

Package: modelit (via r-universe)

July 13, 2024

Type Package

Version 1.4.6

Title Statistical Models for 'SciViews::R'

Description Create and use statistical models (linear, general, nonlinear...) with extensions to support rich-formatted tables, equations and plots for the 'SciViews::R' dialect.

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Depends R (>= 4.2.0), broom (>= 1.0.4)

Imports chart (>= 1.5.0), data.io (>= 1.5.0), flextable (>= 0.9.1), generics (>= 0.1.3), ggplot2 (>= 3.4.2), knitr (>= 1.42), modelr (>= 0.1.11), officer (>= 0.6.2), rlang (>= 1.1.1), stats (>= 4.2.0), svFlow (>= 1.2.0), tabularise (>= 0.6.0)

Suggests broom.mixed (>= 0.2.9.4), datasets (>= 4.2.0), dplyr (>= 1.1.4), equatags (>= 0.2.0), equatiomatic (>= 0.3.0), parsnip (>= 1.1.0), rmarkdown (>= 2.21), spelling (>= 2.2.1), testthat (>= 3.0.0)

Remotes SciViews/data.io, SciViews/chart, SciViews/svFlow, SciViews/tabularise

License MIT + file LICENSE

URL <https://github.com/SciViews/modelit>,
<https://www.sciviews.org/modelit/>

BugReports <https://github.com/SciViews/modelit/issues>

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

VignetteBuilder knitr

Encoding UTF-8

Language en-US

ByteCompile yes

Config/testthat/edition 3

Repository <https://sciviews.r-universe.dev>

RemoteUrl <https://github.com/SciViews/modelit>

RemoteRef HEAD

RemoteSha 94ac67bc83f4c81d609b2b18ed1e172031142ca2

Contents

modelit-package	3
as.function.lm	3
as.function.nls	4
chart.lm	5
chart.nls	7
equation.nls	8
fit_model	10
modelr-reexport	13
tabularise_coef.glm	14
tabularise_coef.lm	15
tabularise_coef.nls	16
tabularise_coef.summary.glm	18
tabularise_coef.summary.lm	18
tabularise_coef.summary.nls	19
tabularise_default.anova	20
tabularise_default.glm	22
tabularise_default.lm	23
tabularise_default.nls	23
tabularise_default.summary.glm	25
tabularise_default.summary.lm	26
tabularise_default.summary.nls	27
tabularise_glance.glm	28
tabularise_glance.lm	29
tabularise_glance.nls	31
tabularise_tidy.anova	32
tabularise_tidy.aov	33
tabularise_tidy.glm	34
tabularise_tidy.lm	35
tabularise_tidy.nls	36

Index

38

Description

The {modelit} package provides an extension to base R functions for model fitting like `lm()`, `glm()` or `nls()` with enhanced plots and utilitarian functions.

Important functions

- `fit_model()` creates a **model_fit** object that has many methods.
- `tabularise()` methods for **lm**, **glm**, **nls**, **model_fit**, **anova** and **aov** objects.
- `chart()` methods for **lm**, **glm**, **nls** and **model_fit** objects.
- `as.function()` transforms an **lm** or **nls** model into a function that can be plotted using `stat_function()`.

Description

Transform an lm or glm model that has only two variables into a function (useful for plotting, see examples).

Usage

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

Arguments

x	An lm or glm model
...	Further arguments to the method (not used for now)

Value

A function with argument x that returns the values predicted by the model for these values of x.

Examples

```

data("trees", package = "datasets")
trees_lm1 <- lm(Volume ~ Girth, data = trees)
trees_lm2 <- lm(Volume ~ Girth + I(Girth^2), data = trees)

# Compare these two models on a chart
library(chart)
chart(trees, Volume ~ Girth) +
  geom_point() +
  stat_function(fun = as.function(trees_lm1), col = "red") +
  stat_function(fun = as.function(trees_lm2), col = "blue")

# The created function can also be used for easy predictions
trees_fn1 <- as.function(trees_lm1)
trees_fn1(10:20) # Volume for Girth 10:20

```

as.function.nls

Transform an nls model into a function

Description

Transforming an nls model into a function could be useful to plot or otherwise manipulate it, see examples.

Usage

```

## S3 method for class 'nls'
as.function(x, ...)

```

Arguments

x	An nls model
...	Further arguments to the method (not used for now)

Value

A function with argument x that returns the values predicted by the model for these values of x.

Examples

```

data("ChickWeight", package = "datasets")
chick1 <- ChickWeight[ChickWeight$Chick == 1, ]

# Adjust a logistic curve
chick1_logis <- nls(weight ~ SSlogis(Time, Asym, xmid, scal), data = chick1)

# Show this on a ggplot
library(ggplot2)
p <- ggplot(chick1, aes(x = Time, y = weight)) +

```

```

    geom_point() +
    stat_function(fun = as.function(chick1_logis), col = "red")
  p

# Visually compare with another model (four-parameter logistic):
chick1_fpl <- nls(weight ~ SSfpl(Time, A, B, xmid, scal), data = chick1)

p + stat_function(fun = as.function(chick1_fpl), col = "blue")

```

chart.lm

Chart an lm or glm model or diagnose its residuals visually

Description

The methods `autoplot()` or `chart()` for **lm** or **glm** objects. If `type = model` (by default for `chart()`), a scatterplot with the model superimposed is produced, providing the model has only two numeric variables (or a combination of these). The other types allow to analyze the residuals of the model.

