Package 'broom.helpers'

November 24, 2020

Title Helpers for Model Coefficients Tibbles

Version 1.1.0

Description Provides suite of functions to work with regression model 'broom::tidy()' tibbles. The suite includes functions to group regression model terms by variable, insert reference and header rows for categorical variables, add variable labels, and more.

License GPL-3

URL https://larmarange.github.io/broom.helpers/

BugReports https://github.com/larmarange/broom.helpers/issues

Imports broom, dplyr, emmeans, labelled, lifecycle, purrr, rlang, stats, stringr, tibble, tidyr, usethis

Suggests broom.mixed, covr, datasets, forcats, gam, gee, geepack, ggplot2, gt, gtsummary, knitr, lavaan, lme4, MASS, mice, nnet, ordinal, rmarkdown, survey, survival, testthat, spelling

VignetteBuilder knitr

RdMacros lifecycle

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Language en-US

NeedsCompilation no

Author Joseph Larmarange [aut, cre] (<https://orcid.org/0000-0001-7097-700X>), Daniel D. Sjoberg [aut] (<https://orcid.org/0000-0003-0862-2018>)

Maintainer Joseph Larmarange <joseph@larmarange.net>

Repository CRAN

Date/Publication 2020-11-24 08:00:02 UTC

R topics documented:

`.clean_backticks` *Remove backticks around variable names*

Description

Remove backticks around variable names

Usage

```
.clean_backticks(x, variable_names = x)
```

Arguments

x	a character vector to be cleaned
variable_names	list of variable names, could be obtained with model_list_variables(only_variable = TRUE) to properly take into account interaction only terms/variables

See Also

Other other_helpers: .escape_regex()

.escape_regex *Escapes any characters that would have special meaning in a regular expression*

Description

This functions has been adapted from Hmisc::escapeRegex()

Usage

```
.escape_regex(string)
```

Arguments

string	a character vector

See Also

Other other_helpers: .clean_backticks()

.formula_list_to_named_list *Convert formula selector to a named list*

Description

Functions takes a list of formulas, e.g. list(starts_with("age") ~ "continuous"), and returns a named list, e.g. list(age = "continuous").

Usage

```
.formula_list_to_named_list(
  x,
  data = NULL,
  var_info = NULL,
  arg_name = NULL,
  select_single = FALSE
)
```

Arguments

x	list of selecting formulas
data	A data frame to select columns from. Default is NULL
var_info	A data frame of variable names and attributes. May also pass a character vector of variable names. Default is NULL
arg_name	Optional string indicating the source argument name. This helps in the error messaging. Default is NULL.
select_single	Logical indicating whether the result must be a single variable. Default is FALSE

.generic_selector *Generate a custom selector function*

Description

Generate a custom selector function

Usage

```
.generic_selector(variable_column, select_column, select_expr, fun_name)
```

Arguments

variable_column	string indicating column variable names are stored
select_column	character vector of columns used in the select_expr= argument
select_expr	unquoted predicate command to subset a data frame to select variables
fun_name	quoted name of function where .generic_selector() is being used. This helps with error messaging.

Value

custom selector functions

.select_to_varnames *Variable selector*

Description

Function takes `select()`-like inputs and converts the selector to a character vector of variable names. Functions accepts tidyselect syntax, and additional selector functions defined within the package

Usage

```
.select_to_varnames(
  select,
  data = NULL,
  var_info = NULL,
  arg_name = NULL,
  select_single = FALSE
)
```

Arguments

`select`	A single object selecting variables, e.g. `c(age,stage)`, `starts_with("age")`
`data`	A data frame to select columns from. Default is NULL
`var_info`	A data frame of variable names and attributes. May also pass a character vector of variable names. Default is NULL
`arg_name`	Optional string indicating the source argument name. This helps in the error messaging. Default is NULL.
`select_single`	Logical indicating whether the result must be a single variable. Default is `FALSE`

Value

A character vector of variable names

model_get_coefficients_type *Get coefficient type*

Description

Indicate the type of coefficient among "generic", "logistic", "poisson" and "prop_hazard".

Usage

```
model_get_coefficients_type(model)

## Default S3 method:
model_get_coefficients_type(model)

## S3 method for class 'glm'
model_get_coefficients_type(model)

## S3 method for class 'negbin'
model_get_coefficients_type(model)

## S3 method for class 'geeglm'
model_get_coefficients_type(model)

## S3 method for class 'glmerMod'
model_get_coefficients_type(model)

## S3 method for class 'clogit'
model_get_coefficients_type(model)

## S3 method for class 'polr'
model_get_coefficients_type(model)

## S3 method for class 'multinom'
model_get_coefficients_type(model)

## S3 method for class 'svyolr'
model_get_coefficients_type(model)

## S3 method for class 'clm'
model_get_coefficients_type(model)

## S3 method for class 'clmm'
model_get_coefficients_type(model)

## S3 method for class 'coxph'
model_get_coefficients_type(model)
```

Arguments

model	a model object

See Also

Other model_helpers: model_get_contrasts(), model_get_model_frame(), model_get_model_matrix(), model_get_model(), model_get_nlevels(), model_get_xlevels(), model_identify_variables(), model_list_contrasts(), model_list_terms_levels(), model_list_variables()

Examples

```
lm(hp ~ mpg + factor(cyl), mtcars) %>%
  model_get_coefficients_type()

Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes"))) %>%
  glm(Survived ~ Class + Age * Sex, data = ., weights = .$n, family = binomial) %>%
  model_get_coefficients_type()
```

