A)
tidy(A)
```

---

zoo_tidiers *Tidying methods for a zoo object*

Description

Tidies `zoo` (Z's ordered observations) time series objects. `zoo` objects are not tidy by default because they contain one row for each index and one series per column, rather than one row per observation per series.

Usage

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

Arguments

x	An object of class "zoo"
...	extra arguments (not used)

Value

tidy returns a data frame with one row for each observation in each series, with the following columns:

index	Index (usually date) for the zoo object
series	Name of the series
value	Value of the observation

Examples

```
if (require("zoo", quietly = TRUE)) {
    set.seed(1071)

    # data generated as shown in the zoo vignette
    Z.index <- as.Date(sample(12450:12500, 10))
    Z.data <- matrix(rnorm(30), ncol = 3)
    colnames(Z.data) <- c("Aa", "Bb", "Cc")
    Z <- zoo(Z.data, Z.index)

    tidy(Z)

    if (require("ggplot2", quietly = TRUE)) {
        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()
    }
}
```

Index