Usage

```

## S3 method for class 'lm'
chart(
  data,
  type = "model",
  ...,
  origdata = NULL,
  title,
  labels = "AUTO",
  name = deparse(substitute(data)),
  lang = getOption("data.io_lang", "en"),
  env = parent.frame()
)

autoplot.lm(
  object,
  origdata = NULL,
  type = c("model", "resfitted", "qqplot", "scalelocation", "cooksd", "resleverage",
    "cookleverage", "reshist", "resautocor"),
  title,
  xlab,
  ylab,
  ...,
  name = deparse(substitute(object)),
  lang = getOption("data.io_lang", "en"),
  env = parent.frame()
)

```

Arguments

<code>data</code>	A lm or glm model.
<code>type</code>	The type of plot: "model", "resfitted", "qqplot", "scalelocation", "cooks", "resleverage", "cookleverage", "reshist" or "resautocor". For <code>chart()</code> , can also be provided as <code>chart\$type(...)</code> . <code>chart()</code> also uses "residuals" that constructs a combined figure with resfitted, qqplot, scalelocation and resleverage.
<code>...</code>	Additional arguments passed to the chart.
<code>origdata</code>	The original dataset this model was fitted to. Only required for <code>type = model</code> and in case untransformed X variable is not in the model.
<code>title</code>	A title for the plot. If not provided, a default title is computed.
<code>labels</code>	A vector of four character strings, one for each plot done with <code>chart\$residuals()</code> .
<code>name</code>	The name of the model. If not provided, it is the name of the model object by default.
<code>lang</code>	The language to use for titles and labels, currently only "en" or "fr".
<code>env</code>	The environment to evaluate code. It is <code>parent.frame()</code> by default, and there is no reasons to change it, unless you really know what you are doing!
<code>object</code>	Idem
<code>xlab</code>	A label for the X axis. A default label is proposed if it is not provided.
<code>ylab</code>	A label for the Y axis (with default if not provided).

Value

The ggplot object produced.

Examples

```
library(chart)
data(trees, package = "datasets")
trees_lm <- lm(Volume ~ Girth, data = trees)
chart(trees_lm) # origdata not needed because untransformed variables
# Residuals analysis
chart$resfitted(trees_lm)
chart$qqplot(trees_lm)
chart$scalelocation(trees_lm)
chart$cooks(trees_lm)
chart$resleverage(trees_lm)
chart$cookleverage(trees_lm)
chart$reshist(trees_lm, bins = 15)
chart$resautocor(trees_lm)

# The four most important residual analysis plots in one figure
chart$residuals(trees_lm)

trees_lm2 <- lm(Volume ~ log(Girth), data = trees)
chart(trees_lm2, origdata = trees) # origdata needed, cf. transformed Girth
trees_lm3 <- lm(Volume ~ Girth + Height, data = trees)
```

```
# chart(trees_lm3) # Error because more than 2 variables!
# Polynomial regressions work too
trees_lm4 <- lm(Volume ~ I(Girth^2) + Girth, data = trees)
chart(trees_lm4)
# or using poly()
trees_lm5 <- lm(Volume ~ poly(Girth, 3), data = trees)
chart(trees_lm5, origdata = trees) # origdata required here!
```

chart.nls

Chart an nls model or diagnose its residuals visually

Description

The methods `autoplot()` or `chart()` for **nls** objects. If `type = model` (by default for `chart()`), a scatterplot with the model superimposed is produced. The other types allow to analyze the residuals of the model.

Usage

```
## S3 method for class 'nls'
chart(
  data,
  type = "model",
  ...,
  title,
  labels = "AUTO",
  name = deparse(substitute(data)),
  lang = getOption("data.io_lang", "en"),
  env = parent.frame()
)

autoplot.nls(
  object,
  type = c("model", "resfitted", "qqplot", "scalelocation", "reshist", "resautocor"),
  title,
  xlab,
  ylab,
  ...,
  name = deparse(substitute(object)),
  lang = getOption("data.io_lang", "en"),
  env = parent.frame()
)
```

Arguments

`data` A **nls** model.

type	The type of plot: "model", "resfitted", "qqplot", "scalelocation", "reshist" or "resautocor". For chart(), can also be provided as chart\$type(...). chart() also uses "residuals" that constructs a combined figure with resfitted, qqplot, scalelocation and resautocor.
...	Additional arguments passed to the chart.
title	A title for the plot. If not provided, a default title is computed.
labels	A vector of four character strings, one for each plot done with chart\$residuals().
name	The name of the model. If not provided, it is the name of the model object by default.
lang	The language to use for titles and labels, currently only "en" or "fr".
env	The environment to evaluate code. It is parent.frame() by default, and there is no reasons to change it, unless you really know what you are doing!
object	Idem
xlab	A label for the X axis. A default label is proposed if it is not provided.
ylab	A label for the Y axis (with default if not provided).

Value

The ggplot object produced.

Examples

```
data("ChickWeight", package = "datasets")
chick1 <- ChickWeight[ChickWeight$Chick == 1, ]

# Adjust a logistic curve
chick1_logis <- nls(weight ~ SSlogis(Time, Asym, xmid, scal), data = chick1)
library(chart)
chart(chick1_logis)
# Residuals analysis
chart$resfitted(chick1_logis)
chart$qqplot(chick1_logis)
chart$scalelocation(chick1_logis)
chart$reshist(chick1_logis, bins = 15)
chart$resautocor(chick1_logis)

# The four most important residual analysis plots in one figure
chart$residuals(chick1_logis)
```

Description

Create the model equation of several self-starting nonlinear models available in the stats package.

