Package 'broom'

April 20, 2020

Type Package

Title Convert Statistical Analysis Objects into Tidy Tibbles

Version 0.5.6

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

License MIT + file LICENSE

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

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

Depends R (>= 3.1)

Imports backports, dplyr, generics (>= 0.0.2), methods, nlme, purrr, reshape2, stringr, tibble (>= 3.0.0), tidyr

Suggests AER, akima, AUC, bbmle, betareg, biglm, binGroup, boot, brms, btergm, car, caret, coda, covr, e1071, emmeans, ergm, gam (>= 1.15), gamlss, gamlss.data, gamlss.dist, geepack, ggplot2, glmnet, gmm, Hmisc, irlba, Kendall, knitr, ks, Lahman, lavaan, lfe, lme4, lmodel2, lmtest, lsmeans, maps, maptools, MASS, Matrix, mclust, mgcv, muhaz, multcomp, network, nnet, orcutt (>= 2.2), ordinal, plm, plyr, poLCA, psych, quantreg, rgeos, rmarkdown, rsample, rstan, rstanarm, sp, speedglm, statnet.common, survey, survival, testthat, tseries, zoo

VignetteBuilder knitr

Encoding UTF-8

LazyData true

RoxygenNote 7.1.0

NeedsCompilation no

Author David Robinson [aut],
Alex Hayes [aut, cre] (<https://orcid.org/0000-0002-4985-5160>),
Matthieu Gomez [ctb],
Boris Demeshev [ctb],
Dieter Menne [ctb],
Benjamin Nutter [ctb],
Luke Johnston [ctb],
Ben Bolker [ctb],
Francois Briatte [ctb],
Jeffrey Arnold [ctb],
Jonah Gabry [ctb],
Luciano Selzer [ctb],
Gavin Simpson [ctb],
Jens Preussner [ctb],
Jay Hesselberth [ctb],
Hadley Wickham [ctb],
Matthew Lincoln [ctb],
Alessandro Gasparini [ctb],
Lukasz Komsta [ctb],
Frederick Novometsky [ctb],
Wilson Freitas [ctb],
Michelle Evans [ctb],
Jason Cory Brunson [ctb],
Simon Jackson [ctb],
Ben Whalley [ctb],
Michael Kuehn [ctb],
Jorge Cimentada [ctb],
Erle Holgersen [ctb],
Karl Dunkle Werner [ctb]

Maintainer Alex Hayes <alexpghayes@gmail.com>

Repository CRAN

Date/Publication 2020-04-20 17:10:02 UTC

R topics documented:

`argument_glossary` *Allowed argument names in tidiers*

Description

Allowed argument names in tidiers

Usage

```
argument_glossary
```

Format

A tibble with 3 variables:

method One of "glance", "augment" or "tidy".

argument Character name of allowed argument name.

description Character description of argument use.

Examples

```
argument_glossary
```

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

Description

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

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

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

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

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

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

Usage

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

Arguments

`x`	A `betareg` object produced by a call to `betareg::betareg()`.

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

Value

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

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

See Also

augment(), betareg::betareg()

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

Description

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

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it **must** be exactly the data that was used to fit the model object.

Pass datasets to newdata to augment data that was not used during model fitting. This still requires that all columns used to fit the model are present.

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

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

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

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

Usage

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

Arguments

x	A coxph object returned from survival::coxph().
data	A data.frame() or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
newdata	A data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
type.predict	Character indicating type of prediction to use. Passed to the type argument of the stats::predict() generic. Allowed arguments vary with model class, so be sure to read the predict.my_class documentation.
type.residuals	Character indicating type of residuals to use. Passed to the type argument of stats::residuals() generic. Allowed arguments vary with model class, so be sure to read the residuals.my_class documentation.

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

Details

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

Value

A tibble::tibble with the passed data and additional columns:

`.fitted`	Fitted values of model
`.se.fit`	Standard errors of fitted values
`.resid`	Residuals (not present if `newdata` specified.)

See Also

na.action

`augment()`, `survival::coxph()`

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

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

`augment.decomposed.ts` *Augment data with information from a(n) decomposed.ts object*

Description

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

Users may pass data to augment via either the `data` argument or the `newdata` argument. If the user passes data to the `data` argument, it **must** be exactly the data that was used to fit the model object.

Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that all columns used to fit the model are present.

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

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

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

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

Usage

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

Arguments

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

Value

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

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

See Also

`augment()`, `stats::decompose()`

Other decompose tidiers: `augment.stl()`

Examples

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

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

# Compare the original series to its decompositions.

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

# Visually compare seasonal decompositions in tidy data frames.

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

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

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

Value

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

When `data` is supplied, `augment.factanal` returns one row for each observation, with a factor score column added for each factor X, (`.fsX`).

See Also

`augment()`, `stats::factanal()`

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

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

Value

A tibble::tibble() containing the data passed to augment, and **additional** columns:

.fitted	The predicted response for that observation.
.resid	The residual for a particular point. Present only when data has been passed to augment via the data argument.

See Also

augment(), lfe::felm()

Other felm tidiers: tidy.felm()

---

augment.glm *Augment a(n) glm object*

Description

This augment method wraps augment.lm().

Usage

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

Arguments

x	A glm object returned from stats::glm().
...	Arguments passed on to augment.lm
	data A data.frame() or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
	newdata A data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
	type.predict Type of predictions to use when x is a glm object. Passed to stats::predict.glm().
	type.residuals Type of residuals to use when x is a glm object. Passed to stats::residuals.glm().

Value

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, 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

See Also

augment(), augment.lm()

stats::glm()

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

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

Details

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

Value

Errors unless x is a chi-squared test. If x is a chi-squared test, for each cell of the tested table returns columns:

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

See Also

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

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

---

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

Description

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

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

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

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

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model

formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

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

Usage

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

Arguments

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

Value

A tibble::tibble() containing the data passed to augment, and **additional** columns:

.fitted	The predicted response for that observation.
.resid	The residual for a particular point. Present only when data has been passed to augment via the data argument.

See Also

augment(), AER::ivreg()

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

Examples

```
library(AER)

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

summary(ivr)

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

augment(ivr)

glance(ivr)
```

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

Value

The original data as a tibble::tibble with one extra column:

.cluster	The cluster assigned by the k-means algorithm

See Also

augment(), stats::kmeans()

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

---

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

Description

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

Users may pass data to augment via either the data argument or the newdata argument. If the user passes data to the data argument, it **must** be exactly the data that was used to fit the model object.

Pass datasets to `newdata` to augment data that was not used during model fitting. This still requires that all columns used to fit the model are present.

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

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

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

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

Usage

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

Arguments

`x`	An `lm` object created by `stats::lm()`.
`data`	A `data.frame()` or `tibble::tibble()` containing the original data that was used to produce the object `x`. Defaults to `stats::model.frame(x)` so that `augment(my_fit)` returns the augmented original data. **Do not** pass new data to the `data` argument. Augment will report information such as influence and cooks distance for data passed to the `data` argument. These measures are only defined for the original training data.
`newdata`	A `data.frame()` or `tibble::tibble()` containing all the original predictors used to create `x`. Defaults to `NULL`, indicating that nothing has been passed to `newdata`. If `newdata` is specified, the `data` argument will be ignored.
`type.predict`	Type of predictions to use when `x` is a `glm` object. Passed to `stats::predict.glm()`.
`type.residuals`	Type of residuals to use when `x` is a `glm` object. Passed to `stats::residuals.glm()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be

used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

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.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, 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

See Also

na.action

`augment()`, `stats::predict.lm()`

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

augment.loess *Tidy a(n) loess object*

Description

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

Usage

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

Arguments

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

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

`augment()`, `stats::loess()`

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))
```

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

Description

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

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

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

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

The augmented dataset is always returned as a tibble::tibble with the **same number of rows** as the passed dataset. This means that the passed data must be coercible to a tibble. At this time, tibbles do not support matrix-columns. This means you should not specify a matrix of covariates in a model

formula during the original model fitting process, and that splines::ns(), stats::poly() and survival::Surv() objects are not supported in input data. If you encounter errors, try explicitly passing a tibble, or fitting the original model on data in a tibble.

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

Usage

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

Arguments

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

Value

A tibble::tibble of the original data with two extra columns:

.class	The class assigned by the Mclust algorithm
.uncertainty	The uncertainty associated with the classification. If a point has a probability of 0.9 of being in its assigned class under the model, then the uncertainty is 0.1.

See Also

augment(), mclust::Mclust()

Other mclust tidiers: tidy.Mclust()

augment.nlrq *Tidy a(n) nlrq object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble() containing the data passed to augment, and **additional** columns:

.fitted	The predicted response for that observation.
.resid	The residual for a particular point. Present only when data has been passed to augment via the data argument.

See Also

augment(), quantreg::nlrq()

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

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

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

Value

A `tibble::tibble()` containing the data passed to `augment`, and **additional** columns:

`.fitted` The predicted response for that observation.

`.resid` The residual for a particular point. Present only when data has been passed to `augment` via the `data` argument.

See Also

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

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

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

Value

A tibble::tibble() containing the data passed to augment, and **additional** columns:

.fitted	The predicted response for that observation.
.resid	The residual for a particular point. Present only when data has been passed to augment via the data argument.

See Also

augment(), plm::plm()

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

---

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

Description

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

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

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

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

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

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

Usage

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

Arguments

`x`	A `poLCA` object returned from `poLCA::poLCA()`.
`data`	The original dataset used to fit the latent class model, as a tibble or data. If not given, uses manifest variables in `x$y` and, if applicable, covariates in `x$x`
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one row for each original observation, with additional 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.

See Also

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

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

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

Value

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

See Also

stats::prcomp(), svd_tidiers

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

---

augment.rlm *Augment a(n) rlm object*

Description

This augment method wraps augment.lm().

Usage

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

Arguments

x	An rlm object returned by MASS::rlm().
...	Arguments passed on to augment.lm
	data A data.frame() or tibble::tibble() containing the original data that was used to produce the object x. Defaults to stats::model.frame(x) so that augment(my_fit) returns the augmented original data. **Do not** pass new data to the data argument. Augment will report information such as influence and cooks distance for data passed to the data argument. These measures are only defined for the original training data.
	newdata A data.frame() or tibble::tibble() containing all the original predictors used to create x. Defaults to NULL, indicating that nothing has been passed to newdata. If newdata is specified, the data argument will be ignored.
	type.predict Type of predictions to use when x is a glm object. Passed to stats::predict.glm().
	type.residuals Type of residuals to use when x is a glm object. Passed to stats::residuals.glm().

Value

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, 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

See Also

augment(), augment.lm()

MASS::rlm()

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

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

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

object object of class rq or rqs or rq.process produced by rq

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

level converage probability for the 'confidence' intervals.

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

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

Value

A tibble::tibble with one row per obseration and columns:

.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

predict.rq does not provide confidence intervals when newdata is provided.

See Also

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

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

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

`...` Arguments passed on to `quantreg::predict.rqs`

`object` object of class rq or rqs or rq.process produced by `rq`

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

`stepfun` If 'TRUE' return stepfunctions otherwise return matrix of predictions. these functions can be estimates of either the conditional quantile or distribution functions depending upon the `type` argument. When `stepfun = FALSE` a matrix of point estimates of the conditional quantile function at the points specified by the `newdata` argument.

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

Value

A tibble::tibble with one row per obseration and columns:

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

Depending on the arguments passed on to `predict.rqs` 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

`predict.rqs` does not provide confidence intervals when `newdata` is provided.

See Also

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

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

`augment.smooth.spline` *Tidy a(n) smooth.spline object*

Description

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

Usage

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

Arguments

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

Value

A `tibble::tibble()` containing the data passed to `augment`, and **additional** columns:

`.fitted`	The predicted response for that observation.
`.resid`	The residual for a particular point. Present only when data has been passed to `augment` via the `data` argument.

See Also

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

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

Examples

```
spl <- smooth.spline(mtcars$wt, mtcars$mpg, df = 4)
augment(spl, mtcars)
augment(spl)  # calls original columns x and y

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

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

Value

A tibble::tibble containing the original data and one additional column .fitted.

See Also

speedglm::speedlm()

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

---

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

Description

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

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

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

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

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

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

Usage

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

Arguments

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

Value

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

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

See Also

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

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

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

Description

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

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

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

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

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

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

Usage

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

Arguments

`x`	An `survreg` object returned from `survival::survreg()`.