---

model_get_contrasts *Get contrasts used in the model*

Description

Get contrasts used in the model

Usage

```
model_get_contrasts(model)

## Default S3 method:
model_get_contrasts(model)
```

Arguments

model	a model object

See Also

Other model_helpers: model_get_coefficients_type(), model_get_model_frame(), model_get_model_matrix(), model_get_model(), model_get_nlevels(), model_get_xlevels(), model_identify_variables(), model_list_contrasts(), model_list_terms_levels(), model_list_variables()

Examples

```
glm(
  am ~ mpg + factor(cyl),
  data = mtcars,
  family = binomial,
  contrasts = list(`factor(cyl)` = contr.sum)
) %>%
  model_get_contrasts()
```

model_get_model *Get the model from model objects*

Description

Most model objects are proper R model objects. There are, however, some model objects that store the proper object internally (e.g. mice models). This function extracts that model object in those cases.

Usage

```
model_get_model(model)

## Default S3 method:
model_get_model(model)

## S3 method for class 'mira'
model_get_model(model)
```

Arguments

model	a model object

See Also

Other model_helpers: model_get_coefficients_type(), model_get_contrasts(), model_get_model_frame(), model_get_model_matrix(), model_get_nlevels(), model_get_xlevels(), model_identify_variables(), model_list_contrasts(), model_list_terms_levels(), model_list_variables()

Examples

```
lm(hp ~ mpg + factor(cyl), mtcars) %>%
  model_get_model()
```

model_get_model_frame *Get the model frame of a model*

Description

The structure of the object returned by stats::model.frame() could slightly differ for certain types of models. model_get_model_frame() will always return an object with the same data structure or NULL if it is not possible to compute model frame from model.

Usage

```
model_get_model_frame(model)

## Default S3 method:
model_get_model_frame(model)

## S3 method for class 'coxph'
model_get_model_frame(model)

## S3 method for class 'survreg'
model_get_model_frame(model)
```

Arguments

model	a model object

See Also

stats::model.frame()

Other model_helpers: model_get_coefficients_type(), model_get_contrasts(), model_get_model_matrix(), model_get_model(), model_get_nlevels(), model_get_xlevels(), model_identify_variables(), model_list_contrasts(), model_list_terms_levels(), model_list_variables()

Examples

```
lm(hp ~ mpg + factor(cyl), mtcars) %>%
  model_get_model_frame() %>%
  head()
```

---

model_get_model_matrix

Get the model matrix of a model

Description

The structure of the object returned by stats::model.matrix() could slightly differ for certain types of models. model_get_model_matrix() will always return an object with the same structure as stats::model.matrix.default().

Usage

```
model_get_model_matrix(model)

## Default S3 method:
model_get_model_matrix(model)

## S3 method for class 'multinom'
```

```
model_get_model_matrix(model)

## S3 method for class 'clm'
model_get_model_matrix(model)
```

Arguments

model	a model object

See Also

stats::model.matrix()

Other model_helpers: model_get_coefficients_type(), model_get_contrasts(), model_get_model_frame(), model_get_model(), model_get_nlevels(), model_get_xlevels(), model_identify_variables(), model_list_contrasts(), model_list_terms_levels(), model_list_variables()

Examples

```
lm(hp ~ mpg + factor(cyl), mtcars) %>%
  model_get_model_matrix() %>%
  head()
```

---

model_get_nlevels *Get the number of levels for each factor used in* xlevels

---

Description

Get the number of levels for each factor used in xlevels

Usage

```
model_get_nlevels(model)

## Default S3 method:
model_get_nlevels(model)
```

Arguments

model	a model object

Value

a tibble with two columns: "variable" and "var_nlevels"

See Also

Other model_helpers: model_get_coefficients_type(), model_get_contrasts(), model_get_model_frame(), model_get_model_matrix(), model_get_model(), model_get_xlevels(), model_identify_variables(), model_list_contrasts(), model_list_terms_levels(), model_list_variables()

Examples

```
lm(hp ~ mpg + factor(cyl), mtcars) %>%
  model_get_nlevels()
```

`model_get_xlevels` *Get xlevels used in the model*

Description

Get xlevels used in the model

Usage

```
model_get_xlevels(model)

## Default S3 method:
model_get_xlevels(model)

## S3 method for class 'lmerMod'
model_get_xlevels(model)

## S3 method for class 'glmerMod'
model_get_xlevels(model)
```

Arguments

`model`	a model object

See Also

Other model_helpers: `model_get_coefficients_type()`, `model_get_contrasts()`, `model_get_model_frame()`, `model_get_model_matrix()`, `model_get_model()`, `model_get_nlevels()`, `model_identify_variables()`, `model_list_contrasts()`, `model_list_terms_levels()`, `model_list_variables()`

Examples

```
lm(hp ~ mpg + factor(cyl), mtcars) %>%
  model_get_xlevels()
```

`model_identify_variables` *Identify for each coefficient of a model the corresponding variable*

Description

It will also identify interaction terms and intercept(s).