Usage

```
## S3 method for class 'nls'
equation(
  object,
  ital_vars = FALSE,
  use_coefs = FALSE,
  coef_digits = 2L,
  fix_signs = TRUE,
  var_names = NULL,
  op_latex = c("\\cdot", "\\times"),
  ...
)

## S3 method for class 'summary.nls'
equation(
  object,
  ital_vars = FALSE,
  use_coefs = FALSE,
  coef_digits = 2L,
  fix_signs = TRUE,
  var_names = NULL,
  op_latex = c("\\cdot", "\\times"),
  ...
)
```

Arguments

<code>object</code>	An nls or summary.nls object.
<code>ital_vars</code>	Logical, defaults to FALSE. Should the variable names not be wrapped in the <code>\operatorname</code> command?
<code>use_coefs</code>	Logical, defaults to FALSE. Should the actual model estimates be included in the equation instead of math symbols? If TRUE, <code>var_names=</code> is ignored.
<code>coef_digits</code>	Integer, defaults to 2. The number of decimal places to round to when displaying model estimates with <code>use_coefs = TRUE</code> .
<code>fix_signs</code>	Logical, defaults to TRUE. If disabled, coefficient estimates that are negative are preceded with a + (e.g. $5(x) + -3(z)$). If enabled, the + - is replaced with a - (e.g. $5(x) - 3(z)$).
<code>var_names</code>	A named character vector as <code>c(old_var_name = "new name")</code>
<code>op_latex</code>	The LaTeX product operator character to use in fancy scientific notation, either <code>\cdot</code> (by default), or <code>\times</code> .
<code>...</code>	Additional arguments (not used yet).

Value

A character string with a LaTeX equation.

Examples

```
equation <- tabularise::equation
chick1 <- ChickWeight[ChickWeight$Chick == 1, ]
chick1_nls <- nls(data = chick1, weight ~ SSlogis(Time, Asym, xmid, scal))
summary(chick1_nls)

equation(chick1_nls)
equation(summary(chick1_nls))

chick1_nls2 <- nls(data = chick1,
  weight ~ SSlogis(Time, Asym = A, xmid = x, scal = scale))
summary(chick1_nls2)

equation(chick1_nls2)
equation(summary(chick1_nls2))

equation(summary(chick1_nls2), var_names = c(
  weight = "Body weight [gm]",
  Time = "Number of days"))
```

fit_model

Fit a parsnip model and manipulate it as a base R model like lm

Description

`fit_model()` takes a **model_spec** object from {parsnip} and it fits it. Then, usual methods like `summary()`, or `coef()` can be applied directly on it, while it can still be used as the {tidymodels} recommends it.

Usage

```
fit_model(data, formula, ..., type = NULL, env = parent.frame())

## S3 method for class 'model_fit'
summary(object, ...)

## S3 method for class 'model_fit'
anova(object, ...)

## S3 method for class 'model_fit'
plot(x, y, ...)

## S3 method for class 'model_fit'
chart(data, ..., type = "model", env = parent.frame())

## S3 method for class 'model_fit'
as.function(x, ...)
```

```
## S3 method for class 'model_fit'  
coef(object, ...)  
  
## S3 method for class 'model_fit'  
vcov(object, ...)  
  
## S3 method for class 'model_fit'  
confint(object, parm, level = 0.95, ...)  
  
## S3 method for class 'model_fit'  
fitted(object, ...)  
  
## S3 method for class 'model_fit'  
residuals(object, ...)  
  
## S3 method for class 'model_fit'  
rstandard(model, ...)  
  
## S3 method for class 'model_fit'  
cooks.distance(model, ...)  
  
## S3 method for class 'model_fit'  
hatvalues(model, ...)  
  
## S3 method for class 'model_fit'  
deviance(object, ...)  
  
## S3 method for class 'model_fit'  
AIC(object, ..., k = 2)  
  
## S3 method for class 'model_fit'  
BIC(object, ...)  
  
## S3 method for class 'model_fit'  
family(object, ...)  
  
## S3 method for class 'model_fit'  
nobs(object, ...)  
  
## S3 method for class 'model_fit'  
formula(x, ...)  
  
## S3 method for class 'model_fit'  
variable.names(object, ...)  
  
## S3 method for class 'model_fit'  
labels(object, ...)
```

Arguments

data	A data frame (or a model_fit object for <code>chart()</code>)
formula	A formula specifying a model
...	Further arguments passed to the method
type	The type of model fitting, specified by a model_spec object or the name of such an object in a string
env	The environment where to evaluate type. It is <code>parent.frame()</code> by default and you probably have no reasons to change it, unless you really know what you are doing!
object	A model_fit object
x	Idem
y	Not used here
parm	Specification of parameters for the confidence intervals (vector of numbers or of names). If missing, all parameters are considered.
level	Confidence level required.
model	Idem
k	The penalty per parameter to be used in the AIC (by default, $k = 2$).

Value

A **model_fit** object.

Examples

```
library(parsnip)
data(trees, package = "datasets")

# Take the habit to prefix your regression model specs by `reg_`
reg_lm <- linear_reg(mod = "regression", engine = "lm")
trees_fit <- fit_model$reg_lm(data = trees, Volume ~ Girth)

# You can use summary(), AIC(), anova(), tidy(), glance(), etc. directly
summary(trees_fit)
anova(trees_fit)
AIC(trees_fit)
coef(trees_fit)
library(chart)
chart(trees_fit)
# etc.
```

modelr-reexport *Reexport of modelr functions*

Description

`add_predictions()` and `add_residuals()` are pipe-friendly functions to add predictions or residuals to a data frame. `geom_ref_line()` adds a vertical or horizontal reference line. `rmse()` (the root-mean-squared-error), `[mae[]]` (the mean absolute error), `qae()` (the quantiles of absolute error) and `rsquare()` (the variance of the predictions divided by the variance of the response) are useful model metrics.

Usage

```
add_predictions(data, model, var = "pred", type = NULL)

add_residuals(data, model, var = "resid")

geom_ref_line(h, v, color = "red", colour = color, size = 1)

rmse(model, data)

mae(model, data)

qae(model, data, probs = c(0.05, 0.25, 0.5, 0.75, 0.95))

rsquare(model, data)
```

Arguments

<code>data</code>	A data frame
<code>model</code>	A model that has a <code>predict()</code> method.
<code>var</code>	A string with the name of the predictions or residuals variable (by default, it is "pred" and "resid" respectively)
<code>type</code>	If the model's <code>predict()</code> method has a <code>type=</code> argument, you can give it here.
<code>h</code>	Position of the horizontal reference line
<code>v</code>	Position of the vertical reference line
<code>color</code>	The color of the reference line
<code>colour</code>	Same as above (use the one you prefer)
<code>size</code>	The width of the reference line
<code>probs</code>	A numeric vector of probabilities

Value

A function with argument `x` that returns the values predicted by the model for these values of `x`.