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

Details

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

Value

A tibble::tibble with the passed data and additional columns:

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

See Also

na.action

augment(), survival::survreg()

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

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

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

---

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.

bootstrap *Set up bootstrap replicates of a dplyr operation*

Description

The `bootstrap()` function is deprecated and will be removed from an upcoming release of broom. For tidy resampling, please use the rsample package instead.

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

Examples

```
## Not run:
library(dplyr)
mtcars %>% bootstrap(10) %>% do(tidy(lm(mpg ~ wt, .)))

## End(Not run)
```

brms_tidiers *Tidying methods for a brms model*

Description

`brms` tidiers will soon be deprecated in `broom` and there is no ongoing development of these functions at this time. `brms` tidiers are being developed in the `broom.mixed` package, which is not yet on CRAN.

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 brms::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\ lies within the corresponding interval. Only used if intervals = TRUE.
...	Extra arguments, not used

Details

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

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::brms()`, `brms::brmsfit()`

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 Objects into Tidy Tibbles*

Description

Convert statistical analysis objects from R into tidy tibbles, 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.

column_glossary *Allowed column names in tidied tibbles*

Description

Allowed column names in tidied tibbles

Usage

```
column_glossary
```

Format

A tibble with 4 variables:

method One of "glance", "augment" or "tidy".

column Character name of allowed output column.

description Character description of expected column contents.

used_by A list of character vectors detailing the *classes* that use the column when tidied. For example `c("Arima","betareg")`.

Examples

```
column_glossary
```

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 follows 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`	A function to compute a confidence interval for `x`. Calling `func(x,level = conf.level,...)` must return an object coercable to a tibble. This dataframe like object should have to columns corresponding the lower and upper bounds on the confidence interval.
`...`	extra arguments passed on to `confint`

Details

confint_tidy

Value

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

See Also

confint

---

data.frame_tidiers *Tidiers for data.frame objects*

Description

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

Usage

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

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

Arguments

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

Details

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

Value

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

`column`	name of original column
`n`	Number of valid (non-NA) values
`mean`	mean
`sd`	standard deviation
`median`	median
`trimmed`	trimmed mean, with trim defaulting to .1
`mad`	median absolute deviation (from the median)
`min`	minimum value
`max`	maximum value
`range`	range
`skew`	skew
`kurtosis`	kurtosis
`se`	standard error

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

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

Author(s)

David Robinson, Benjamin Nutter

Source

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

Examples

```
## Not run:
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()

## End(Not run)
```

`durbinWatsonTest_tidiers` *Tidy/glance a(n) durbinWatsonTest object*

Description

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

Usage

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

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

Arguments

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

Value

A one-row tibble::tibble with columns:

`statistic`	Test statistic for Durbin-Watson test.
`p.value`	P-value of test statistic.
`autocorrelation`	Residual autocorrelations.
`method`	Always `"Durbin-Watson Test"`.
`alternative`	Alternative hypothesis (character).

See Also

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

Examples

```
dw <- car::durbinWatsonTest(lm(mpg ~ wt, data = mtcars))
tidy(dw)
glance(dw)  # same output for all durbinWatsonTests
```

---

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
`...`	Additional arguments passed to `emmeans::summary.emmGrid()` or `lsmeans::summary.ref.grid()`. **Cautionary note**: mispecified arguments may be silently ignored!

Details

There are a large number of arguments that can be passed on to `emmeans::summary.emmGrid()` or `lsmeans::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)
  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))
}
```

`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

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

glance.aareg *Glance at a(n) aareg object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

glance(), survival::aareg()

Other aareg tidiers: tidy.aareg()

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

---

glance.Arima *Glance at a(n) Arima object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with 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

stats::arima()

Other Arima tidiers: tidy.Arima()

---

glance.betareg *Glance at a(n) betareg object*

Description

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

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

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

Usage

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

Arguments

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

Value

glance returns a one-row tibble with 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

See Also

glance(), betareg::betareg()

Examples

```
library(betareg)

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

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

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

augment(mod)

glance(mod)
```

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

Description

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

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

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

Usage

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

Arguments

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

Value

`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

See Also

`glance()`, `biglm::biglm()`, `biglm::bigglm()`

Other biglm tidiers: `tidy.biglm()`

`glance.binDesign` *Glance at a(n) binDesign object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

`glance()`, `binGroup::binDesign()`

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

Examples

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

    glance(des)
    tidy(des)

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

`glance.cch` *Glance at a(n) cch object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

glance(), survival::cch()

Other cch tidiers: glance.survfit(), tidy.cch()

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

---

glance.coxph *Glance at a(n) coxph object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns: TODO.

See Also

glance(), survival::coxph()

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

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

---

glance.cv.glmnet *Glance at a(n) cv.glmnet object*

Description

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

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

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

Usage

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

Arguments

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

Value

A tibble::tibble with one-row with columns:

lambda.min	The value of the penalization parameter lambda that achieved minimum loss as estimated by cross validation.
lambda.1se	The value of the penalization parameter lambda that results in the sparsest model while remaining within one standard error of the minimum loss.

See Also

glance(), glmnet::cv.glmnet()

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

---

glance.ergm *Glance at a(n) ergm object*

Description

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

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

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

Usage

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

Arguments

x	An ergm object returned from a call to ergm::ergm().
deviance	Logical indicating whether or not to report null and residual deviance for the model, as well as degrees of freedom. Defaults to FALSE.
mcmc	Logical indicating whether or not to report MCMC interval, burn-in and sample size used to estimate the model. Defaults to FALSE.
...	Additional arguments to pass to ergm::summary(). **Cautionary note**: Mispecified arguments may be silently ignored.

Value

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

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

If deviance = TRUE, and if the model supports it, the 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

See Also

glance(), ergm::ergm(), ergm::summary.ergm()

Other ergm tidiers: tidy.ergm()

---

glance.factanal *Glance at a(n) factanal object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

n.factors	The number of fitted factors
total.variance	Total cumulative proportion of variance accounted for by all factors
statistic	Significance-test statistic
p.value	p-value from the significance test, describing whether the covariance matrix estimated from the factors is significantly different from the observed covariance matrix
df	Degrees of freedom used by the factor analysis
n	Sample size used in the analysis
method	The estimation method; always Maximum Likelihood, "mle"
converged	Whether the factor analysis converged

See Also

glance(), stats::factanal()

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

---

glance.felm *Glance at a(n) felm object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with 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

`glance.fitdistr` *Glance at a(n) fitdistr object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

`tidy()`, `MASS::fitdistr()`

Other fitdistr tidiers: `tidy.fitdistr()`

`glance.Gam` *Glance at a(n) Gam object*

Description

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

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

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

Usage

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

Arguments

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

Details

Glance at `gam` objects created by calls to `mgcv::gam()` with `glance.gam()`.

Value

A one-row tibble::tibble with columns:

`logLik`	Log-likelihood of the model.
`AIC`	Akaike's Information Criterion for the model.
`BIC`	Bayesian Information Criterion for the model.
`deviance`	Deviance of the model.
`df.residual`	Residual degrees of freedom for the model.

See Also

glance(), gam::gam()

Other gam tidiers: tidy.Gam()

---

glance.gam *Glance at a(n) gam object*

Description

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

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

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

Usage

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

Arguments

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

Details

To glance Gam objects created by calls to gam::gam(), see glance.Gam().

Value

A one-row tibble::tibble with columns:

logLik	Log-likelihood of the model.
AIC	Akaike's Information Criterion for the model.
BIC	Bayesian Information Criterion for the model.
deviance	Deviance of the model.
df.residual	Residual degrees of freedom for the model.

See Also

glance(), mgcv::gam(), glance.Gam()

Other mgcv tidiers: tidy.gam()

---

glance.garch *Tidy a(n) garch object*

Description

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

Usage

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

Arguments

x	A garch object returned by tseries::garch().
test	Character specification of which hypothesis test to use. The garch function reports 2 hypothesis tests: Jarque-Bera to residuals and Box-Ljung to squared residuals.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A one-row tibble::tibble with columns:

statistic	Test statistic used to compute the p-value
p.value	P-value
parameter	Parameter field in the htest, typically degrees of freedom
method	Method used to compute the statistic as a string
logLik	the data's log-likelihood under the model
AIC	the Akaike Information Criterion
BIC	the Bayesian Information Criterion

See Also

glance(), tseries::garch(), []

Other garch tidiers: tidy.garch()

---

glance.glm *Glance at a(n) glm object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with 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

stats::glm()

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

Examples

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

`glance.glmnet` *Glance at a(n) glmnet object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

glance(), glmnet::glmnet()

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

---

glance.gmm *Glance at a(n) gmm object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with 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 x.

See Also

glance(), gmm::gmm()

Other gmm tidiers: tidy.gmm()

`glance.ivreg` *Glance at a(n) ivreg object*

Description

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

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

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

Usage

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

Arguments

`x`	An `ivreg` object created by a call to `AER::ivreg()`.
`diagnostics`	Logical indicating whether to include statistics and p-values for Sargan, Wu-Hausman and weak instrument tests. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A one-row tibble with 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`	Wald test statistic
`p.value`	p-value from the Wald test
`df`	Degrees of freedom used by the coefficients
`df.residual`	residual degrees of freedom

If `diagnostics = TRUE`, will also return the following columns:

`statistic.Sargan`
: Statistic for Sargan test

`p.value.Sargan`
: P-value for Sargan test

`statistic.Wu.Hausman`
: Statistic for Wu-Hausman test

`p.value.Wu.Hausman`
: P-value for Wu-Hausman test

`statistic.weakinst`
: Statistic for Wu-Hausman test

`p.value.weakinst`
: P-value for weak instruments test

See Also

`glance()`, `AER::ivreg()`

Other ivreg tidiers: `augment.ivreg()`, `tidy.ivreg()`

Examples

```
library(AER)

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

summary(ivr)

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

augment(ivr)

glance(ivr)
```

---

glance.kmeans *Glance at a(n) kmeans object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with 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

`glance()`, `stats::kmeans()`

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

`glance.lavaan` *Glance at a(n) lavaan object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

`chisq`	Model chi squared
`npar`	Number of parameters in the model
`rmsea`	Root mean square error of approximation
`rmsea.conf.high`	95 percent upper bound on RMSEA
`srmr`	Standardised root mean residual
`agfi`	Adjusted goodness of fit
`cfi`	Comparative fit index
`tli`	Tucker Lewis index
`aic`	Akaike information criterion

bic	Bayesian information criterion
ngroups	Number of groups in model
nobs	Number of observations included
norig	Number of observation in the original dataset
nexcluded	Number of excluded observations
converged	Logical - Did the model converge
estimator	Estimator used
missing_method	Method for eliminating missing data

For further recommendations on reporting SEM and CFA models see Schreiber, J. B. (2017). Update to core reporting practices in structural equation modeling. Research in Social and Administrative Pharmacy, 13(3), 634-643. https://doi.org/10.1016/j.sapharm.2016.06.006

See Also

glance(), lavaan::cfa(), lavaan::sem(), lavaan::fitmeasures()

Other lavaan tidiers: tidy.lavaan()

Examples

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

 library(lavaan)

 cfa.fit <- cfa(
   'F =~ x1 + x2 + x3 + x4 + x5',
   data = HolzingerSwineford1939, group = "school"
 )
 glance(cfa.fit)

}
```

---

glance.lm *Glance at a(n) lm object*

Description

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

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

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

Usage

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

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

glance()

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

`glance.lmodel2` *Glance at a(n) lmodel2 object*

Description

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

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

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

Usage

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

Arguments

`x`	A `lmodel2` object returned by `lmodel2::lmodel2()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A one-row tibble::tibble with columns:

- `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

See Also

`glance()`, `lmodel2::lmodel2()`

Other lmodel2 tidiers: `tidy.lmodel2()`

`glance.Mclust` *Glance at a(n) Mclust object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with 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`	If the other model contains a noise component, the value of the hypervolume parameter. Otherwise `NA`.