Usage

```
model_identify_variables(model)

## Default S3 method:
model_identify_variables(model)

## S3 method for class 'lavaan'
model_identify_variables(model)

## S3 method for class 'aov'
model_identify_variables(model)

## S3 method for class 'clm'
model_identify_variables(model)

## S3 method for class 'clmm'
model_identify_variables(model)
```

Arguments

`model`	a model object

Value

A tibble with four columns:

- `term`: coefficients of the model
- `variable`: the corresponding variable
- `var_class`: class of the variable (cf. `stats::.MFclass()`)
- `var_type`: `"continuous"`, `"dichotomous"` (categorical variable with 2 levels), `"categorical"` (categorical variable with 3 or more levels), `"intercept"` or `"interaction"`
- `var_nlevels`: number of original levels for categorical variables

See Also

`tidy_identify_variables()`

Other model_helpers: `model_get_coefficients_type()`, `model_get_contrasts()`, `model_get_model_frame()`, `model_get_model_matrix()`, `model_get_model()`, `model_get_nlevels()`, `model_get_xlevels()`, `model_list_contrasts()`, `model_list_terms_levels()`, `model_list_variables()`

Examples

```
Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes"))) %>%
  glm(
    Survived ~ Class + Age * Sex,
    data = ., weights = .$n,
    family = binomial
  ) %>%
  model_identify_variables()

iris %>%
  lm(
    Sepal.Length ~ poly(Sepal.Width, 2) + Species,
    data = .,
    contrasts = list(Species = contr.sum)
  ) %>%
  model_identify_variables()
```

---

model_list_contrasts *List contrasts used by a model*

Description

List contrasts used by a model

Usage

```
model_list_contrasts(model)

## Default S3 method:
model_list_contrasts(model)
```

Arguments

model	a model object

Value

A tibble with three columns:

- variable: variable name
- contrasts: contrasts used
- contrasts_type: type of contrasts ("treatment", "sum", "poly", "helmert" or "other")
- reference: for variables with treatment, SAS or sum contrasts, position of the reference level

See Also

Other model_helpers: model_get_coefficients_type(), model_get_contrasts(), model_get_model_frame(), model_get_model_matrix(), model_get_model(), model_get_nlevels(), model_get_xlevels(), model_identify_variables(), model_list_terms_levels(), model_list_variables()

Examples

```
glm(
  am ~ mpg + factor(cyl),
  data = mtcars,
  family = binomial,
  contrasts = list(`factor(cyl)` = contr.sum)
) %>%
  model_list_contrasts()
```

---

model_list_terms_levels

List levels of categorical terms

Description

Only for categorical variables with treatment, SAS or sum contrasts.

Usage

```
model_list_terms_levels(
  model,
  label_pattern = "{level}",
  variable_labels = NULL
)

## Default S3 method:
model_list_terms_levels(
  model,
  label_pattern = "{level}",
  variable_labels = NULL
)
```

Arguments

model	a model object
label_pattern	a glue pattern for term labels (see examples)
variable_labels	an optional named list or named vector of custom variable labels passed to model_list_variables()

Value

A tibble with ten columns:

- `variable`: variable
- `contrasts_type`: type of contrasts ("sum" or "treatment")
- `term`: term name
- `level`: term level
- `reference`: logical indicating which term is the reference level
- `reference_level`: level of the reference term
- `var_label`: variable label obtained with `model_list_variables()`
- `var_nlevels`: number of levels in this variable
- `dichotomous`: logical indicating if the variable is dichotomous
- `label`: term label (by default equal to term level) The first nine columns can be used in `label_pattern`.

See Also

Other model_helpers: `model_get_coefficients_type()`, `model_get_contrasts()`, `model_get_model_frame()`, `model_get_model_matrix()`, `model_get_model()`, `model_get_nlevels()`, `model_get_xlevels()`, `model_identify_variables()`, `model_list_contrasts()`, `model_list_variables()`

Examples

```
glm(
  am ~ mpg + factor(cyl),
  data = mtcars,
  family = binomial,
  contrasts = list(`factor(cyl)` = contr.sum)
) %>%
  model_list_terms_levels()

df <- Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes")))

mod <- df %>%
  glm(
    Survived ~ Class + Age + Sex,
    data = ., weights = .$n, family = binomial,
    contrasts = list(Age = contr.sum, Class = "contr.helmert")
  )
mod %>% model_list_terms_levels()
mod %>% model_list_terms_levels("{level} vs {reference_level}")
mod %>% model_list_terms_levels("{variable} [{level} - {reference_level}]")
mod %>% model_list_terms_levels(
  "{ifelse(reference, level, paste(level, '-', reference_level))}"
)
```

`model_list_variables` *List all the variables used in a model*

Description

Including variables used only in an interaction.

Usage

```
model_list_variables(model, labels = NULL, only_variable = FALSE)

## Default S3 method:
model_list_variables(model, labels = NULL, only_variable = FALSE)

## S3 method for class 'lavaan'
model_list_variables(model, labels = NULL, only_variable = FALSE)
```

Arguments

`model`	a model object
`labels`	an optional named list or named vector of custom variable labels
`only_variable`	if `TRUE`, will return only "variable" column

Value

A tibble with three columns:

- `variable`: the corresponding variable
- `var_class`: class of the variable (cf. `stats::.MFclass()`)
- `label_attr`: variable label defined in the original data frame with the label attribute (cf. `labelled::var_label()`)
- `var_label`: a variable label (by priority, `labels` if defined, `label_attr` if available, otherwise `variable`)

See Also

Other model_helpers: `model_get_coefficients_type()`, `model_get_contrasts()`, `model_get_model_frame()`, `model_get_model_matrix()`, `model_get_model()`, `model_get_nlevels()`, `model_get_xlevels()`, `model_identify_variables()`, `model_list_contrasts()`, `model_list_terms_levels()`

Examples

```
Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes"))) %>%
  glm(
    Survived ~ Class + Age : Sex,
```

```
    data = ., weights = .$n,
    family = binomial
  ) %>%
  model_list_variables()

iris %>%
  lm(
    Sepal.Length ~ poly(Sepal.Width, 2) + Species,
    data = .,
    contrasts = list(Species = contr.sum)
  ) %>%
  model_list_variables()

if (requireNamespace("gtsummary")) {
  glm(
    response ~ poly(age, 3) + stage + grade * trt,
    na.omit(gtsummary::trial),
    family = binomial,
  ) %>%
    model_list_variables()
}
```

`select_helpers` *Select helper functions*

Description

Set of functions to supplement the tidyselect set of functions for selecting columns of data frames (and other items as well).