Examples

```

data(trees, package = "datasets")
trees_lm <- lm(Volume ~ Girth + I(Girth^2), data = trees)
rmse(trees_lm, trees)
rsquare(trees_lm, trees)
mae(trees_lm, trees)
qae(trees_lm, trees, probs = c(0, 0.25, 0.5, 0.75, 1)) # Resids five numbers

add_predictions(trees, trees_lm)
add_residuals(trees, trees_lm)

library(chart)
chart(trees_lm) +
  geom_ref_line(h = 0) # Not particularly useful here, just an example

```

tabularise_coef.glm *Create a rich-formatted table using the coefficients of a glm object*

Description

Extract and format the table of coefficients from a **glm** object, similar to `stats::coef()`, but in a rich-formatted **flextable** object.

Usage

```

## S3 method for class 'glm'
tabularise_coef(
  data,
  header = TRUE,
  title = NULL,
  equation = header,
  auto.labs = TRUE,
  origdata = NULL,
  labs = NULL,
  lang = getOption("data.io_lang", "en"),
  ...,
  kind = "ft",
  env = parent.frame()
)

```

Arguments

data	A glm object
header	If TRUE (by default), add a header to the table
title	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).

equation	If TRUE (by default), add an equation to the table header. The equation can also be passed in the form of a character string (LaTeX).
auto.labs	If TRUE (by default), use labels (and units) automatically from origdata=.
origdata	The original data set this model was fitted to. By default it is NULL and labels of the original data set are not used.
labs	Labels to change the names of elements in the term column of the table. By default, it is NULL and no change is performed.
lang	The natural language to use. The default value can be set with, e.g., options(data.io_lang = "fr") for French.
...	Additional arguments (not used yet).
kind	The kind of table to produce: "t" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object is returned. You can print it in different formats (HTML, LaTeX, Word, PowerPoint) or rearrange it with the {flextable} functions.

Examples

```
iris_glm <- glm(data = iris, Petal.Length ~ Sepal.Length)
tabularise::tabularise$coef(iris_glm)
```

tabularise_coef.lm *Create a rich-formatted table using the coefficients of an lm object*

Description

This function extracts and formats the table of coefficients from an **lm** object, similar to `stats::coef()`, but in a rich-formatted table using {flextable}.

Usage

```
## S3 method for class 'lm'
tabularise_coef(
  data,
  header = TRUE,
  title = NULL,
  equation = header,
  auto.labs = TRUE,
  origdata = NULL,
  labs = NULL,
  lang = getOption("data.io_lang", "en"),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

data	An lm object
header	If TRUE (by default), add a header to the table
title	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
equation	If TRUE (by default), add a equation to the table header. The equation can also be passed in the form of a character string.
auto.labs	If TRUE (by default), use labels (and units) automatically from data or origdata=.
origdata	The original data set this model was fitted to. By default it is NULL and no label is used.
labs	Labels to change the names of elements in the term column of the table. By default it is NULL and nothing is changed.
lang	The natural language to use. The default value can be set with, e.g., options(data.io_lang = "fr") for French.
...	Additional arguments
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object that you can print in different formats (HTML, LaTeX, Word, PowerPoint) or rearrange with the {flextable} functions.

Examples

```
iris_lm <- lm(data = iris, Petal.Length ~ Sepal.Length)
tabularise::tabularise$coef(iris_lm)
```

tabularise_coef.nls *Create a rich-formatted table using the coefficients of the nls object*

Description

This method extracts and formats the coefficients from an **nls** object, similar to `stats::coef()`, but in flextable object.

Usage

```
## S3 method for class 'nls'
tabularise_coef(
  data,
  header = TRUE,
  title = NULL,
  equation = header,
  lang = getOption("data.io_lang", "en"),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

data	An nls object.
header	If TRUE (by default), add a title to the table.
title	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
equation	If TRUE (by default), add the equation of the model
lang	The language to use. The default value can be set with, e.g., <code>options(data.io_lang = "fr")</code> for French.
...	Additional arguments.
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object that you can print in different forms or rearrange with the {flextable} functions.

Examples

```
data("ChickWeight", package = "datasets")
chick1 <- ChickWeight[ChickWeight$Chick == 1, ]
# Adjust a logistic curve
chick1_logis <- nls(data = chick1, weight ~ SSlogis(Time, Asym, xmid, scal))

tabularise::tabularise$coef(chick1_logis)
```

```
tabularise_coef.summary.glm
```

Create a rich-formatted table using the table of coefficients of the summary.glm object

Description

Create a rich-formatted {flextable} object with the table of coefficients from the `summary()` of a `glm` object.

Usage

```
## S3 method for class 'summary.glm'
tabularise_coef(data, ..., kind = "ft", env = parent.frame())
```

Arguments

<code>data</code>	A summary.glm object
<code>...</code>	Additional arguments passed to <code>tabularise_tidy.glm()</code>
<code>kind</code>	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate the model.

Value

A **flextable** object that you can print in different formats (HTML, LaTeX, Word, PowerPoint) or rearrange with the {flextable} functions.

Examples

```
iris_glm <- glm(data = iris, Petal.Length ~ Sepal.Length)
iris_glm_sum <- summary(iris_glm)
tabularise::tabularise$coef(iris_glm_sum)
```

```
tabularise_coef.summary.lm
```

Create a rich-formatted table using the table of coefficients of the summary.lm object

Description

Create a rich-formatted table using the table of coefficients of the `summary.lm` object

Usage

```
## S3 method for class 'summary.lm'
tabularise_coef(data, ..., kind = "ft", env = parent.frame())
```

Arguments

data	A summary.lm object
...	Additional arguments passed to <code>tabularise_tidy.lm()</code>
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate the model.