`glance.muhaz` *Glance at a(n) muhaz object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

`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

See Also

`glance()`, `muhaz::muhaz()`

Other muhaz tidiers: `tidy.muhaz()`

`glance.multinom` *Glance at a(n) multinom object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

`glance()`, `nnet::multinom()`

Other multinom tidiers: `tidy.multinom()`

`glance.nlrq` *Glance at a(n) nlrq object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row `tibble::tibble()` with columns:

`tau`	quantile
`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

See Also

`glance()`, `quantreg::nlrq()`

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

`glance.nls` *Glance at a(n) nls object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with 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

tidy, stats::nls()

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

---

glance.orcutt *Glance at a(n) orcutt object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

`glance()`, `orcutt::cochrane.orcutt()`

Other orcutt tidiers: `tidy.orcutt()`

`glance.plm` *Glance at a(n) plm object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble 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

glance(), plm::plm()

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

---

glance.poLCA *Augment data with information from a(n) poLCA object*

Description

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

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

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

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

glance(), poLCA::poLCA()

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

---

glance.pyears *Glance at a(n) pyears object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

glance(), survival::pyears()

Other pyears tidiers: tidy.pyears()

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

---

glance.ridgelm *Glance at a(n) ridgelm object*

Description

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

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

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

Usage

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

Arguments

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

Details

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

Value

A one-row tibble::tibble with columns:

kHKB	modified HKB estimate of the ridge constant
kLW	modified L-W estimate of the ridge constant
lambdaGCV	choice of lambda that minimizes GCV

See Also

glance(), MASS::select.ridgelm(), MASS::lm.ridge()

Other ridgelm tidiers: tidy.ridgelm()

---

glance.rlm *Glance at a(n) rlm object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with 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

glance(), MASS::rlm()

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

Examples

```
library(MASS)

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

glance.rq *Glance at a(n) rq object*

Description

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

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

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

Usage

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

Arguments

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

Details

Only models with a single tau value may be passed. For multiple values, please use a purrr::map() workflow instead, e.g.

```
taus %>%
  map(function(tau_val) rq(y ~ x, tau = tau_val)) %>%
  map_dfr(glance)
```

Value

A one-row tibble::tibble with 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

See Also

glance(), quantreg::rq()

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

`glance.smooth.spline` *Tidy a(n) smooth.spine object*

Description

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

augment(), stats::smooth.spline()

Other smoothing spline tidiers: augment.smooth.spline()

`glance.speedlm` *Glance at a(n) speedlm object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble 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
`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

speedglm::speedlm()

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

`glance.survdiff` *Glance at a(n) survdiff object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

statistic	value of the test statistic
df	degrees of freedom
p.value	p-value

See Also

glance(), survival::survdiff()

Other survdiff tidiers: tidy.survdiff()

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

`glance.survexp` *Glance at a(n) survexp object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

glance(), survival::survexp()

Other survexp tidiers: tidy.survexp()

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

---

glance.survfit *Glance at a(n) survfit object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

records	number of observations
n.max	n.max
n.start	n.start

events	number of events
rmean	Restricted mean (see survival::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

See Also

glance(), survival::survfit()

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

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

---

glance.survreg *Glance at a(n) survreg object*

Description

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

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

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

glance(), survival::survreg()

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

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

---

glance_optim *Tidy a(n) optim object masquerading as list*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, stats::optim(), svd() and akima::interp() produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves are implemented as functions of the form tidy_<function> or glance_<function> and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.

Usage

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

Arguments

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

Value

A one-row tibble::tibble with 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

glance(), optim()

Other list tidiers: list_tidiers, tidy_irlba(), tidy_optim(), tidy_svd(), tidy_xyz()

---

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

list_tidiers *Tidying methods for lists / returned values that are not S3 objects*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, `stats::optim()`, `svd()` and `akima::interp()` produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

Usage

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

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

Arguments

x	A list, potentially representing an object that can be tidied.
...	Additionally arguments passed to the tidying function.

Details

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves are implemented as functions of the form tidy_<function> or glance_<function> and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.

See Also

Other list tidiers: `glance_optim()`, `tidy_irlba()`, `tidy_optim()`, `tidy_svd()`, `tidy_xyz()`

lme4_tidiers *Tidying methods for mixed effects models*

Description

`lme4` tidiers will soon be deprecated in `broom` and there is no ongoing development of these functions at this time. `lme4` tidiers are being developed in the `broom.mixed` package, which is not yet on CRAN.

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

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

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", ".eta".

`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

```
## Not run:
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)
}

## End(Not run)
```

`matrix_tidiers` *Tidiers for matrix objects*

Description

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

Usage

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

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

Arguments

x	A matrix
...	extra arguments, not used

Details

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.

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

```
## Not run:
mat <- as.matrix(mtcars)
tidy(mat)
glance(mat)

## End(Not run)
```

---

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

Description

MCMC tidiers will soon be deprecated in `broom` and there is no ongoing development of these functions at this time. MCMC tidiers are being developed in the `broom.mixed` package, which is not yet on CRAN.

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))
}
```

nlme_tidiers *Tidying methods for mixed effects models*

Description

nlme tidiers will soon be deprecated in broom and there is no ongoing development of these functions at this time. nlme tidiers are being developed in the broom.mixed package, which is not yet on CRAN.

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

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

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

```
## Not run:
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)
}

## End(Not run)
```

`null_tidiers` *Tidiers for NULL inputs*

Description

`tidy(NULL)`, `glance(NULL)` and `augment(NULL)` all return an empty tibble::tibble. This empty tibble can be treated a tibble with zero rows, making it convenient to combine with other tibbles using functions like `purrr::map_df()` on lists of potentially `NULL` objects.

Usage

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

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

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

Arguments

`x`	The value `NULL`.
`...`	Additional arguments (not used).

Value

An empty tibble::tibble.

See Also

tibble::tibble

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

Description

Rowwise tidiers are deprecated and will be removed from an upcoming version of broom. We strongly recommend moving to a `nest-map-unnest` workflow over a `rowwise-do` workflow. See the vignettes for examples.

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

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.

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
```

---

rstanarm_tidiers *Tidying methods for an rstanarm model*

Description

rstanarm tidiers will soon be deprecated in broom and there is no ongoing development of these functions at this time.

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 rstanarm::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\ rstanarm::posterior_interval() for details.
prob	See rstanarm::posterior_interval().
...	For glance, if looic=TRUE, optional arguments to rstanarm::loo.stanreg().
looic	Should the LOO Information Criterion (and related info) be included? See rstanarm::loo.stanreg() for details. Note: for models fit to very large datasets this can be a slow computation.

Details

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

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 stats::mad(). See the *Uncertainty estimates* section in rstanarm::print.stanreg() for more details.

For models with group-specific parameters (e.g., models fit with rstanarm::stan_glmer()), setting parameters="varying" selects the group-level parameters instead of the non-varying regression coefficients. Additional 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 rstanarm::stan_glm.nb() the overdispersion parameter is included if parameters="aux", for rstanarm::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 `rstanarm::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

`rstanarm::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)
```

sparse_tidiers *Tidy a sparseMatrix object from the Matrix package*

Description

sparseMatrix tidiers are deprecated and will be removed from an upcoming version of broom.

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

Details

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

sp_tidiers *Tidy a(n) SpatialPolygonsDataFrame object*

Description

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

Note that the `sf` package now defines tidy spatial objects and is the recommend approach to spatial data. `sp` tidiers are likely to be deprecated in the near future in favor of `sf::st_as_sf()`. Development of `sp` tidiers has halted in `broom`.

Usage

```
## S3 method for class 'SpatialPolygonsDataFrame'
tidy(x, region = NULL, ...)

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

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

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

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

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

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

Arguments

x	A SpatialPolygonsDataFrame, SpatialPolygons, Polygons, Polygon, SpatialLinesDataFrame, Lines or Line object.
region	name of variable used to split up regions
...	not used by this method

---

summary_tidiers *Tidy/glance a(n) summaryDefault object*

Description

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

Usage

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

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

Arguments

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

Value

A one-row tibble::tibble with columns:

minimum	Minimum value in original vector.
q1	First quartile of original vector.
median	Median of original vector.
mean	Mean of original vector.
q3	Third quartile of original vector.
maximum	Maximum value in original vector.
na	Number of NA values in original vector. Column present only when original vector had at least one NA entry.

See Also

tidy(), summary()

Examples

```
v <- rnorm(1000)
s <- summary(v)
s

tidy(s)
glance(s)

v2 <- c(v,NA)
tidy(summary(v2))
```

`tidy.aareg` *Tidy a(n) aareg object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with one row for each coefficient and 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

See Also

`tidy()`, `survival::aareg()`

Other aareg tidiers: `glance.aareg()`

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

Examples

```
library(survival)

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

tidy(afit)
```

`tidy.acf` *Tidy a(n) acf object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with columns:

`lag`	lag values
`acf`	calculated correlation

See Also

tidy(), stats::acf(), stats::pacf(), stats::ccf()

Other time series tidiers: tidy.spec(), tidy.ts(), tidy.zoo()

Examples

```
tidy(acf(lh, plot = FALSE))
tidy(ccf(mdeaths, fdeaths, plot = FALSE))
tidy(pacf(lh, plot = FALSE))
```

---

tidy.anova *Tidy a(n) anova object*

Description

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

Usage

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

Arguments

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

Details

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

Value

A tibble::tibble 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

See Also

tidy(), stats::anova(), car::Anova()

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

Examples

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

---

tidy.aov *Tidy a(n) aov object*

Description

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

Usage

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

Arguments

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

Details

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

Value

A tibble::tibble 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

See Also

`tidy()`, `stats::aov()`

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

Examples

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

---

tidy.aovlist *Tidy a(n) aovlist object*

Description

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

Usage

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

Arguments

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

Details

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

Value

A tibble::tibble 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
stratum	The error stratum

See Also

tidy(), stats::aov()

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

Examples

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

`tidy.Arima` *Tidy a(n) Arima object*

Description

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

Usage

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

Arguments

`x`	An object of class `Arima` created by `stats::arima()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one row for each coefficient and 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

See Also

stats::arima()

Other Arima tidiers: glance.Arima()

Examples

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

---

tidy.betareg *Tidy a(n) betareg object*

Description

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

Usage

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

Arguments

x	A betareg object produced by a call to betareg::betareg().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with one row for each term in the regression. The tibble has columns:

term	The name of the regression term.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
statistic	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
p.value	The two-sided p-value associated with the observed statistic.
conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

In additional the standard columns, the returned tibble has an additional column component. component indicates whether a particular term was used to model either the "mean" or "precision". Here the precision is the inverse of the variance, often referred to as phi. At least one term will have been used to model phi.

See Also

tidy(), betareg::betareg()

Examples

```
library(betareg)

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

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

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

augment(mod)

glance(mod)
```

`tidy.biglm` *Tidy a(n) biglm object*

Description

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

Usage

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

Arguments

`x`	A `biglm` object created by a call to `biglm::biglm()` or `biglm::bigglm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`quick`	Logical indiciating if the only the `term` and `estimate` columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble() with one row for each term in the regression. The tibble has columns:

term	The name of the regression term.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
statistic	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
p.value	The two-sided p-value associated with the observed statistic.
conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

See Also

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

Other biglm tidiers: glance.biglm()

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)
}
```

`tidy.binDesign` *Tidy a(n) binDesign object*

Description

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

`n`	Number of trials in given iteration.
`power`	Power achieved for given value of n.

See Also

`tidy()`, `binGroup::binDesign()`

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

Examples

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

    glance(des)
    tidy(des)
```

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

`tidy.binWidth` *Tidy a(n) binWidth object*

Description

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

Usage

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

Arguments

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

Value

A one-row tibble::tibble with columns:

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

See Also

`tidy()`, `binGroup::binWidth()`

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

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")
}
```

`tidy.boot` *Tidy a(n) boot object*

Description

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

Usage

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

Arguments

`x`	A `boot::boot()` object.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`conf.method`	Passed to the `type` argument of `boot::boot.ci()`. Defaults to `"perc"`.

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

Value

A tibble::tibble with one row per bootstrapped statistic and columns:

`term`	Name of the computed statistic, if present.
`statistic`	Original value of the statistic.
`bias`	Bias of the statistic.
`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.