- `all_continuous()` selects continuous variables
- `all_categorical()` selects categorical (including `"dichotomous"`) variables
- `all_dichotomous()` selects only type `"dichotomous"`
- `all_interaction()` selects interaction terms from a regression model
- `all_intercepts()` selects intercept terms from a regression model
- `all_contrasts()` selects variables in regression model based on their type of contrast

Usage

```
all_continuous()

all_dichotomous()

all_categorical(dichotomous = TRUE)

all_interaction()

all_intercepts()

all_contrasts(contrasts_type = NULL)
```

Arguments

dichotomous	Logical indicating whether to include dichotomous variables. Default is TRUE
contrasts_type	type of contrast to select. When NULL, all variables with a contrast will be selected. Default is NULL. Select among contrast types c("treatment","sum","poly","helmert","other

Value

A character vector of column names selected

Examples

```
glm(response ~ age * trt + grade, gtsummary::trial, family = binomial) %>%
  tidy_plus_plus(exponentiate = TRUE, include = all_categorical())

glm(response ~ age + trt + grade + stage,
    gtsummary::trial,
    family = binomial,
    contrasts = list(trt = contr.SAS, grade = contr.sum, stage = contr.poly)) %>%
tidy_plus_plus(exponentiate = TRUE,
               include = all_contrasts(c("treatment", "sum")))
```

---

tidy_add_coefficients_type

Add coefficients type and label as attributes

Description

Add the type of coefficients ("generic", "logistic", "poisson" or "prop_hazard") and the corresponding coefficient labels, as attributes to x (respectively named coefficients_type and coefficients_label).

Usage

```
tidy_add_coefficients_type(
  x,
  exponentiate = attr(x, "exponentiate"),
  model = tidy_get_model(x)
)
```

Arguments

x	a tidy tibble
exponentiate	logical indicating whether or not to exponentiate the coefficient estimates. It should be consistent with the original call to broom::tidy()
model	the corresponding model, if not attached to x

See Also

Other tidy_helpers: tidy_add_contrasts(), tidy_add_estimate_to_reference_rows(), tidy_add_header_rows(), tidy_add_reference_rows(), tidy_add_term_labels(), tidy_add_variable_labels(), tidy_attach_model(), tidy_identify_variables(), tidy_plus_plus(), tidy_remove_intercept(), tidy_select_variables()

Examples

```
ex1 <- lm(hp ~ mpg + factor(cyl), mtcars) %>%
  tidy_and_attach() %>%
  tidy_add_coefficients_type()
attr(ex1, "coefficients_type")
attr(ex1, "coefficients_label")

ex2 <- Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes"))) %>%
  glm(Survived ~ Class + Age * Sex, data = ., weights = .$n, family = binomial) %>%
  tidy_and_attach(exponentiate = TRUE) %>%
  tidy_add_coefficients_type()
attr(ex2, "coefficients_type")
attr(ex2, "coefficients_label")
```

---

tidy_add_contrasts *Add contrasts type for categorical variables*

Description

Add a `contrasts` column corresponding to contrasts used for a categorical variable and a `contrasts_type` column equal to "treatment", "sum", "poly", "helmert" or "other".

Usage

```
tidy_add_contrasts(x, model = tidy_get_model(x))
```

Arguments

`x`	a tidy tibble
`model`	the corresponding model, if not attached to `x`

Details

If the `variable` column is not yet available in `x`, `tidy_identify_variables()` will be automatically applied.

See Also

Other tidy_helpers: tidy_add_coefficients_type(), tidy_add_estimate_to_reference_rows(), tidy_add_header_rows(), tidy_add_reference_rows(), tidy_add_term_labels(), tidy_add_variable_labels(), tidy_attach_model(), tidy_identify_variables(), tidy_plus_plus(), tidy_remove_intercept(), tidy_select_variables()

Examples

```
df <- Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes")))

df %>%
  glm(
    Survived ~ Class + Age + Sex,
    data = ., weights = .$n, family = binomial,
    contrasts = list(Age = contr.sum, Class = "contr.helmert")
  ) %>%
  tidy_and_attach() %>%
  tidy_add_contrasts()
```

`tidy_add_estimate_to_reference_rows` *Add an estimate value to references rows for categorical variables*

Description

For categorical variables with a treatment contrast (`stats::contr.treatment()`) or a SAS contrast (`stats::contr.SAS()`) will add an estimate equal to `0` (or `1` if `exponentiate = TRUE`) to the reference row.

Usage

```
tidy_add_estimate_to_reference_rows(
  x,
  exponentiate = attr(x, "exponentiate"),
  model = tidy_get_model(x),
  quiet = FALSE
)
```

Arguments

`x`	a tidy tibble
`exponentiate`	logical indicating whether or not to exponentiate the coefficient estimates. It should be consistent with the original call to `broom::tidy()`
`model`	the corresponding model, if not attached to `x`
`quiet`	logical argument whether broom.helpers should not return a message when requested output cannot be generated. Default is FALSE

Details

For categorical variables with a sum contrast (`stats::contr.sum()`), the estimate value of the reference row will be equal to the sum of all other coefficients multiplied by `-1` (eventually exponentiated if `exponentiate = TRUE`), and obtained with `stats::dummy.coef()`. For sum contrasts, the model coefficient corresponds to the difference of each level with the grand mean.