Value

A **flextable** object you can print in different formats (HTML, LaTeX, Word, PowerPoint) or rearrange with the `{flextable}` functions.

Examples

```
iris_lm <- lm(data = iris, Petal.Length ~ Sepal.Length)
iris_lm_sum <- summary(iris_lm)
tabularise::tabularise$coef(iris_lm_sum)
```

```
tabularise_coef.summary.nls
```

Create a rich-formatted table using the table of coefficients of the summary.nls object

Description

This function extracts and formats the table of coefficients from a **summary.nls** object, similar to `stats::coef()`, but in flextable object.

Usage

```
## S3 method for class 'summary.nls'
tabularise_coef(
  data,
  header = TRUE,
  title = NULL,
  equation = header,
  lang = getOption("data.io_lang", "en"),
  show.signif.stars = getOption("show.signif.stars", TRUE),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

data	A summary.nls object.
header	If TRUE (by default), add a title to the table.
title	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
equation	Add equation of the model to the table. If TRUE, <code>equation()</code> is used. The equation can also be passed in the form of a character string (LaTeX equation).
lang	The language to use. The default value can be set with, e.g., <code>options(data.io_lang = "fr")</code> for French.
show.signif.stars	If TRUE (by default), add the significance stars to the table.
...	Additional arguments passed to <code>equation()</code>
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object that you can print in different forms or rearrange with the `{flextable}` functions.

Examples

```
data("ChickWeight", package = "datasets")
chick1 <- ChickWeight[ChickWeight$Chick == 1, ]
# Adjust a logistic curve
chick1_logis <- nls(data = chick1, weight ~ SSlogis(Time, Asym, xmid, scal))
chick1_logis_sum <- summary(chick1_logis)

tabularise::tabularise$coef(chick1_logis_sum)
tabularise::tabularise$coef(chick1_logis_sum, header = FALSE, equation = TRUE)
```

tabularise_default.anova

Create a rich-formatted table from an anova object

Description

Create a rich-formatted table from an anova object

Usage

```
## S3 method for class 'anova'
tabularise_default(
  data,
  header = TRUE,
  title = header,
  auto.labs = TRUE,
  origdata = NULL,
  labs = NULL,
  lang = getOption("data.io_lang", "en"),
  show.signif.stars = getOption("show.signif.stars", TRUE),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

<code>data</code>	An anova object
<code>header</code>	If TRUE (by default), add a header to the table
<code>title</code>	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
<code>auto.labs</code>	If TRUE (by default), use labels (and units) automatically (from origdata=)
<code>origdata</code>	The original data set used for the ANOVA. By default it is NULL. Used to extract labels that are lost in the anova object.
<code>labs</code>	Labels to change the default names in the term column of the table. By default it is NULL and nothing is changed.
<code>lang</code>	The natural language to use. The default value is set with, e.g., <code>options(data.io_lang = "fr")</code> for French.
<code>show.signif.stars</code>	If TRUE, add the significance stars to the table. The default is taken from <code>getOption("show.signif.stars")</code> .
<code>...</code>	Additional arguments (not used for now)
<code>kind</code>	The kind of table to produce: "tt" for tynytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate lazyeval expressions (not used for now)

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions.

Examples

```
iris_anova <- anova(lm(data = iris, Petal.Length ~ Species))
tabularise::tabularise(iris_anova)
```

`tabularise_default.glm`*Create a rich-formatted table from a glm object*

Description

Create a rich-formatted table from a glm object

Usage

```
## S3 method for class 'glm'  
tabularise_default(  
  data,  
  footer = TRUE,  
  lang = getOption("data.io_lang", "en"),  
  ...,  
  kind = "ft",  
  env = parent.frame()  
)
```

Arguments

<code>data</code>	A glm object
<code>footer</code>	If TRUE (by default), add a footer to the table
<code>lang</code>	The natural language to use. The default value can be set with, e.g., <code>options(data.io_lang = "fr")</code> for French.
<code>...</code>	Additional arguments passed to <code>tabularise_coef.glm()</code>
<code>kind</code>	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate the model.

Value

A **flextable** object is returned. You can print it in different formats (HTML, LaTeX, Word, Power-Point) or rearrange it with the `{flextable}` functions.

Examples

```
iris_glm <- glm(data = iris, Petal.Length ~ Sepal.Length)  
tabularise::tabularise(iris_glm)
```

tabularise_default.lm *Create a rich-formatted table from an lm object*

Description

The default `tabularise()` method for **lm** objects create a minimalist table with result of the analysis in a rich-formatted tabular presentation.

Usage

```
## S3 method for class 'lm'  
tabularise_default(data, ..., kind = "ft", env = parent.frame())
```

Arguments

data	An lm object
...	Additional arguments passed to <code>tabularise_coef.lm()</code>
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate the model.

Value

A **flextable** object that you can print in different formats (HTML, LaTeX, Word, PowerPoint) or rearrange with the {flextable} functions.

Examples

```
iris_lm <- lm(data = iris, Petal.Length ~ Sepal.Length)  
tabularise::tabularise(iris_lm)
```

tabularise_default.nls

Create a rich-formatted table from a nls object

Description

This method extracts and formats an **nls** object, similar to `print()`, but in flextable object.