See Also

`tidy()`, `boot::boot()`, `boot::tsboot()`, `boot::boot.ci()`, `rsample::bootstraps()`

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)
}
```

`tidy.btergm` *Tidy a(n) btergm object*

Description

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

This method tidies the coefficients of a bootstrapped temporal exponential random graph model estimated with the **xergm**. It simply returns the coefficients and their confidence intervals.

Usage

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

Arguments

`x`	A `btergm::btergm()` object.
`conf.level`	Confidence level for confidence intervals. Defaults to 0.95.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`quick`	Logical indiciating if the only the `term` and `estimate` columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one row per term in the random graph model and columns:

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

See Also

tidy(), btergm::btergm()

Examples

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

# Create 10 random networks with 10 actors

networks <- list()

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

# Create 10 matrices as covariates

covariates <- list()

for (i in 1:10) {
    mat <- matrix(rnorm(100), nrow = 10, ncol = 10)
    covariates[[i]] <- mat
}

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

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

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

---

tidy.cch *Tidy a(n) cch object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble() with one row for each term in the regression. The tibble has columns:

term	The name of the regression term.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
statistic	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
p.value	The two-sided p-value associated with the observed statistic.
conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

See Also

tidy(), survival::cch()

Other cch tidiers: glance.cch(), glance.survfit()

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

Examples

```
library(survival)

# examples come from cch documentation
```

```
subcoh <- nwtco$in.subcohort
selccoh <- with(nwtco, rel==1|subcoh==1)
ccoh.data <- nwtco[selccoh,]
ccoh.data$subcohort <- subcoh[selccoh]
## central-lab histology
ccoh.data$histol <- factor(ccoh.data$histol,labels=c("FH","UH"))
## tumour stage
ccoh.data$stage <- factor(ccoh.data$stage,labels=c("I","II","III" ,"IV"))
ccoh.data$age <- ccoh.data$age/12 # Age in years

fit.ccP <- cch(Surv(edrel, rel) ~ stage + histol + age, data = ccoh.data,
               subcoh = ~subcohort, id= ~seqno, cohort.size = 4028)

tidy(fit.ccP)

# coefficient plot
library(ggplot2)
ggplot(tidy(fit.ccP), aes(x = estimate, y = term)) +
  geom_point() +
  geom_errorbarh(aes(xmin = conf.low, xmax = conf.high), height = 0) +
  geom_vline(xintercept = 0)
```

`tidy.cld` *Tidy a(n) cld object*

Description

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

Usage

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

Arguments

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

See Also

tidy(), multcomp::cld(), multcomp::summary.glht(), multcomp::confint.glht(), multcomp::glht()

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

`tidy.coeftest` *Tidy a(n) coeftest object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with one row for each coefficient and 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

See Also

tidy(), lmtest::coeftest()

Examples

```
if (require("lmtest", quietly = TRUE)) {
    data(Mandible)
    fm <- lm(length ~ age, data=Mandible, subset=(age <= 28))

    lmtest::coeftest(fm)
    tidy(coeftest(fm))
}
```

`tidy.confint.glht` *Tidy a(n) confint.glht object*

Description

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

Usage

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

Arguments

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

See Also

`tidy()`, `multcomp::confint.glht()`, `multcomp::glht()`

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

`tidy.confusionMatrix` *Tidy a(n) confusionMatrix object*

Description

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

Usage

```
## S3 method for class 'confusionMatrix'
tidy(x, by_class = TRUE, ...)
```

Arguments

`x`	An object of class `confusionMatrix` created by a call to `caret::confusionMatrix()`.
`by_class`	Logical indicating whether or not to show performance measures broken down by class. Defaults to `TRUE`. When `by_class = FALSE` only returns a tibble with accuracy and kappa statistics.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one or more of the following columns:

`term`	The name of a statistic from the confusion matrix
`class`	Which class the term is a measurement of
`estimate`	The value of the statistic
`conf.low`	Low end of 95 percent CI only applicable to accuracy
`conf.high`	High end of 95 percent CI only applicable to accuracy
`p.value`	P-value for accuracy and kappa statistics

See Also

`tidy()`, `caret::confusionMatrix()`

Examples

```
if (requireNamespace("caret", quietly = TRUE)) {

  set.seed(27)

  two_class_sample1 <- as.factor(sample(letters[1:2], 100, TRUE))
  two_class_sample2 <- as.factor(sample(letters[1:2], 100, TRUE))

  two_class_cm <- caret::confusionMatrix(
    two_class_sample1,
    two_class_sample2
  )

  tidy(two_class_cm)
  tidy(two_class_cm, by_class = FALSE)

  # multiclass example

  six_class_sample1 <- as.factor(sample(letters[1:6], 100, TRUE))
  six_class_sample2 <- as.factor(sample(letters[1:6], 100, TRUE))

  six_class_cm <- caret::confusionMatrix(
    six_class_sample1,
    six_class_sample2
  )

  tidy(six_class_cm)
  tidy(six_class_cm, by_class = FALSE)
}
```

---

tidy.coxph *Tidy a(n) coxph object*

Description

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

Usage

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

Arguments

x	A coxph object returned from survival::coxph().
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble with one row for each term and columns:

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

See Also

tidy(), survival::coxph()

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

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

Examples

```
library(survival)

cfit <- coxph(Surv(time, status) ~ age + sex, lung)

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

lp <- augment(cfit, lung)
```

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

glance(cfit)

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

logan2$case <- (logan2$occupation == logan2$tocc)

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

library(ggplot2)

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

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

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

`tidy.cv.glmnet` *Tidy a(n) cv.glmnet object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with one-row for each value of the penalization parameter lambda in x and columns:

lambda	Value of the penalty parameter lambda.
estimate	Median loss across all cross-validation folds for a given lambda.
std.error	Standard error of the cross-validation estimated loss.
conf.low	lower bound on confidence interval for cross-validation estimated loss.
conf.high	Upper bound on confidence interval for cross-validation estimated loss.
nzero	Number of coefficients that are exactly zero for given lambda

.

See Also

tidy(), glmnet::cv.glmnet()

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

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {

    library(glmnet)
    set.seed(27)

    nobs <- 100
    nvar <- 50
    real <- 5

    x <- matrix(rnorm(nobs * nvar), nobs, nvar)
    beta <- c(rnorm(real, 0, 1), rep(0, nvar - real))
    y <- c(t(beta) %*% t(x)) + rnorm(nvar, sd = 3)

    cvfit1 <- cv.glmnet(x,y)

    tidy(cvfit1)
    glance(cvfit1)
```

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

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

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

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

    # 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)
}
```

`tidy.density` *Tidy a(n) density object*

Description

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

Usage

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

Arguments

`x`	A `density` object returned from `stats::density()`.

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

Value

A tibble::tibble with two columns: points `x` where the density is estimated, and estimated density `y`.

See Also

`tidy()`, `stats::density()`

Other stats tidiers: `tidy.dist()`, `tidy.ftable()`

---

tidy.dist *Tidy a(n) dist object*

Description

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

Usage

```
## S3 method for class 'dist'
tidy(x, diagonal = attr(x, "Diag"), upper = attr(x, "Upper"), ...)
```

Arguments

`x`	A `dist` object returned from `stats::dist()`.
`diagonal`	Logical indicating whether or not to tidy the diagonal elements of the distance matrix. Defaults to whatever was based to the `diag` argument of `stats::dist()`.
`upper`	Logical indicating whether or not to tidy the upper half of the distance matrix. Defaults to whatever was based to the `upper` argument of `stats::dist()`.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

If the distance matrix does not include an upper triangle and/or diagonal, the tidied version will not either.

Value

A tibble::tibble with one row for each pair of items in the distance matrix, with columns:

`item1`	First item
`item2`	Second item
`distance`	Distance between items

See Also

`tidy()`, `stats::dist()`

Other stats tidiers: `tidy.density()`, `tidy.ftable()`

Examples

```
iris_dist <- dist(t(iris[, 1:4]))
iris_dist

tidy(iris_dist)
tidy(iris_dist, upper = TRUE)
tidy(iris_dist, diagonal = TRUE)
```

---

tidy.ergm *Tidy a(n) ergm object*

Description

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

The methods should work with any model that conforms to the **ergm** class, such as those produced from weighted networks by the **ergm.count** package.

Usage

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

Arguments

x	An ergm object returned from a call to ergm::ergm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
...	Additional arguments to pass to ergm::summary(). **Cautionary note**: Misspecified arguments may be silently ignored.

Value

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

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

References

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

tidy(), ergm::ergm(), ergm::control.ergm(), ergm::summary()

Other ergm tidiers: glance.ergm()

Examples

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

# Fit a model where the propensity to form ties between
# families depends on the absolute difference in wealth
gest <- ergm(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)
```

`tidy.factanal` *Tidy a(n) factanal object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with one row for each variable used in the analysis and columns:

`variable`	The variable being estimated in the factor analysis
`uniqueness`	Proportion of residual, or unexplained variance
`flX`	Factor loading of term on factor X. There will be as many columns of this format as there were factors fitted.

See Also

tidy(), stats::factanal()

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

Examples

```
mod <- factanal(mtcars, 3, scores = "regression")

glance(mod)
tidy(mod)
augment(mod)
augment(mod, mtcars)
```

---

tidy.felm *Tidy a(n) felm object*

Description

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

Usage

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

Arguments

x	A felm object returned from lfe::felm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
fe	Logical indicating whether or not to include estimates of fixed effects. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with one row for each term in the regression. The tibble has columns:

term	The name of the regression term.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
statistic	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
p.value	The two-sided p-value associated with the observed statistic.
conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

If fe = TRUE, also includes rows for for fixed effects estimates.

See Also

tidy(), lfe::felm()

Other felm tidiers: augment.felm()

Examples

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

    library(lfe)

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

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

```
    augment(result_felm)
    glance(result_felm)
}
```

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

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with one row for estimated parameter, with columns:

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

See Also

`tidy()`, `MASS::fitdistr()`

Other fitdistr tidiers: `glance.fitdistr()`

Examples

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

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

tidy(fit)
glance(fit)
```

`tidy.ftable` *Tidy a(n) ftable object*

Description

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

Usage

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

Arguments

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

Value

An ftable contains a "flat" contingency table. This melts it into a tibble::tibble with one column for each variable, then a `Freq` column.

See Also

`tidy()`, `stats::ftable()`

Other stats tidiers: `tidy.density()`, `tidy.dist()`

Examples

```
tidy(ftable(Titanic, row.vars = 1:3))
```

`tidy.Gam` *Tidy a(n) Gam object*

Description

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

Usage

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

Arguments

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

Details

Tidy `gam` objects created by calls to `mgcv::gam()` with `tidy.gam()`.

Value

The tidied output of the parametric ANOVA for the GAM model as a tibble::tibble with one row for each term in the model.

See Also

`tidy()`, `gam::gam()`, `tidy.anova()`, `tidy.gam()`

Other gam tidiers: `glance.Gam()`

Examples

```
library(gam)
g <- gam(mpg ~ s(hp, 4) + am + qsec, data = mtcars)

tidy(g)
glance(g)
```

`tidy.gam` *Tidy a(n) gam object*

Description

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

Usage

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

Arguments

`x`	A `gam` object returned from a call to `mgcv::gam()`.
`parametric`	Logical indicating if parametric or smooth terms should be tidied. Defaults to `FALSE`, meaning that smooth terms are tidied by default.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

To tidy `Gam` objects created by calls to `gam::gam()`, see `tidy.Gam()`.

See Also

`tidy()`, `mgcv::gam()`, `tidy.Gam()`

Other mgcv tidiers: `glance.gam()`

Examples

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

tidy(g)
tidy(g, parametric = TRUE)
glance(g)
```

`tidy.gamlss` *Tidy a(n) gamlss object*

Description

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

Usage

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

Arguments

`x`	A `gamlss` object returned from `gamlss::gamlss()`.
`quick`	Logical indiciating if the only the `term` and `estimate` columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one row for each coefficient, containing columns

`parameter`	Type of coefficient being estimated: `mu`, `sigma`, `nu`, or `tau`.
`term`	Name of term in the model.
`estimate`	Estimate coefficient of given term.

std.error	Standard error of given term.
statistic	T-statistic used to test hypothesis that coefficien equals zero.
p.value	Two sided p-value based on null hypothesis of coefficient equaling zero.