For other variables, no change will be made.

If the `reference_row` column is not yet available in `x`, `tidy_add_reference_rows()` will be automatically applied.

See Also

Other tidy_helpers: `tidy_add_coefficients_type()`, `tidy_add_contrasts()`, `tidy_add_header_rows()`, `tidy_add_reference_rows()`, `tidy_add_term_labels()`, `tidy_add_variable_labels()`, `tidy_attach_model()`, `tidy_identify_variables()`, `tidy_plus_plus()`, `tidy_remove_intercept()`, `tidy_select_variables()`

Examples

```
df <- Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(dplyr::across(where(is.character), factor))

df %>%
  glm(
    Survived ~ Class + Age + Sex,
    data = ., weights = .$n, family = binomial,
    contrasts = list(Age = contr.sum, Class = "contr.SAS")
  ) %>%
  tidy_and_attach(exponentiate = TRUE) %>%
  tidy_add_reference_rows() %>%
  tidy_add_estimate_to_reference_rows()

if (requireNamespace("gtsummary")) {
  glm(
    response ~ stage + grade * trt,
    gtsummary::trial,
    family = binomial,
    contrasts = list(
      stage = contr.treatment(4, base = 3),
      grade = contr.treatment(3, base = 2),
      trt = contr.treatment(2, base = 2)
    )
  ) %>%
    tidy_and_attach() %>%
    tidy_add_reference_rows() %>%
    tidy_add_estimate_to_reference_rows()
}
```

`tidy_add_header_rows` *Add header rows variables with several terms*

Description

For variables with several terms (usually categorical variables but could also be the case of continuous variables with polynomial terms or splines), `tidy_add_header_rows()` will add an additional row per variable, where `label` will be equal to `var_label`. These additional rows could be identified with `header_row` column.

Usage

```
tidy_add_header_rows(
  x,
  show_single_row = NULL,
  model = tidy_get_model(x),
  quiet = FALSE,
  strict = FALSE
)
```

Arguments

`x`	a tidy tibble
`show_single_row`	a vector indicating the names of binary variables that should be displayed on a single row. Accepts tidyselect syntax. Default is `NULL`. See also `all_dichotomous()`
`model`	the corresponding model, if not attached to `x`
`quiet`	logical argument whether broom.helpers should not return a message when requested output cannot be generated. Default is FALSE
`strict`	logical argument whether broom.helpers should return an error when requested output cannot be generated. Default is FALSE

Details

The `show_single_row` argument allows to specify a list of dichotomous variables that should be displayed on a single row instead of two rows.

The added `header_row` column will be equal to:

- `TRUE` for an header row;
- `FALSE` for a normal row of a variable with an header row;
- `NA` for variables without an header row.

If the `label` column is not yet available in `x`, `tidy_add_term_labels()` will be automatically applied.

See Also

Other tidy_helpers: tidy_add_coefficients_type(), tidy_add_contrasts(), tidy_add_estimate_to_reference_rows(), tidy_add_reference_rows(), tidy_add_term_labels(), tidy_add_variable_labels(), tidy_attach_model(), tidy_identify_variables(), tidy_plus_plus(), tidy_remove_intercept(), tidy_select_variables()

Examples

```
df <- Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes")))

res <- df %>%
  glm(
    Survived ~ Class + Age + Sex,
    data = ., weights = .$n, family = binomial,
    contrasts = list(Age = contr.sum, Class = "contr.SAS")
  ) %>%
  tidy_and_attach() %>%
  tidy_add_variable_labels(labels = list(Class = "Custom label for Class")) %>%
  tidy_add_reference_rows()
res %>% tidy_add_header_rows()
res %>% tidy_add_header_rows(show_single_row = all_dichotomous())

if (requireNamespace("gtsummary")) {
  glm(
    response ~ stage + grade * trt,
    gtsummary::trial,
    family = binomial,
    contrasts = list(
      stage = contr.treatment(4, base = 3),
      grade = contr.treatment(3, base = 2),
      trt = contr.treatment(2, base = 2)
    )
  ) %>%
    tidy_and_attach() %>%
    tidy_add_reference_rows() %>%
    tidy_add_header_rows()
}
```

---

tidy_add_reference_rows

Add references rows for categorical variables

Description

For categorical variables with a treatment contrast (stats::contr.treatment()), a SAS contrast (stats::contr.SAS()) or a sum contrast (stats::contr.sum()), add a reference row.

Usage

```
tidy_add_reference_rows(
  x,
  no_reference_row = NULL,
  model = tidy_get_model(x),
  quiet = FALSE
)
```

Arguments

x	a tidy tibble
no_reference_row	a vector indicating the name of variables for those no reference row should be added. Accepts tidyselect syntax. Default is NULL. See also all_categorical() and all_dichotomous()
model	the corresponding model, if not attached to x
quiet	logical argument whether broom.helpers should not return a message when requested output cannot be generated. Default is FALSE

Details

The added reference_row column will be equal to:

- TRUE for a reference row;
- FALSE for a normal row of a variable with a reference row;
- NA for variables without a reference row.