Usage

```
## S3 method for class 'nls'
tabularise_default(
  data,
  header = TRUE,
  title = NULL,
  equation = header,
  footer = TRUE,
  lang = getOption("data.io_lang", "en"),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

<code>data</code>	An nls object.
<code>header</code>	If TRUE (by default), add a title to the table.
<code>title</code>	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
<code>equation</code>	Add equation of the model to the table. If TRUE, <code>equation()</code> is used. The equation can also be passed in the form of a character string (LaTeX equation).
<code>footer</code>	If TRUE (by default), add a footer to the table.
<code>lang</code>	The language to use. The default value can be set with, e.g., <code>options(data.io_lang = "fr")</code> for French.
<code>...</code>	Additional arguments. Not used.
<code>kind</code>	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object that you can print in different forms or rearrange with the `{flextable}` functions.

Examples

```
data("ChickWeight", package = "datasets")
chick1 <- ChickWeight[ChickWeight$Chick == 1, ]
# Adjust a logistic curve
chick1_logis <- nls(data = chick1, weight ~ SSlogis(Time, Asym, xmid, scal))

tabularise::tabularise(chick1_logis)
```

`tabularise_default.summary.glm`*Create a rich-formatted table from a `summary.glm` object*

Description

Create a rich-formatted table version of the `summary()` of a `glm` object.

Usage

```
## S3 method for class 'summary.glm'  
tabularise_default(  
  data,  
  footer = TRUE,  
  lang = getOption("data.io_lang", "en"),  
  ...,  
  kind = "ft",  
  env = parent.frame()  
)
```

Arguments

<code>data</code>	A <code>summary.glm</code> object
<code>footer</code>	If TRUE (by default), add a footer to the table
<code>lang</code>	The natural language to use. The default value can be set with, e.g., <code>options(data.io_lang = "fr")</code> for French.
<code>...</code>	Additional arguments passed to <code>tabularise_coef.summary.glm()</code>
<code>kind</code>	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate the model.

Value

A `flextable` object that you can print in different form or rearrange with the `{flextable}` functions.

Examples

```
iris_glm <- glm(data = iris, Petal.Length ~ Sepal.Length)  
iris_glm_sum <- summary(iris_glm)  
tabularise::tabularise(iris_glm_sum)
```

```
tabularise_default.summary.lm
```

Create a rich-formatted table from an `summary.lm` object

Description

Create a rich-formatted table from an `summary.lm` object

Usage

```
## S3 method for class 'summary.lm'
tabularise_default(
  data,
  footer = TRUE,
  lang = getOption("data.io_lang", "en"),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

<code>data</code>	A summary.lm object
<code>footer</code>	If TRUE (by default), add a footer to the table
<code>lang</code>	The natural language to use. The default value can be set with, e.g., <code>options(data.io_lang = "fr")</code> for French.
<code>...</code>	Additional arguments passed to <code>tabularise_coef.summary.lm()</code>
<code>kind</code>	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate the model.

Value

A **flextable** object you can print in different formats (HTML, LaTeX, Word, PowerPoint) or rearrange with the `{flextable}` functions.

Examples

```
iris_lm <- lm(data = iris, Petal.Length ~ Sepal.Length)
iris_lm_sum <- summary(iris_lm)
tabularise::tabularise(iris_lm_sum)
```

 tabularise_default.summary.nls

Create a rich-formatted table from the summary of a nls object

Description

Create a table of a **summary.nls** object. This table looks like the output of `print.summary.nls()` but richly formatted. The `tabularise_coef()` function offers more customization options for this object.

Usage

```
## S3 method for class 'summary.nls'
tabularise_default(
  data,
  header = TRUE,
  title = NULL,
  equation = header,
  footer = TRUE,
  lang = getOption("data.io_lang", "en"),
  show.signif.stars = getOption("show.signif.stars", TRUE),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

<code>data</code>	A summary.nls object.
<code>header</code>	If TRUE (by default), add a header to the table
<code>title</code>	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
<code>equation</code>	Add equation of the model to the table. If TRUE, <code>equation()</code> is used. The equation can also be passed in the form of a character string (LaTeX equation).
<code>footer</code>	If TRUE (by default), add a footer to the table.
<code>lang</code>	The language to use. The default value can be set with, e.g. <code>options(data.io_lang = "fr")</code> for French.
<code>show.signif.stars</code>	If TRUE (by default), add the significance stars to the table.
<code>...</code>	Additional arguments (Not used).
<code>kind</code>	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object that you can print in different forms or rearrange with the {flextable} functions.

See Also

`tabularise::tabularise()`, `tabularise::tabularise_tidy()`, `tabularise_coef.summary.nls()`

Examples

```
data("ChickWeight", package = "datasets")
chick1 <- ChickWeight[ChickWeight$Chick == 1, ]

# Adjust a logistic curve
chick1_logis <- nls(data = chick1, weight ~ SSlogis(Time, Asym, xmid, scal))
chick1_logis_sum <- summary(chick1_logis)

tabularise::tabularise(chick1_logis_sum)
tabularise::tabularise(chick1_logis_sum, footer = FALSE)
```

`tabularise_glance.glm` *Create a glance version of the glm object as a rich-formatted table*

Description

Turn the glance of **glm** object into a rich-formatted table with {flextable}. The table can be printed in different formats (HTML, LaTeX, Word, PowerPoint), or rearranged later on.

Usage

```
## S3 method for class 'glm'
tabularise_glance(
  data,
  header = TRUE,
  title = NULL,
  equation = TRUE,
  auto.labs = TRUE,
  origdata = NULL,
  labs = NULL,
  lang = getOption("data.io_lang", "en"),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

data	A glm object
header	If TRUE (by default), add an header to the table
title	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
equation	If TRUE (by default), add an equation to the table header. The equation can also be passed in the form of a character string (LaTeX).
auto.labs	If TRUE (by default), use labels (and units) automatically from origdata=.
origdata	The original data set this model was fitted to. By default it is NULL and original labels are not used.
labs	Labels to change the names of elements in the term column of the table. By default it is NULL and nothing is changed.
lang	The natural language to use. The default value can be set with, e.g., options(data.io_lang = "fr") for French.
...	Additional arguments passed to <code>tabularise::equation()</code>
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object is produced that you can print in different formats (HTML, LaTeX, Word, PowerPoint) or rearrange with the {flextable} functions.