Examples

```
library(gamlss)

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

tidy(g)
```

---

tidy.garch *Tidy a(n) garch object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with one row for each coefficient and 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

See Also

tidy(), tseries::garch()

Other garch tidiers: glance.garch()

Examples

```
library(tseries)

data(EuStockMarkets)
dax <- diff(log(EuStockMarkets))[,"DAX"]
dax.garch <- garch(dax)
dax.garch

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

---

tidy.geeglm *Tidy a(n) geeglm object*

Description

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

Usage

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

Arguments

x	A geeglm object returned from a call to geepack::geeglm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
quick	Logical indiciating if the only the term and estimate columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

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

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

Value

A tibble::tibble with 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, includes includes columns conf.low and conf.high, which are computed internally.

See Also

tidy(), geepack::geeglm()

Examples

```
if (requireNamespace("geepack", quietly = TRUE)) {
  library(geepack)
  data(state)

  ds <- data.frame(state.region, state.x77)

  geefit <- geeglm(Income ~ Frost + Murder, id = state.region,
                   data = ds, family = gaussian,
                   corstr = "exchangeable")

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

`tidy.glht` *Tidy a(n) glht object*

Description

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

Usage

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

Arguments

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

See Also

`tidy()`, `multcomp::glht()`

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

Examples

```
if (require("multcomp") && require("ggplot2")) {

    library(multcomp)
    library(ggplot2)

    amod <- aov(breaks ~ wool + tension, data = warpbreaks)
    wht <- glht(amod, linfct = mcp(tension = "Tukey"))

    tidy(wht)
    ggplot(wht, aes(lhs, estimate)) + geom_point()

    CI <- confint(wht)
    tidy(CI)
    ggplot(CI, aes(lhs, estimate, ymin = lwr, ymax = upr)) +
       geom_pointrange()

    tidy(summary(wht))
    ggplot(mapping = aes(lhs, estimate)) +
       geom_linerange(aes(ymin = lwr, ymax = upr), data = CI) +
       geom_point(aes(size = p), data = summary(wht)) +
       scale_size(trans = "reverse")

    cld <- cld(wht)
    tidy(cld)
}
```

---

tidy.glm *Tidy a(n) glm object*

Description

This method wraps `tidy.lm()`.

Usage

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

Arguments

x	A glm object returned from `stats::glm()`.
...	Arguments passed on to `tidy.lm`
	`conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.

`conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

`quick` Logical indiciating if the only the `term` and `estimate` columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to `FALSE`.

`exponentiate` Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.

Value

A `tibble::tibble()` with one row for each term in the regression. The tibble has columns:

`term`	The name of the regression term.
`estimate`	The estimated value of the regression term.
`std.error`	The standard error of the regression term.
`statistic`	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
`p.value`	The two-sided p-value associated with the observed statistic.
`conf.low`	The low end of a confidence interval for the regression term. Included only if `conf.int = TRUE`.
`conf.high`	The high end of a confidence interval for the regression term. Included only if `conf.int = TRUE`.

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)

See Also

`tidy()`, `tidy.lm()`

`stats::glm()`

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

`tidy.glmnet` *Tidy a(n) glmnet object*

Description

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

Usage

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

Arguments

x	A glmnet object returned from glmnet::glmnet().
return_zeros	Logical indicating whether coefficients with value zero zero should be included in the results. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Details

Note that while this representation of GLMs is much easier to plot and combine than the default structure, it is also much more memory-intensive. Do not use for large, sparse matrices.

No augment method is yet provided even though the model produces predictions, because the input data is not tidy (it is a matrix that may be very wide) and therefore combining predictions with it is not logical. Furthermore, predictions make sense only with a specific choice of lambda.

Value

A tibble::tibble with columns:

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

See Also

tidy(), glmnet::glmnet()

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

Examples

```
if (requireNamespace("glmnet", quietly = TRUE)) {

    library(glmnet)
```

```
    set.seed(2014)
    x <- matrix(rnorm(100*20),100,20)
    y <- rnorm(100)
    fit1 <- glmnet(x,y)

    tidy(fit1)
    glance(fit1)

    library(dplyr)
    library(ggplot2)

    tidied <- tidy(fit1) %>% filter(term != "(Intercept)")

    ggplot(tidied, aes(step, estimate, group = term)) + geom_line()
    ggplot(tidied, aes(lambda, estimate, group = term)) +
        geom_line() + scale_x_log10()

    ggplot(tidied, aes(lambda, dev.ratio)) + geom_line()

    # works for other types of regressions as well, such as logistic
    g2 <- sample(1:2, 100, replace=TRUE)
    fit2 <- glmnet(x, g2, family="binomial")
    tidy(fit2)
}
```

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

Description

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

Usage

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

Arguments

x	A gmm object returned from gmm::gmm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
quick	Logical indiciating if the only the term and estimate columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with one row for each term in the regression. The tibble has columns:

term	The name of the regression term.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
statistic	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
p.value	The two-sided p-value associated with the observed statistic.
conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

See Also

tidy(), gmm::gmm()

Other gmm tidiers: glance.gmm()

Examples

```
if (requireNamespace("gmm", quietly = TRUE)) {

  library(gmm)

  # examples come from the "gmm" package
  ## CAPM test with GMM
  data(Finance)
  r <- Finance[1:300, 1:10]
  rm <- Finance[1:300, "rm"]
  rf <- Finance[1:300, "rf"]

  z <- as.matrix(r-rf)
  t <- nrow(z)
  zm <- rm-rf
  h <- matrix(zm, t, 1)
  res <- gmm(z ~ zm, x = h)

  # tidy result
  tidy(res)
  tidy(res, conf.int = TRUE)
  tidy(res, conf.int = TRUE, conf.level = .99)

  # coefficient plot
  library(ggplot2)
  library(dplyr)
  tidy(res, conf.int = TRUE) %>%
    mutate(variable = reorder(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)
}
```

`tidy.htest` *Tidy/glance a(n) htest object*

Description

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

Usage

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

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

Arguments

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

Value

A one-row tibble::tibble with one or more of the following columns, depending on which hypothesis test was used.

`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

See Also

`tidy()`, `stats::cor.test()`, `stats::t.test()`, `stats::wilcox.test()`, `stats::chisq.test()`

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

Examples

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

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

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

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

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

`tidy.ivreg` *Tidy a(n) ivreg object*

Description

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

Usage

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

Arguments

`x`	An `ivreg` object created by a call to `AER::ivreg()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with one row for each term in the regression. The tibble has columns:

`term`	The name of the regression term.
`estimate`	The estimated value of the regression term.
`std.error`	The standard error of the regression term.
`statistic`	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
`p.value`	The two-sided p-value associated with the observed statistic.

conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

See Also

tidy(), AER::ivreg()

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

Examples

```
library(AER)

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

summary(ivr)

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

augment(ivr)

glance(ivr)
```

---

tidy.kappa *Tidy a(n) kappa object*

Description

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

Usage

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

Arguments

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

Details

Note that confidence level (alpha) for the confidence interval cannot be set in tidy. Instead you must set the alpha argument to psych::cohen.kappa() when creating the kappa object.

Value

A tibble::tibble with columns:

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

tidy(), psych::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))
```

`tidy.kde` *Tidy a(n) kde object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble 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.

See Also

`tidy()`, `ks::kde()`

Examples

```
if (requireNamespace("ks", quietly = TRUE)) {

  library(ks)

  dat <- replicate(2, rnorm(100))
  k <- kde(dat)

  td <- tidy(k)
  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)
  td3
}
```

`tidy.Kendall` *Tidy a(n) Kendall object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with one row and columns:

`statistic`	Kendall's tau statistic.
`p.value`	two-sided p-value.
`kendall_score`	Kendall score.

`denominator` The denominator, which is tau=kendall_score/denominator.

`var_kendall_score`
Variance of the kendall_score.

See Also

`tidy()`, `Kendall::Kendall()`, `Kendall::MannKendall()`, `Kendall::SeasonalMannKendall()`

Examples

```
library(Kendall)

A <- c(2.5,2.5,2.5,2.5,5,6.5,6.5,10,10,10,10,10,14,14,14,16,17)
B <- c(1,1,1,1,2,1,1,2,1,1,1,1,1,1,2,2,2)

f_res <- Kendall(A, B)
tidy(f_res)

s_res <- MannKendall(B)
tidy(s_res)

t_res <- SeasonalMannKendall(ts(A))
tidy(t_res)
```

`tidy.kmeans` *Tidy a(n) kmeans object*

Description

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

Usage

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

Arguments

`x`	A kmeans object created by `stats::kmeans()`.
`col.names`	Dimension names. Defaults to x1, x2, ...
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed

using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

For examples, see the kmeans vignette.

Value

A tibble::tibble with one row per cluster, and columns:

`size`	Number of points in cluster
`withinss`	The within-cluster sum of squares
`cluster`	A factor describing the cluster from 1:k

See Also

`tidy()`, `stats::kmeans()`

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

`tidy.lavaan` *Tidy a(n) lavaan object*

Description

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

Usage

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

Arguments

`x`	A `lavaan` object, such as those return from `lavaan::cfa()`, and `lavaan::sem()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments passed to `lavaan::parameterEstimates()`. **Cautionary note**: Misspecified arguments may be silently ignored.

Value

A tibble::tibble with one row for each estimated parameter and columns:

term	The result of paste(lhs, op, rhs)
op	The operator in the model syntax (e.g. ~~ for covariances, or ~ for regression parameters)
group	The group (if specified) in the lavaan model
estimate	The parameter estimate (may be standardized)
std.error	
statistic	The z value returned by lavaan::parameterEstimates()
p.value	
conf.low	
conf.high	
std.lv	Standardized estimates based on the variances of the (continuous) latent variables only
std.all	Standardized estimates based on both the variances of both (continuous) observed and latent variables.
std.nox	Standardized estimates based on both the variances of both (continuous) observed and latent variables, but not the variances of exogenous covariates.

See Also

tidy(), lavaan::cfa(), lavaan::sem(), lavaan::parameterEstimates()

Other lavaan tidiers: glance.lavaan()

Examples

```
if (require("lavaan")) {

 library(lavaan)

 cfa.fit <- cfa('F =~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9',
                data = HolzingerSwineford1939, group = "school")
 tidy(cfa.fit)
}
```

`tidy.lm` *Tidy a(n) lm object*

Description

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

Usage

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

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

Arguments

`x`	An `lm` object created by `stats::lm()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`exponentiate`	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.
`quick`	Logical indiciating if the only the `term` and `estimate` columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

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

Value

A `tibble::tibble()` with one row for each term in the regression. The tibble has columns:

`term`	The name of the regression term.
`estimate`	The estimated value of the regression term.
`std.error`	The standard error of the regression term.
`statistic`	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
`p.value`	The two-sided p-value associated with the observed statistic.
`conf.low`	The low end of a confidence interval for the regression term. Included only if `conf.int = TRUE`.
`conf.high`	The high end of a confidence interval for the regression term. Included only if `conf.int = TRUE`.

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)

See Also

`tidy()`, `stats::summary.lm()`

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

Examples

```
library(ggplot2)
library(dplyr)

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

tidy(mod)
glance(mod)

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

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

```
    geom_errorbarh()

augment(mod)
augment(mod, mtcars)

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

au <- augment(mod, data = mtcars)

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

plot(mod, which = 6)
ggplot(au, aes(.hat, .cooksd)) +
  geom_vline(xintercept = 0, colour = NA) +
  geom_abline(slope = seq(0, 3, by = 0.5), colour = "white") +
  geom_smooth(se = FALSE) +
  geom_point()

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

`tidy.lmodel2` *Tidy a(n) lmodel2 object*

Description

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

Usage

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

Arguments

x	A `lmodel2` object returned by `lmodel2::lmodel2()`.

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

Details

There are always only two terms in an `lmodel2`: `"Intercept"` and `"Slope"`. These are computed by four methods: OLS (ordinary least squares), MA (major axis), SMA (standard major axis), and RMA (ranged major axis).

Value

A tibble::tibble within eight rows (one for each term estimated with each method) and columns:

- `method`: Either OLS/MA/SMA/RMA
- `term`: Either "Intercept" or "Slope"
- `estimate`: Estimated coefficient
- `conf.low`: Lower bound of 95\
- `conf.high`: Upper bound of 95\

See Also

`tidy()`, `lmodel2::lmodel2()`

Other lmodel2 tidiers: `glance.lmodel2()`

Examples

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

  library(lmodel2)

  data(mod2ex2)
  Ex2.res <- lmodel2(Prey ~ Predators, data=mod2ex2, "relative", "relative", 99)
  Ex2.res

  tidy(Ex2.res)
  glance(Ex2.res)

  # this allows coefficient plots with ggplot2
  library(ggplot2)
  ggplot(tidy(Ex2.res), aes(estimate, term, color = method)) +
    geom_point() +
    geom_errorbarh(aes(xmin = conf.low, xmax = conf.high)) +
    geom_errorbarh(aes(xmin = conf.low, xmax = conf.high))
}
```

tidy.manova *Tidy a(n) manova object*

Description

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

Usage

```
## S3 method for class 'manova'
tidy(x, test = "Pillai", ...)
```

Arguments

x	A manova object return from stats::manova().
test	One of "Pillai" (Pillai's trace), "Wilks" (Wilk's lambda), "Hotelling-Lawley" (Hotelling-Lawley trace) or "Roy" (Roy's greatest root) indicating which test statistic should be used. Defaults to "Pillai".
...	Arguments passed on to stats::summary.manova object An object of class "manova" or an aov object with multiple responses. intercept logical. If TRUE, the intercept term is included in the table. tol tolerance to be used in deciding if the residuals are rank-deficient: see qr.