If the contrasts column is not yet available in x, tidy_add_contrasts() will be automatically applied.

tidy_add_reference_rows() will not populate the label of the reference term. It is therefore better to apply tidy_add_term_labels() after tidy_add_reference_rows() rather than before.

See Also

Other tidy_helpers: tidy_add_coefficients_type(), tidy_add_contrasts(), tidy_add_estimate_to_reference_row(), tidy_add_header_rows(), tidy_add_term_labels(), tidy_add_variable_labels(), tidy_attach_model(), tidy_identify_variables(), tidy_plus_plus(), tidy_remove_intercept(), tidy_select_variables()

Examples

```
df <- Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes")))

res <- df %>%
  glm(
    Survived ~ Class + Age + Sex,
    data = ., weights = .$n, family = binomial,
```

```
    contrasts = list(Age = contr.sum, Class = "contr.SAS")
  ) %>%
  tidy_and_attach()
res %>% tidy_add_reference_rows()
res %>% tidy_add_reference_rows(no_reference_row = all_dichotomous())
res %>% tidy_add_reference_rows(no_reference_row = "Class")

if (requireNamespace("gtsummary")) {
  glm(
    response ~ stage + grade * trt,
    gtsummary::trial,
    family = binomial,
    contrasts = list(
      stage = contr.treatment(4, base = 3),
      grade = contr.treatment(3, base = 2),
      trt = contr.treatment(2, base = 2)
    )
  ) %>%
    tidy_and_attach() %>%
    tidy_add_reference_rows()
}
```

`tidy_add_term_labels` *Add term labels*

Description

Will add term labels in a `label` column, based on:

1. labels provided in `labels` argument if provided;
2. factor levels for categorical variables coded with treatment, SAS or sum contrasts (the label could be customized with `categorical_terms_pattern` argument);
3. variable labels when there is only one term per variable;
4. term name otherwise.

Usage

```
tidy_add_term_labels(
  x,
  labels = NULL,
  interaction_sep = " * ",
  categorical_terms_pattern = "{level}",
  model = tidy_get_model(x),
  quiet = FALSE,
  strict = FALSE
)
```

Arguments

x	a tidy tibble
labels	an optional named list or named vector of custom term labels
interaction_sep	separator for interaction terms
categorical_terms_pattern	a glue pattern for labels of categorical terms with treatment or sum contrasts (see examples and model_list_terms_levels())
model	the corresponding model, if not attached to x
quiet	logical argument whether broom.helpers should not return a message when requested output cannot be generated. Default is FALSE
strict	logical argument whether broom.helpers should return an error when requested output cannot be generated. Default is FALSE

Details

If the variable_label column is not yet available in x, tidy_add_variable_labels() will be automatically applied. If the contrasts column is not yet available in x, tidy_add_contrasts() will be automatically applied.

It is possible to pass a custom label for any term in labels, including interaction terms.

See Also

Other tidy_helpers: tidy_add_coefficients_type(), tidy_add_contrasts(), tidy_add_estimate_to_reference_rows(), tidy_add_header_rows(), tidy_add_reference_rows(), tidy_add_variable_labels(), tidy_attach_model(), tidy_identify_variables(), tidy_plus_plus(), tidy_remove_intercept(), tidy_select_variables()

Examples

```
df <- Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes"))) %>%
  labelled::set_variable_labels(
    Class = "Passenger's class",
    Sex = "Sex"
  )

mod <- df %>%
  glm(Survived ~ Class * Age * Sex, data = ., weights = .$n, family = binomial)
mod %>%
  tidy_and_attach() %>%
  tidy_add_term_labels()
mod %>%
  tidy_and_attach() %>%
  tidy_add_term_labels(
    interaction_sep = " x ",
    categorical_terms_pattern = "{level} / {reference_level}"
  )
```

`tidy_add_variable_labels` *Add variable labels*

Description

Will add variable labels in a `var_label` column, based on:

1. labels provided in `labels` argument if provided;
2. variable labels defined in the original data frame with the `label` attribute (cf. `labelled::var_label()`);
3. variable name otherwise.

Usage

```
tidy_add_variable_labels(
  x,
  labels = NULL,
  interaction_sep = " * ",
  model = tidy_get_model(x),
  quiet = FALSE,
  strict = FALSE
)
```

Arguments

`x`	a tidy tibble
`labels`	an optional named list or named vector of custom variable labels
`interaction_sep`	separator for interaction terms
`model`	the corresponding model, if not attached to `x`
`quiet`	logical argument whether broom.helpers should not return a message when requested output cannot be generated. Default is FALSE
`strict`	logical argument whether broom.helpers should return an error when requested output cannot be generated. Default is FALSE

Details

If the `variable` column is not yet available in `x`, `tidy_identify_variables()` will be automatically applied.

It is possible to pass a custom label for an interaction term in `labels` (see examples).

See Also

Other tidy_helpers: `tidy_add_coefficients_type()`, `tidy_add_contrasts()`, `tidy_add_estimate_to_reference_rows()`, `tidy_add_header_rows()`, `tidy_add_reference_rows()`, `tidy_add_term_labels()`, `tidy_attach_model()`, `tidy_identify_variables()`, `tidy_plus_plus()`, `tidy_remove_intercept()`, `tidy_select_variables()`

Examples

```
df <- Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes"))) %>%
  labelled::set_variable_labels(
    Class = "Passenger's class",
    Sex = "Sex"
  )

df %>%
  glm(Survived ~ Class * Age * Sex, data = ., weights = .$n, family = binomial) %>%
  tidy_and_attach() %>%
  tidy_add_variable_labels(
    labels = list(
      "(Intercept)" = "Custom intercept",
      Sex = "Gender",
      "Class:Age" = "Custom label"
    )
  )
```

---

tidy_attach_model *Attach a full model to the tibble of model terms*

Description

To facilitate the use of broom helpers with pipe, it is recommended to attach the original model as an attribute to the tibble of model terms generated by `broom::tidy()`.