Examples

```
iris_glm <- glm(data = iris, Petal.Length ~ Sepal.Length)
tabularise::tabularise$glance(iris_glm)
```

tabularise_glance.lm *Glance version of the lm object into a flextable object*

Description

Create a rich-formatted table with the 'glance' information from an **lm** object.

Usage

```
## S3 method for class 'lm'
tabularise_glance(
  data,
  header = TRUE,
  title = NULL,
  equation = TRUE,
```

```

    auto.labs = TRUE,
    origdata = NULL,
    labs = NULL,
    lang = getOption("data.io_lang", "en"),
    ...,
    kind = "ft",
    env = parent.frame()
  )

```

Arguments

data	An lm object
header	If TRUE (by default), add a header to the table
title	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
equation	If TRUE (by default), add a equation to the table header. The equation can also be passed in the form of a character string (LaTeX).
auto.labs	If TRUE (by default), use labels (and units) automatically from data or origdata=.
origdata	The original data set this model was fitted to. By default it is NULL and no label is used (only the name of the variables).
labs	Labels to change the names of elements in the term column of the table. By default it is NULL and nothing is changed.
lang	The natural language to use. The default value can be set with, e.g., options(data.io_lang = "fr") for French.
...	Additional arguments passed to <code>tabularise::equation()</code>
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object that you can print in different form or rearrange with the `{flextable}` functions.

Examples

```

iris_lm <- lm(data = iris, Petal.Length ~ Sepal.Length)
tabularise::tabularise$glance(iris_lm)

```

tabularise_glance.nls *Glance version of the nls object into a flextable object*

Description

Extract the information contained in an **nls** object in a table as it could be obtained by `broom::glance()`. Here, the table is nicely formatted as an (almost) publication-ready form (good for informal reports, notebooks, etc).

Usage

```
## S3 method for class 'nls'
tabularise_glance(
  data,
  header = TRUE,
  title = NULL,
  equation = header,
  lang = getOption("data.io_lang", "en"),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

<code>data</code>	An nls object.
<code>header</code>	If TRUE (by default), add a title to the table.
<code>title</code>	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
<code>equation</code>	Add equation of the model to the table. If TRUE, <code>equation()</code> is used. The equation can also be passed in the form of a character string (LaTeX).
<code>lang</code>	The language to use. The default value can be set with, e.g., <code>options(data.io_lang = "fr")</code> for French.
<code>...</code>	Additional arguments passed to <code>equation()</code>
<code>kind</code>	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
<code>env</code>	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object that you can print in different forms or rearrange with the {flextable} functions.

See Also

`tabularise::tabularise_glance()`, `tabularise_coef.summary.nls()`

Examples

```

data("ChickWeight", package = "datasets")
chick1 <- ChickWeight[ChickWeight$Chick == 1, ]

# Adjust a logistic curve
chick1_logis <- nls(data = chick1, weight ~ SSlogis(Time, Asym, xmid, scal))

tabularise::tabularise$glance(chick1_logis)
tabularise::tabularise$glance(chick1_logis, lang = "fr")

```

tabularise_tidy.anova *Tidy version of the anova object into a flextable object*

Description

Tidy version of the anova object into a flextable object

Usage

```

## S3 method for class 'anova'
tabularise_tidy(
  data,
  header = TRUE,
  title = header,
  auto.labs = TRUE,
  origdata = NULL,
  labs = NULL,
  lang = getOption("data.io_lang", "en"),
  show.signif.stars = getOption("show.signif.stars", TRUE),
  ...,
  kind = "ft",
  env = parent.frame()
)

```

Arguments

data	An anova object
header	If TRUE (by default), add a header to the table
title	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
auto.labs	If TRUE (by default), use labels (and units) from origdata=.
origdata	The original data set used for the ANOVA (used for changing the labels). By default, it is NULL.
labs	Labels to use to change the names of elements in the term column. By default, it is NULL.

lang	The natural language to use. The default value can be set with, e.g., <code>options(data.io_lang = "fr")</code> for French.
show.signif.stars	If TRUE, add the significance stars to the table. The default is taken from <code>getOption("show.signif.stars")</code> .
...	Additional arguments (not used for now)
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate lazyeval expressions (not used for now)

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions.

Examples

```
iris_anova <- anova(lm(data = iris, Petal.Length ~ Species))
tabularise::tabularise$tidy(iris_anova)
```

`tabularise_tidy.aov` *Tidy version of the aov object into a flextable object*

Description

Tidy version of the aov object into a flextable object

Usage

```
## S3 method for class 'aov'
tabularise_tidy(data, ..., kind = "ft", env = parent.frame())
```

Arguments

data	An anova object
...	Additional arguments passed to <code>tabularise_tidy.anova()</code>
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate the object.

Value

flextable object you can print in different form or rearrange with the {flextable} functions.

Examples

```
iris_aov <- aov(data = iris, Petal.Length ~ Species)
tabularise::tabularise$tidy(iris_aov)
```

tabularise_tidy.glm *Create a tidy version of the glm object as a rich-formatted table*

Description

Turn the tidy of **glm** object into a rich-formatted table with {flextable}. The table can be printed in different formats (HTML, LaTeX, Word, PowerPoint), or rearranged later on.

Usage

```
## S3 method for class 'glm'
tabularise_tidy(
  data,
  header = TRUE,
  title = NULL,
  equation = header,
  auto.labs = TRUE,
  origdata = NULL,
  labs = NULL,
  conf.int = FALSE,
  conf.level = 0.95,
  lang = getOption("data.io_lang", "en"),
  show.signif.stars = getOption("show.signif.stars", TRUE),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