Value

A tibble::tibble with columns:

```
\item{term}{Term in design}
\item{statistic}{Approximate F statistic}
\item{num.df}{Degrees of freedom}
\item{p.value}{P-value}
```

Depending on which test statistic is specified, one of the following columns is also included:

```
\item{pillai}{Pillai's trace}
\item{wilks}{Wilk's lambda}
\item{hl}{Hotelling-Lawley trace}
\item{roy}{Roy's greatest root}
```

See Also

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

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

Examples

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

`tidy.map` *Tidy a(n) map object*

Description

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

Usage

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

Arguments

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

See Also

`tidy()`, `maps::map()`

Examples

```
if (require("maps") && require("ggplot2")) {

    library(maps)
    library(ggplot2)

    ca <- map("county", "ca", plot = FALSE, fill = TRUE)
    tidy(ca)
    qplot(long, lat, data = ca, geom = "polygon", group = group)
```

```
    tx <- map("county", "texas", plot = FALSE, fill = TRUE)
    tidy(tx)
    qplot(long, lat, data = tx, geom = "polygon", group = group,
          colour = I("white"))
}
```

`tidy.Mclust` *Tidy a(n) Mclust object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with one row per component:

`component`	Cluster id as a factor. For a model `k` clusters, these will be `as.factor(1:k)`, or `as.factor(0:k)` if there's a noise term.
`size`	Number of observations assigned to 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 2+ dimensional models, a column with the mean is added for each dimension. NA for noise component

See Also

tidy(), mclust::Mclust()

Other mclust tidiers: augment.Mclust()

Examples

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

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

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

m <- mclust::Mclust(points)

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

---

tidy.mle2 *Tidy a(n) mle2 object*

Description

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

Usage

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

Arguments

x	An mle2 object created by a call to bbmle::mle2().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

See Also

tidy(), bbmle::mle2(), tidy_optim()

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)
}
```

---

tidy.muhaz *Tidy a(n) muhaz object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with two columns:

time	The time at which the hazard rate was estimated.
estimate	The estimated hazard rate.

See Also

tidy(), muhaz::muhaz()

Other muhaz tidiers: glance.muhaz()

Examples

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

---

tidy.multinom *Tidying methods for multinomial logistic regression models*

Description

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

Usage

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

Arguments

x	A multinom object returned from nnet::multinom().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

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, also includes columns for conf.low and conf.high.

See Also

tidy(), nnet::multinom()

Other multinom tidiers: glance.multinom()

Examples

```
if (require(nnet) & require(MASS)){
  library(nnet)
  library(MASS)

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

```
  glance(bwt.mu)

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

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

`tidy.nlrq` *Tidy a(n) nlrq object*

Description

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

Usage

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

Arguments

`x`	A `nlrq` object returned from `quantreg::nlrq()`.
`conf.int`	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
`conf.level`	The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble() with one row for each term in the regression. The tibble has columns:

term	The name of the regression term.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
statistic	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
p.value	The two-sided p-value associated with the observed statistic.
conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

See Also

tidy(), quantreg::nlrq()

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

---

tidy.nls *Tidy a(n) nls object*

Description

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

Usage

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

Arguments

x	An nls object returned from stats::nls().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

quick	Logical indiciating if the only the term and estimate columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with one row for each term in the regression. The tibble has columns:

term	The name of the regression term.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
statistic	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
p.value	The two-sided p-value associated with the observed statistic.
conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

See Also

tidy, stats::nls(), stats::summary.nls()

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

Examples

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

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

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

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

`tidy.numeric` *Tidy atomic vectors*

Description

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

Usage

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

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

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

Arguments

`x`	An object of class "numeric", "integer", "character", or "logical". Most likely a named vector
`...`	Extra arguments (not used)

Details

Turn atomic vectors into data frames, where the names of the vector (if they exist) are a column and the values of the vector are a column.

Examples

```
## Not run:
x <- 1:5
names(x) <- letters[1:5]
tidy(x)

## End(Not run)
```

`tidy.orcutt` *Tidy a(n) orcutt object*

Description

This method wraps `tidy.lm()`.

Usage

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

Arguments

`x`	An `orcutt` object returned from `orcutt::cochrane.orcutt()`.
`...`	Arguments passed on to `tidy.lm`
	`conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`.
	`conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
	`quick` Logical indiciating if the only the `term` and `estimate` columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to `FALSE`.
	`exponentiate` Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.

Value

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

See Also

`tidy()`, `tidy.lm()`

`orcutt::cochrane.orcutt()`

Other orcutt tidiers: `glance.orcutt()`

Examples

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

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

```
  co

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

`tidy.pairwise.htest` *Tidy a(n) pairwise.htest object*

Description

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

Usage

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

Arguments

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

Details

Note that in one-sided tests, the alternative hypothesis of each test can be stated as "group1 is greater/less than group2".

Note also that the columns of group1 and group2 will always be a factor, even if the original input is (e.g.) numeric.

Value

A tibble::tibble with one row per group/group comparison and columns:

`group1`	First group being compared
`group2`	Second group being compared
`p.value`	(Adjusted) p-value of comparison

See Also

stats::pairwise.t.test(), stats::pairwise.wilcox.test(), tidy()

Other htest tidiers: augment.htest(), tidy.htest(), tidy.power.htest()

Examples

```
attach(airquality)
Month <- factor(Month, labels = month.abb[5:9])
ptt <- pairwise.t.test(Ozone, Month)
tidy(ptt)

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.plm *Tidy a(n) plm object*

Description

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

Usage

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

Arguments

x	A plm objected returned by plm::plm().
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.

exponentiate	Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to FALSE.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with one row for each term in the regression. The tibble has columns:

term	The name of the regression term.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
statistic	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
p.value	The two-sided p-value associated with the observed statistic.
conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

See Also

tidy(), plm::plm(), tidy.lm()

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

Examples

```
library(plm)

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

summary(zz)

tidy(zz)
tidy(zz, conf.int = TRUE)
tidy(zz, conf.int = TRUE, conf.level = .9)

augment(zz)
glance(zz)
```

`tidy.poLCA` *Tidy a(n) poLCA object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with one row 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

See Also

`tidy()`, `poLCA::poLCA()`

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

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)
  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 <- tibble(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)
  td

  # show

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

  au <- augment(nes2a)
  au
  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)
  au2
  dim(au2)
}
```

---

tidy.polr *Tidying methods for ordinal logistic regression models*

Description

These methods tidy the coefficients of ordinal logistic regression models generated by `ordinal::clm()` or `ordinal::clmm()` of the `ordinal` package, `MASS::polr()` of the `MASS` packge, or `survey::svyolr()` of the `survey` package.

Usage

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

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

## S3 method for class 'polr'
augment(
  x,
  data = stats::model.frame(x),
  newdata,
  type.predict = c("probs", "class"),
  ...
)

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

```
  quick = FALSE,
  conf.type = c("profile", "Wald"),
  ...
)

## S3 method for class 'clmm'
tidy(
  x,
  conf.int = FALSE,
  conf.level = 0.95,
  exponentiate = FALSE,
  quick = FALSE,
  conf.type = c("profile", "Wald"),
  ...
)

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

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

## S3 method for class 'clm'
augment(
  x,
  data = stats::model.frame(x),
  newdata,
  type.predict = c("prob", "class"),
  ...
)

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

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

Arguments

x	a model of class clm, clmm, polr or svyolr
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 ordinal logistic regression)
quick	whether to compute a smaller and faster version, containing only the term, estimate and coefficient_type columns
...	extra arguments
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 CLM; passed on to ordinal::predict.clm() or predict.polr
conf.type	the type of confidence interval (see ordinal::confint.clm())

Value

tidy.clm, tidy.clmm, tidy.polr and tidy.svyolr return one row for each coefficient at each level of the response variable, with six columns:

term	term in the model
estimate	estimated coefficient
std.error	standard error
statistic	z-statistic
p.value	two-sided p-value
coefficient_type	type of coefficient, see ordinal::clm()

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

glance.clm, glance.clmm, glance.polr and glance.svyolr return a one-row data.frame with the columns:

edf	the effective degrees of freedom
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.clm and augment.polr returns one row for each observation, with additional columns added to the original data:

.fitted	fitted values of model
.se.fit	standard errors of fitted values

augment is not supportted for ordinal::clmm() and survey::svyolr() models.

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

Examples

```
if (require(ordinal)){
  clm_mod <- clm(rating ~ temp * contact, data = wine)
  tidy(clm_mod)
  tidy(clm_mod, conf.int = TRUE)
  tidy(clm_mod, conf.int = TRUE, conf.type = "Wald", exponentiate = TRUE)
  glance(clm_mod)
  augment(clm_mod)

  clm_mod2 <- clm(rating ~ temp, nominal = ~ contact, data = wine)
  tidy(clm_mod2)

  clmm_mod <- clmm(rating ~ temp + contact + (1 | judge), data = wine)
  tidy(clmm_mod)
  glance(clmm_mod)
}
if (require(MASS)) {
  polr_mod <- polr(Sat ~ Infl + Type + Cont, weights = Freq, data = housing)
  tidy(polr_mod, exponentiate = TRUE, conf.int = TRUE)
  glance(polr_mod)
  augment(polr_mod, type.predict = "class")
}
```

---

tidy.power.htest *Tidy a(n) power.htest object*

Description

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

Usage

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

Arguments

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

Value

A data frame with one row per parameter passed in, with columns `n`, `delta`, `sd`, `sig.level`, and `power`.

See Also

`stats::power.t.test()`

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

Examples

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

library(ggplot2)

ggplot(tidy(ptt), aes(n, power)) +
  geom_line()
```

`tidy.prcomp` *Tidy a(n) prcomp object*

Description

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

Usage

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

Arguments

`x`	A `prcomp` object returned by `stats::prcomp()`.
`matrix`	Character specifying which component of the PCA should be tidied. • `"u"`, `"samples"`, or `"x"`: returns information about the map from the original space into principle components space. • `"v"`, `"rotation"`, or `"variables"`: returns information about the map from principle components space back into the original space. • `"d"` or `"pcs"`: returns information about the eigenvalues will return information about

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

Details

See https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca for information on how to interpret the various tidied matrices. Note that SVD is only equivalent to PCA on centered data.

Value

A tibble::tibble with columns depending on the component of PCA being tidied.

If `matrix` is `"u"`, `"samples"`, or `"x"` each row in the tidied output corresponds to the original data in PCA space. The columns are:

`row`	ID of the original observation (i.e. rowname from original data).
`PC`	Integer indicating a principle component.
`value`	The score of the observation for that particular principle component. That is, the location of the observation in PCA space.

If `matrix` is `"v"`, `"rotation"`, or `"variables"`, each row in the tidied ouput corresponds to information about the principle components in the original space. The columns are:

`row`	The variable labels (colnames) of the data set on which PCA was performed
`PC`	An integer vector indicating the principal component
`value`	The value of the eigenvector (axis score) on the indicated principal component

If `matrix` is `"d"` or `"pcs"`, the columns are:

`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

See Also

`stats::prcomp()`, svd_tidiers

Other svd tidiers: `augment.prcomp()`, `tidy_irlba()`, `tidy_svd()`

Examples

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

# information about rotation
tidy(pc)

# information about samples (states)
tidy(pc, "samples")

# information about PCs
tidy(pc, "pcs")

# state map
library(dplyr)
library(ggplot2)

pc %>%
  tidy(matrix = "samples") %>%
  mutate(region = tolower(row)) %>%
  inner_join(map_data("state"), by = "region") %>%
  ggplot(aes(long, lat, group = group, fill = value)) +
  geom_polygon() +
  facet_wrap(~ PC) +
  theme_void() +
  ggtitle("Principal components of arrest data")

au <- augment(pc, data = USArrests)
au

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

---

tidy.pyears *Tidy a(n) pyears object*

Description

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

Usage

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

Arguments

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

Value

A tibble::tibble with one row for each time point and 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.