Usage

```
tidy_attach_model(x, model, .attributes = NULL)

tidy_and_attach(model, tidy_fun = broom::tidy, exponentiate = FALSE, ...)

tidy_get_model(x)

tidy_detach_model(x)
```

Arguments

`x`	a tibble of model terms
`model`	a model to be attached/tidied
`.attributes`	named list of additional attributes to be attached to `x`
`tidy_fun`	option to specify a custom tidier function
`exponentiate`	logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic, Poisson and Cox models, but a bad idea if there is no log or logit link; defaults to `FALSE`
`...`	other arguments passed to `tidy_fun()`

Details

tidy_attach_model() attach the model to a tibble already generated while tidy_and_attach() will apply broom::tidy() and attach the model.

Use tidy_get_model() to get the model attached to the tibble and tidy_detach_model() to remove the attribute containing the model.

See Also

Other tidy_helpers: tidy_add_coefficients_type(), tidy_add_contrasts(), tidy_add_estimate_to_reference_rows(), tidy_add_header_rows(), tidy_add_reference_rows(), tidy_add_term_labels(), tidy_add_variable_labels(), tidy_identify_variables(), tidy_plus_plus(), tidy_remove_intercept(), tidy_select_variables()

Examples

```
mod <- lm(Sepal.Length ~ Sepal.Width + Species, data = iris)
tt <- mod %>%
  tidy_and_attach(conf.int = TRUE)
tt
tidy_get_model(tt)
```

---

tidy_identify_variables

Identify the variable corresponding to each model coefficient

---

Description

tidy_identify_variables() will add to the tidy tibble three additional columns: variable, var_class, var_type and var_nlevels.

Usage

```
tidy_identify_variables(
  x,
  model = tidy_get_model(x),
  quiet = FALSE,
  strict = FALSE
)
```

Arguments

x	a tidy tibble
model	the corresponding model, if not attached to x
quiet	logical argument whether broom.helpers should not return a message when requested output cannot be generated. Default is FALSE
strict	logical argument whether broom.helpers should return an error when requested output cannot be generated. Default is FALSE

Details

It will also identify interaction terms and intercept(s).

`var_type` could be:

- `"continuous"`,
- `"dichotomous"` (categorical variable with 2 levels),
- `"categorical"` (categorical variable with 3 levels or more),
- `"intercept"`
- `"interaction"`
- `"unknown"` in the rare cases where `tidy_identify_variables()` will fail to identify the list of variables

For dichotomous and categorical variables, `var_nlevels` corresponds to the number of original levels in the corresponding variables.

See Also

`model_identify_variables()`

Other tidy_helpers: `tidy_add_coefficients_type()`, `tidy_add_contrasts()`, `tidy_add_estimate_to_reference_rows()`, `tidy_add_header_rows()`, `tidy_add_reference_rows()`, `tidy_add_term_labels()`, `tidy_add_variable_labels()`, `tidy_attach_model()`, `tidy_plus_plus()`, `tidy_remove_intercept()`, `tidy_select_variables()`

Examples

```
Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived, c("No", "Yes"))) %>%
  glm(Survived ~ Class + Age * Sex, data = ., weights = .$n, family = binomial) %>%
  tidy_and_attach() %>%
  tidy_identify_variables()

lm(
  Sepal.Length ~ poly(Sepal.Width, 2) + Species,
  data = iris,
  contrasts = list(Species = contr.sum)
) %>%
  tidy_and_attach(conf.int = TRUE) %>%
  tidy_identify_variables()
```

`tidy_plus_plus` *Tidy a model and compute additional informations*

Description

This function will apply sequentially:

- tidy_and_attach()
- tidy_identify_variables()
- tidy_add_contrasts()
- tidy_add_reference_rows()
- tidy_add_estimate_to_reference_rows()
- tidy_add_variable_labels()
- tidy_add_term_labels()
- tidy_add_header_rows()
- tidy_remove_intercept()
- tidy_select_variables()
- tidy_add_coefficients_type()
- tidy_detach_model()

Usage

```
tidy_plus_plus(
  model,
  tidy_fun = broom::tidy,
  conf.int = TRUE,
  exponentiate = FALSE,
  variable_labels = NULL,
  term_labels = NULL,
  interaction_sep = " * ",
  categorical_terms_pattern = "{level}",
  add_reference_rows = TRUE,
  no_reference_row = NULL,
  add_estimate_to_reference_rows = TRUE,
  add_header_rows = FALSE,
  show_single_row = NULL,
  intercept = FALSE,
  include = everything(),
  keep_model = FALSE,
  quiet = FALSE,
  strict = FALSE,
  ...
)
```

Arguments

model	a model to be attached/tidied
tidy_fun	option to specify a custom tidier function
conf.int	should confidence intervals be computed? (see broom::tidy())

exponentiate	logical indicating whether or not to exponentiate the coefficient estimates. This is typical for logistic, Poisson and Cox models, but a bad idea if there is no log or logit link; defaults to FALSE.
variable_labels	a named list or a named vector of custom variable labels
term_labels	a named list or a named vector of custom term labels
interaction_sep	separator for interaction terms
categorical_terms_pattern	a glue pattern for labels of categorical terms with treatment or sum contrasts (see model_list_terms_levels())
add_reference_rows	should reference rows be added?
no_reference_row	variables (accepts tidyselect notation) for those no reference row should be added, when add_reference_rows = TRUE
add_estimate_to_reference_rows	should an estimate value be added to reference rows?
add_header_rows	should header rows be added?
show_single_row	variables that should be displayed on a single row (accepts tidyselect notation), when add_header_rows is TRUE
intercept	should the intercept(s) be included?
include	variables to include. Accepts tidyselect syntax. Use - to remove a variable. Default is everything(). See also all_continuous(), all_categorical(), all_dichotomous() and all_interaction()
keep_model	should the model be kept as an attribute of the final result?
quiet	logical argument whether broom.helpers should not return a message when requested output cannot be generated. Default is FALSE
strict	logical argument whether broom.helpers should return an error when requested output cannot be generated. Default is FALSE
...	other arguments passed to tidy_fun()

See Also

Other tidy_helpers: tidy_add_coefficients_type(), tidy_add_contrasts(), tidy_add_estimate_to_reference_rows(), tidy_add_header_rows(), tidy_add_reference_rows(), tidy_add_term_labels(), tidy_add_variable_labels(), tidy_attach_model(), tidy_identify_variables(), tidy_remove_intercept(), tidy_select_variables()

Examples

```
ex1 <- lm(Sepal.Length ~ Sepal.Width + Species, data = iris) %>%
  tidy_plus_plus()
ex1
```

```
df <- Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(
    Survived = factor(Survived, c("No", "Yes"))
  ) %>%
  labelled::set_variable_labels(
    Class = "Passenger's class",
    Sex = "Gender"
  )

ex2 <- glm(
  Survived ~ Class + Age * Sex,
  data = df, weights = df$n,
  family = binomial
) %>%
  tidy_plus_plus(
    exponentiate = TRUE,
    add_reference_rows = FALSE,
    categorical_terms_pattern = "{level} / {reference_level}"
  )
ex2

if (requireNamespace("gtsummary")) {
  ex3 <- glm(
    response ~ poly(age, 3) + stage + grade * trt,
    na.omit(gtsummary::trial),
    family = binomial,
    contrasts = list(
      stage = contr.treatment(4, base = 3),
      grade = contr.sum
    )
  ) %>%
    tidy_plus_plus(
      exponentiate = TRUE,
      variable_labels = c(age = "Age (in years)"),
      add_header_rows = TRUE,
      show_single_row = all_dichotomous(),
      term_labels = c("poly(age, 3)3" = "Cubic age"),
      keep_model = TRUE
    )
  ex3
}
```

`tidy_remove_intercept` *Remove intercept(s)*

Description

Will remove terms where `var_type == "intercept"`.

Usage

```
tidy_remove_intercept(x, model = tidy_get_model(x))
```

Arguments

x	a tidy tibble
model	the corresponding model, if not attached to x

Details

If the variable column is not yet available in x, tidy_identify_variables() will be automatically applied.

See Also

Other tidy_helpers: tidy_add_coefficients_type(), tidy_add_contrasts(), tidy_add_estimate_to_reference_rows(), tidy_add_header_rows(), tidy_add_reference_rows(), tidy_add_term_labels(), tidy_add_variable_labels(), tidy_attach_model(), tidy_identify_variables(), tidy_plus_plus(), tidy_select_variables()

Examples

```
Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived)) %>%
  glm(Survived ~ Class + Age + Sex, data = ., weights = .$n, family = binomial) %>%
  tidy_and_attach() %>%
  tidy_remove_intercept()
```

---

tidy_select_variables *Select variables to keep/drop*

Description

Will remove unselected variables from the results. To remove the intercept, use tidy_remove_intercept().

Usage

```
tidy_select_variables(x, include = everything(), model = tidy_get_model(x))
```

Arguments

x	a tidy tibble
include	variables to include. Accepts tidyselect syntax. Use - to remove a variable. Default is everything(). See also all_continuous(), all_categorical(), all_dichotomous() and all_interaction()
model	the corresponding model, if not attached to x

Details

If the `variable` column is not yet available in `x`, `tidy_identify_variables()` will be automatically applied.

See Also

Other tidy_helpers: `tidy_add_coefficients_type()`, `tidy_add_contrasts()`, `tidy_add_estimate_to_reference_rows()`, `tidy_add_header_rows()`, `tidy_add_reference_rows()`, `tidy_add_term_labels()`, `tidy_add_variable_labels()`, `tidy_attach_model()`, `tidy_identify_variables()`, `tidy_plus_plus()`, `tidy_remove_intercept()`

Examples

```
res <- Titanic %>%
  dplyr::as_tibble() %>%
  dplyr::mutate(Survived = factor(Survived)) %>%
  glm(Survived ~ Class + Age * Sex, data = ., weights = .$n, family = binomial) %>%
  tidy_and_attach() %>%
  tidy_identify_variables()

res
res %>% tidy_select_variables()
res %>% tidy_select_variables(include = "Class")
res %>% tidy_select_variables(include = -c("Age", "Sex"))
res %>% tidy_select_variables(include = starts_with("A"))
res %>% tidy_select_variables(include = all_categorical())
res %>% tidy_select_variables(include = all_dichotomous())
res %>% tidy_select_variables(include = all_interaction())
res %>% tidy_select_variables(
  include = c("Age", all_categorical(dichotomous = FALSE), all_interaction())
)
```

Index