See Also

tidy(), survival::pyears()

Other pyears tidiers: glance.pyears()

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

Examples

```
library(survival)

temp.yr  <- tcut(mgus$dxyr, 55:92, labels=as.character(55:91))
temp.age <- tcut(mgus$age, 34:101, labels=as.character(34:100))
ptime <- ifelse(is.na(mgus$pctime), mgus$futime, mgus$pctime)
pstat <- ifelse(is.na(mgus$pctime), 0, 1)
pfit <- pyears(Surv(ptime/365.25, pstat) ~ temp.yr + temp.age + sex,  mgus,
               data.frame=TRUE)
tidy(pfit)
glance(pfit)

# if data.frame argument is not given, different information is present in
# output
pfit2 <- pyears(Surv(ptime/365.25, pstat) ~ temp.yr + temp.age + sex,  mgus)
tidy(pfit2)
glance(pfit2)
```

`tidy.rcorr` *Tidy a(n) rcorr object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'rcorr'
tidy(x, diagonal = FALSE, ...)
```

Arguments

`x`	An `rcorr` object returned from `Hmisc::rcorr()`.
`diagonal`	Logical indicating whether or not to include diagonal elements of the correlation matrix, or the correlation of a column with itself. For the elements, `estimate` is always 1 and `p.value` is always `NA`. Defaults to `FALSE`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

Suppose the original data has columns A and B. In the correlation matrix from `rcorr` there may be entries for both the `cor(A,B)` and `cor(B,A)`. Only one of these pairs will ever be present in the tidy output.

Value

A tibble::tibble with one row for each unique pair of columns in the correlatin matrix and columns:

`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

See Also

tidy(), Hmisc::rcorr()

Examples

```
if (requireNamespace("Hmisc", quietly = TRUE)) {

    library(Hmisc)

    mat <- replicate(52, rnorm(100))
    # add some NAs
    mat[sample(length(mat), 2000)] <- NA
    # also column names
    colnames(mat) <- c(LETTERS, letters)

    rc <- rcorr(mat)

    td <- tidy(rc)
    td

    library(ggplot2)
    ggplot(td, aes(p.value)) +
        geom_histogram(binwidth = .1)

    ggplot(td, aes(estimate, p.value)) +
        geom_point() +
        scale_y_log10()
}
```

---

tidy.ridgelm *Tidy a(n) ridgelm object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'ridgelm'
tidy(x, ...)
```

Arguments

x	A ridgelm object returned from MASS::lm.ridge().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble with one row for each combination of lambda and a 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

See Also

tidy(), MASS::lm.ridge()

Other ridgelm tidiers: glance.ridgelm()

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)
```

`tidy.rlm` *Tidy a(n) rlm object*

Description

This method wraps `tidy.lm()`.

Usage

```
## S3 method for class 'rlm'
tidy(x, ...)
```

Arguments

`x`	An `rlm` object returned by `MASS::rlm()`.
`...`	Arguments passed on to `tidy.lm` `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`. `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval. `quick` Logical indiciating if the only the `term` and `estimate` columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to `FALSE`. `exponentiate` Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.

Value

A `tibble::tibble()` with one row for each term in the regression. The tibble has columns:

`term`	The name of the regression term.
`estimate`	The estimated value of the regression term.
`std.error`	The standard error of the regression term.
`statistic`	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
`p.value`	The two-sided p-value associated with the observed statistic.
`conf.low`	The low end of a confidence interval for the regression term. Included only if `conf.int = TRUE`.
`conf.high`	The high end of a confidence interval for the regression term. Included only if `conf.int = TRUE`.

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)

See Also

tidy(), tidy.lm()

MASS::rlm()

Other rlm tidiers: augment.rlm(), glance.rlm()

---

tidy.roc *Tidy a(n) roc object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'roc'
tidy(x, ...)
```

Arguments

x	An roc object returned from a call to AUC::roc().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble() with three columns:

cutoff	The cutoff used for classification. Observations with predicted probabilities above this value were assigned class 1, and observations with predicted probabilities below this value were assigned class 0.
tpr	The true positive rate at the given cutoff.
fpr	The false positive rate at the given cutoff.

See Also

tidy(), AUC::roc()

Examples

```
if (require("AUC", quietly = TRUE)) {
  data(churn)
  r <- roc(churn$predictions,churn$labels)

  td <- tidy(r)
  td

  library(ggplot2)

  ggplot(td, aes(fpr, tpr)) +
    geom_line()

  # compare the ROC curves for two prediction algorithms

  library(dplyr)
  library(tidyr)

  rocs <- churn %>%
    gather(algorithm, value, -labels) %>%
    nest(-algorithm) %>%
    mutate(tidy_roc = purrr::map(data, ~tidy(roc(.x$value, .x$labels)))) %>%
    unnest(tidy_roc)

  ggplot(rocs, aes(fpr, tpr, color = algorithm)) +
    geom_line()
}
```

`tidy.rq` *Tidy a(n) rq object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'rq'
tidy(x, se.type = "rank", conf.int = TRUE, conf.level = 0.95, ...)
```

Arguments

x	An rq object returned from quantreg::rq().

se.type	Character specifying the method to use to calculate standard errors. Passed to quantreg::summary.rq() se argument. Defaults to "rank".
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments passed to quantreg::summary.rq().

Details

If se.type = "rank" confidence intervals are calculated by summary.rq. When only a single predictor is included in the model, no confidence intervals are calculated and the confidence limits are set to NA.

Value

A tibble::tibble() with one row for each term in the regression. The tibble has columns:

term	The name of the regression term.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
statistic	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
p.value	The two-sided p-value associated with the observed statistic.
conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

See Also

tidy(), quantreg::rq()

Other quantreg tidiers: augment.nlrq(), augment.rqs(), augment.rq(), glance.nlrq(), glance.rq(), tidy.nlrq(), tidy.rqs()

---

tidy.rqs *Tidy a(n) rqs object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'rqs'
tidy(x, se.type = "rank", conf.int = TRUE, conf.level = 0.95, ...)
```

Arguments

x	An rqs object returned from quantreg::rq().
se.type	Character specifying the method to use to calculate standard errors. Passed to quantreg::summary.rq() se argument. Defaults to "rank".
conf.int	Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval.
...	Additional arguments passed to quantreg::summary.rqs()

Details

If se.type = "rank" confidence intervals are calculated by summary.rq. When only a single predictor is included in the model, no confidence intervals are calculated and the confidence limits are set to NA.

Value

A tibble::tibble() with one row for each term in the regression. The tibble has columns:

term	The name of the regression term.
estimate	The estimated value of the regression term.
std.error	The standard error of the regression term.
statistic	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
p.value	The two-sided p-value associated with the observed statistic.
conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

An additional quantile column indicating with quantile the coefficient corresponds to.

See Also

tidy(), quantreg::rq()

Other quantreg tidiers: augment.nlrq(), augment.rqs(), augment.rq(), glance.nlrq(), glance.rq(), tidy.nlrq(), tidy.rq()

`tidy.spec` *Tidy a(n) spec object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'spec'
tidy(x, ...)
```

Arguments

`x`	A spec object created by `stats::spectrum()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with two columns: `freq` and `spec`.

See Also

`tidy()`, `stats::spectrum()`

Other time series tidiers: `tidy.acf()`, `tidy.ts()`, `tidy.zoo()`

Examples

```
spc <- spectrum(lh)
tidy(spc)

library(ggplot2)
ggplot(tidy(spc), aes(freq, spec)) +
  geom_line()
```

`tidy.speedlm` *Tidy a(n) speedlm object*

Description

This method wraps `tidy.lm()`.

Usage

```
## S3 method for class 'speedlm'
tidy(x, ...)
```

Arguments

`x`	A `speedlm` object returned from `speedglm::speedlm()`.
`...`	Arguments passed on to `tidy.lm` `conf.int` Logical indicating whether or not to include a confidence interval in the tidied output. Defaults to `FALSE`. `conf.level` The confidence level to use for the confidence interval if `conf.int = TRUE`. Must be strictly greater than 0 and less than 1. Defaults to 0.95, which corresponds to a 95 percent confidence interval. `quick` Logical indiciating if the only the `term` and `estimate` columns should be returned. Often useful to avoid time consuming covariance and standard error calculations. Defaults to `FALSE`. `exponentiate` Logical indicating whether or not to exponentiate the the coefficient estimates. This is typical for logistic and multinomial regressions, but a bad idea if there is no log or logit link. Defaults to `FALSE`.

Value

A `tibble::tibble()` with one row for each term in the regression. The tibble has columns:

`term`	The name of the regression term.
`estimate`	The estimated value of the regression term.
`std.error`	The standard error of the regression term.
`statistic`	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
`p.value`	The two-sided p-value associated with the observed statistic.
`conf.low`	The low end of a confidence interval for the regression term. Included only if `conf.int = TRUE`.
`conf.high`	The high end of a confidence interval for the regression term. Included only if `conf.int = TRUE`.

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)

See Also

tidy(), tidy.lm()

speedglm::speedlm()

Other speedlm tidiers: augment.speedlm(), glance.speedlm()

Examples

```
mod <- speedglm::speedlm(mpg ~ wt + qsec, data = mtcars)

tidy(mod)
glance(mod)
augment(mod)
```

---

tidy.summary.glht *Tidy a(n) summary.glht object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'summary.glht'
tidy(x, ...)
```

Arguments

x	A summary.glht object created by calling multcomp::summary.glht() on a glht object created with multcomp::glht().
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

See Also

tidy(), multcomp::summary.glht(), multcomp::glht()

Other multcomp tidiers: tidy.cld(), tidy.confint.glht(), tidy.glht()

`tidy.survdiff` *Tidy a(n) survdiff object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'survdiff'
tidy(x, ...)
```

Arguments

`x`	An `survdiff` object returned from `survival::survdiff()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one row for each time point and columns:

`...`	The initial columns correspond to the grouping factors on the right hand side of the model 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

See Also

`tidy()`, `survival::survdiff()`

Other survdiff tidiers: `glance.survdiff()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survexp()`, `tidy.survfit()`, `tidy.survreg()`

Examples

```
library(survival)

s <- survdiff(
  Surv(time, status) ~ pat.karno + strata(inst),
  data = lung
)

tidy(s)
glance(s)
```

`tidy.survexp` *Tidy a(n) survexp object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'survexp'
tidy(x, ...)
```

Arguments

`x`	An `survexp` object returned from `survival::survexp()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one row for each time point and columns:

`time`	time point
`estimate`	estimated survival
`n.risk`	number of individuals at risk

See Also

tidy(), survival::survexp()

Other survexp tidiers: glance.survexp()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survfit(), tidy.survreg()

Examples

```
library(survival)
sexpfit <- survexp(
  futime ~ 1,
  rmap = list(
    sex = "male",
    year = accept.dt,
    age = (accept.dt - birth.dt)
  ),
  method = 'conditional',
  data = jasa
)

tidy(sexpfit)
glance(sexpfit)
```

---

tidy.survfit *Tidy a(n) survfit object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'survfit'
tidy(x, ...)
```

Arguments

x	An survfit object returned from survival::survfit().

`...` Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one row for each time point and 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

See Also

`tidy()`, `survival::survfit()`

Other survival tidiers: `augment.coxph()`, `augment.survreg()`, `glance.aareg()`, `glance.cch()`, `glance.coxph()`, `glance.pyears()`, `glance.survdiff()`, `glance.survexp()`, `glance.survfit()`, `glance.survreg()`, `tidy.aareg()`, `tidy.cch()`, `tidy.coxph()`, `tidy.pyears()`, `tidy.survdiff()`, `tidy.survexp()`, `tidy.survreg()`

Examples

```
library(survival)
cfit <- coxph(Surv(time, status) ~ age + sex, lung)
sfit <- survfit(cfit)

tidy(sfit)
glance(sfit)

library(ggplot2)
ggplot(tidy(sfit), aes(time, estimate)) + geom_line() +
    geom_ribbon(aes(ymin=conf.low, ymax=conf.high), alpha=.25)

# multi-state
fitCI <- survfit(Surv(stop, status * as.numeric(event), type = "mstate") ~ 1,
```

```
              data = mgus1, subset = (start == 0))
td_multi <- tidy(fitCI)
td_multi

ggplot(td_multi, aes(time, estimate, group = state)) +
    geom_line(aes(color = state)) +
    geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = .25)
```

`tidy.survreg` *Tidy a(n) survreg object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'survreg'
tidy(x, conf.level = 0.95, ...)
```

Arguments

`x`	An `survreg` object returned from `survival::survreg()`.
`conf.level`	confidence level for CI
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A `tibble::tibble()` with one row for each term in the regression. The tibble has columns:

`term`	The name of the regression term.
`estimate`	The estimated value of the regression term.
`std.error`	The standard error of the regression term.
`statistic`	The value of a statistic, almost always a T-statistic, to use in a hypothesis that the regression term is non-zero.
`p.value`	The two-sided p-value associated with the observed statistic.

conf.low	The low end of a confidence interval for the regression term. Included only if conf.int = TRUE.
conf.high	The high end of a confidence interval for the regression term. Included only if conf.int = TRUE.

See Also

tidy(), survival::survreg()

Other survreg tidiers: augment.survreg(), glance.survreg()

Other survival tidiers: augment.coxph(), augment.survreg(), glance.aareg(), glance.cch(), glance.coxph(), glance.pyears(), glance.survdiff(), glance.survexp(), glance.survfit(), glance.survreg(), tidy.aareg(), tidy.cch(), tidy.coxph(), tidy.pyears(), tidy.survdiff(), tidy.survexp(), tidy.survfit()

Examples

```
library(survival)

sr <- survreg(
  Surv(futime, fustat) ~ ecog.ps + rx,
  ovarian,
  dist = "exponential"
)

td <- tidy(sr)
augment(sr, ovarian)
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)
```

---

tidy.table *Tidy a(n) table object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'table'
tidy(x, ...)
```

Arguments

x	A table object.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

Directly calls `tibble::as_tibble()` on a table object, which does the same things as `as.data.frame.table()` but also gives the returned object tibble::tibble class.

Value

A tibble::tibble in long-form containing frequency information for the table in a `Freq` column. The result is much like what you get from `tidyr::gather()`.

See Also

`as_tibble.table()`

Examples

```
tab <- with(airquality, table(Temp = cut(Temp, quantile(Temp)), Month))
tidy(tab)
```

---

tidy.ts *Tidy a(n) ts object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'ts'
tidy(x, ...)
```

Arguments

x	A univariate or multivariate ts times series object.
...	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in ..., where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass conf.lvel = 0.9, all computation will proceed using conf.level = 0.95. Additionally, if you pass newdata = my_tibble to an augment() method that does not accept a newdata argument, it will use the default value for the data argument.

Value

A tibble::tibble with one row for each observation and columns:

index	Index (i.e. date or time) for the "ts" object.
series	Name of the series (multivariate "ts" objects only).
value	Value of the observation.

See Also

tidy(), stats::ts()

Other time series tidiers: tidy.acf(), tidy.spec(), tidy.zoo()

Examples

```
set.seed(678)

tidy(ts(1:10, frequency = 4, start = c(1959, 2)))

z <- ts(matrix(rnorm(300), 100, 3), start = c(1961, 1), frequency = 12)
colnames(z) <- c("Aa", "Bb", "Cc")
tidy(z)
```

`tidy.TukeyHSD` *Tidy a(n) TukeyHSD object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'TukeyHSD'
tidy(x, ...)
```

Arguments

`x`	A `TukeyHSD` object return from `stats::TukeyHSD()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one row per comparison and 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

See Also

`tidy()`, `stats::TukeyHSD()`

Other anova tidiers: `tidy.anova()`, `tidy.aovlist()`, `tidy.aov()`, `tidy.manova()`

Examples

```
fm1 <- aov(breaks ~ wool + tension, data = warpbreaks)
thsd <- TukeyHSD(fm1, "tension", ordered = TRUE)
tidy(thsd)

# may include comparisons on multiple terms
fm2 <- aov(mpg ~ as.factor(gear) * as.factor(cyl), data = mtcars)
tidy(TukeyHSD(fm2))
```

`tidy.zoo` *Tidy a(n) zoo object*

Description

Tidy summarizes information about the components of a model. A model component might be a single term in a regression, a single hypothesis, a cluster, or a class. Exactly what tidy considers to be a model component varies cross models but is usually self-evident. If a model has several distinct types of components, you will need to specify which components to return.

Usage

```
## S3 method for class 'zoo'
tidy(x, ...)
```

Arguments

`x`	A zoo object such as those created by `zoo::zoo()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one row for each observation in the zoo time series and columns:

`index`	Index (usually date) for the zoo object
`series`	Name of the series
`value`	Value of the observation

See Also

tidy(), zoo::zoo()

Other time series tidiers: tidy.acf(), tidy.spec(), tidy.ts()

Examples

```
library(zoo)
library(ggplot2)

set.seed(1071)

# data generated as shown in the zoo vignette
Z.index <- as.Date(sample(12450:12500, 10))
Z.data <- matrix(rnorm(30), ncol = 3)
colnames(Z.data) <- c("Aa", "Bb", "Cc")
Z <- zoo(Z.data, Z.index)

tidy(Z)

ggplot(tidy(Z), aes(index, value, color = series)) +
  geom_line()

ggplot(tidy(Z), aes(index, value)) +
  geom_line() +
  facet_wrap(~ series, ncol = 1)

Zrolled <- rollmean(Z, 5)
ggplot(tidy(Zrolled), aes(index, value, color = series)) +
  geom_line()
```

---

tidy_irlba *Tidy a(n) irlba object masquerading as list*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, stats::optim(), svd() and akima::interp() produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves are implemented as functions of the form tidy_<function> or glance_<function> and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.

Usage

```
tidy_irlba(x, ...)
```

Arguments

x	A list returned from irlba::irlba().
...	Arguments passed on to tidy_svd
matrix	Character specifying which component of the PCA should be tidied. • "u", "samples", or "x": returns information about the map from the original space into principle components space. • "v", "rotation", or "variables": returns information about the map from principle components space back into the original space. • "d" or "pcs": returns information about the eigenvalues will return information about

Details

A very thin wrapper around tidy_svd().

Value

A tibble::tibble with columns depending on the component of PCA being tidied.

If matrix is "u", "samples", or "x" each row in the tidied output corresponds to the original data in PCA space. The columns are:

row	ID of the original observation (i.e. rowname from original data).
PC	Integer indicating a principle component.
value	The score of the observation for that particular principle component. That is, the location of the observation in PCA space.

If matrix is "v", "rotation", or "variables", each row in the tidied ouput corresponds to information about the principle components in the original space. The columns are:

row	The variable labels (colnames) of the data set on which PCA was performed
PC	An integer vector indicating the principal component
value	The value of the eigenvector (axis score) on the indicated principal component

If matrix is "d" or "pcs", the columns are:

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

See Also

tidy(), irlba::irlba()

Other list tidiers: glance_optim(), list_tidiers, tidy_optim(), tidy_svd(), tidy_xyz()

Other svd tidiers: augment.prcomp(), tidy.prcomp(), tidy_svd()

`tidy_optim` *Tidy a(n) optim object masquerading as list*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, `stats::optim()`, `svd()` and `akima::interp()` produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves are implemented as functions of the form tidy_<function> or glance_<function> and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.

Usage

```
tidy_optim(x, ...)
```

Arguments

`x`	A list returned from `stats::optim()`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with one row per parameter estimated by `optim` and 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

See Also

`tidy()`, `stats::optim()`

Other list tidiers: `glance_optim()`, `list_tidiers`, `tidy_irlba()`, `tidy_svd()`, `tidy_xyz()`

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)
```

`tidy_svd` *Tidy a(n) svd object masquerading as list*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, `stats::optim()`, `svd()` and `akima::interp()` produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves are implemented as functions of the form tidy_<function> or glance_<function> and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.

Usage

```
tidy_svd(x, matrix = "u", ...)
```

Arguments

`x`	A list with components `u`, `d`, `v` returned by `svd()`.
`matrix`	Character specifying which component of the PCA should be tidied. • `"u"`, `"samples"`, or `"x"`: returns information about the map from the original space into principle components space. • `"v"`, `"rotation"`, or `"variables"`: returns information about the map from principle components space back into the original space. • `"d"` or `"pcs"`: returns information about the eigenvalues will return information about
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Details

See https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to-use-svd-to-perform-pca for information on how to interpret the various tidied matrices. Note that SVD is only equivalent to PCA on centered data.

Value

A tibble::tibble with columns depending on the component of PCA being tidied.

If `matrix` is `"u"`, `"samples"`, or `"x"` each row in the tidied output corresponds to the original data in PCA space. The columns are:

`row`	ID of the original observation (i.e. rowname from original data).
`PC`	Integer indicating a principle component.
`value`	The score of the observation for that particular principle component. That is, the location of the observation in PCA space.

If `matrix` is `"v"`, `"rotation"`, or `"variables"`, each row in the tidied ouput corresponds to information about the principle components in the original space. The columns are:

`row`	The variable labels (colnames) of the data set on which PCA was performed
`PC`	An integer vector indicating the principal component
`value`	The value of the eigenvector (axis score) on the indicated principal component

If `matrix` is `"d"` or `"pcs"`, the columns are:

`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

See Also

`svd()`

Other svd tidiers: `augment.prcomp()`, `tidy.prcomp()`, `tidy_irlba()`

Other list tidiers: `glance_optim()`, `list_tidiers`, `tidy_irlba()`, `tidy_optim()`, `tidy_xyz()`

Examples

```
mat <- scale(as.matrix(iris[, 1:4]))
s <- svd(mat)

tidy_u <- tidy(s, matrix = "u")
tidy_u

tidy_d <- tidy(s, matrix = "d")
tidy_d
```

```
tidy_v <- tidy(s, matrix = "v")
tidy_v

library(ggplot2)
library(dplyr)

ggplot(tidy_d, aes(PC, percent)) +
    geom_point() +
    ylab("% of variance explained")

tidy_u %>%
    mutate(Species = iris$Species[row]) %>%
    ggplot(aes(Species, value)) +
    geom_boxplot() +
    facet_wrap(~ PC, scale = "free_y")
```

`tidy_xyz` *Tidy a(n) xyz object masquerading as list*

Description

Broom tidies a number of lists that are effectively S3 objects without a class attribute. For example, `stats::optim()`, `svd()` and `akima::interp()` produce consistent output, but because they do not have a class attribute, they cannot be handled by S3 dispatch.

These functions look at the elements of a list and determine if there is an appropriate tidying method to apply to the list. Those tidiers are themselves are implemented as functions of the form tidy_<function> or glance_<function> and are not exported (but they are documented!).

If no appropriate tidying method is found, throws an error.

xyz lists (lists where `x` and `y` are vector of coordinates and `z` is a matrix of values) are typically used by functions such as `graphics::persp()` or `graphics::image()` and returned by interpolation functions such as `akima::interp()`.

Usage

```
tidy_xyz(x, ...)
```

Arguments

`x`	A list with component `x`, `y` and `z`, where `x` and `y` are vectors and `z` is a matrix. The length of `x` must equal the number of rows in `z` and the length of `y` must equal the number of columns in `z`.
`...`	Additional arguments. Not used. Needed to match generic signature only. **Cautionary note:** Misspelled arguments will be absorbed in `...`, where they will be ignored. If the misspelled argument has a default value, the default value will be

used. For example, if you pass `conf.lvel = 0.9`, all computation will proceed using `conf.level = 0.95`. Additionally, if you pass `newdata = my_tibble` to an `augment()` method that does not accept a `newdata` argument, it will use the default value for the `data` argument.

Value

A tibble::tibble with vector columns `x`, `y` and `z`.

See Also

`tidy()`, `graphics::persp()`, `graphics::image()`, `akima::interp()`

Other list tidiers: `glance_optim()`, `list_tidiers`, `tidy_irlba()`, `tidy_optim()`, `tidy_svd()`

Examples

```
A <- list(x = 1:5, y = 1:3, z = matrix(runif(5 * 3), nrow = 5))
image(A)
tidy(A)
```

Index