<code>data</code>	A glm object
<code>header</code>	If TRUE (by default), add a header to the table
<code>title</code>	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
<code>equation</code>	If TRUE (by default), add an equation to the table header. The equation can also be passed in the form of a character string (LaTeX).
<code>auto.labs</code>	If TRUE (by default), use labels (and units) automatically from <code>origdata</code> .
<code>origdata</code>	The original data set this model was fitted to. By default it is NULL and variables labels from this data set are not used.
<code>labs</code>	Labels to change the names of elements in the term column of the table. By default it is NULL and nothing is changed.
<code>conf.int</code>	If TRUE, add the confidence interval. The default is FALSE.
<code>conf.level</code>	The confidence level to use for the confidence interval if <code>conf.int = TRUE</code> . The default is 0.95.

lang	The natural language to use. The default value can be set with, e.g., <code>options(data.io_lang = "fr")</code> for French.
show.signif.stars	If TRUE, add the significance stars to the table. Its value is obtained from <code>getOption("show.signif.stars")</code> .
...	Additional arguments passed to <code>tabularise::equation()</code>
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object is returned. You can print it in different formats (HTML, LaTeX, Word, Power-Point), or rearrange it with the `{flextable}` functions.

Examples

```
iris_glm <- glm(data = iris, Petal.Length ~ Sepal.Length)
tabularise::tabularise$tidy(iris_glm)
```

tabularise_tidy.lm *Tidy version of the lm object into a flextable object*

Description

Create a rich-formatted table with the 'tidy' information from an **lm** object.

Usage

```
## S3 method for class 'lm'
tabularise_tidy(
  data,
  header = TRUE,
  title = NULL,
  equation = header,
  auto.labs = TRUE,
  origdata = NULL,
  labs = NULL,
  conf.int = FALSE,
  conf.level = 0.95,
  lang = getOption("data.io_lang", "en"),
  show.signif.stars = getOption("show.signif.stars", TRUE),
  ...,
  kind = "ft",
  env = parent.frame()
)
```

Arguments

data	An lm object
header	If TRUE (by default), add an header to the table
title	If TRUE, add a title to the table header. Default to the same value than header, except outside of a chunk where it is FALSE if a table caption is detected (tbl-cap YAML entry).
equation	If TRUE (by default), add an equation to the table header. The equation can also be passed in the form of a character string (LaTeX).
auto.labs	If TRUE (by default), use labels (and units) automatically from data or origdata=.
origdata	The original data set this model was fitted to. By default it is NULL and no label is used.
labs	Labels to change the names of elements in the term column of the table. By default it is NULL and no term is changed.
conf.int	If TRUE, add the confidence interval. The default is FALSE.
conf.level	The confidence level to use for the confidence interval if conf.int = TRUE. The default is 0.95.
lang	The natural language to use. The default value can be set with, e.g., options(data.io_lang = "fr") for French.
show.signif.stars	If TRUE, add the significance stars to the table. The default is getOption("show.signif.stars")
...	Additional arguments passed to <code>tabularise::equation()</code>
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object that you can print in different formats (HTML, LaTeX, Word, PowerPoint) or rearrange with the {flextable} functions.

Examples

```
iris_lm <- lm(data = iris, Petal.Length ~ Sepal.Length)
tabularise::tabularise$tidy(iris_lm)
```

tabularise_tidy.nls *Tidy version of the nls object into a flextable object*

Description

Extract the information contained in a nls object into a rectangular table as it could be obtained by `broom::tidy()`. Here, the table is nicely formatted as an (almost) publication-ready form (good for informal reports, notebooks, etc).

Usage

```
## S3 method for class 'nls'  
tabularise_tidy(data, ..., kind = "ft", env = parent.frame())
```

Arguments

data	A nls object
...	arguments of <code>tabularise_coef.summary.nls()</code>
kind	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
env	The environment where to evaluate lazyeval expressions (unused for now).

Value

A **flextable** object that you can print in different forms or rearrange with the {flextable} functions.

See Also

`tabularise::tabularise()`, `tabularise::tabularise_tidy()`, `tabularise_coef.summary.nls()`

Examples

```
data("ChickWeight", package = "datasets")  
chick1 <- ChickWeight[ChickWeight$Chick == 1, ]  
  
# Adjust a logistic curve  
chick1_logis <- nls(data = chick1, weight ~ SSlogis(Time, Asym, xmid, scal))  
  
tabularise::tabularise$tidy(chick1_logis)  
tabularise::tabularise$tidy(chick1_logis, lang = "fr")
```

Index

`add_predictions (modelr-reexport)`, 13
`add_predictions()`, 13
`add_residuals (modelr-reexport)`, 13
`add_residuals()`, 13
`AIC.model_fit (fit_model)`, 10
`anova.model_fit (fit_model)`, 10
`as.function()`, 3
`as.function.lm`, 3
`as.function.model_fit (fit_model)`, 10
`as.function.nls`, 4
`autoplot.lm (chart.lm)`, 5
`autoplot.nls (chart.nls)`, 7

`BIC.model_fit (fit_model)`, 10
`broom::glance()`, 31
`broom::tidy()`, 36

`chart()`, 3, 12
`chart.lm`, 5
`chart.model_fit (fit_model)`, 10
`chart.nls`, 7
`coef()`, 10
`coef.model_fit (fit_model)`, 10
`confint.model_fit (fit_model)`, 10
`cooks.distance.model_fit (fit_model)`, 10

`deviance.model_fit (fit_model)`, 10

`equation()`, 20, 24, 27, 31
`equation.nls`, 8
`equation.summary.nls (equation.nls)`, 8

`family.model_fit (fit_model)`, 10
`fit_model`, 10
`fit_model()`, 3, 10
`fitted.model_fit (fit_model)`, 10
`formula.model_fit (fit_model)`, 10

`geom_ref_line (modelr-reexport)`, 13
`geom_ref_line()`, 13
`glm()`, 3

`hatvalues.model_fit (fit_model)`, 10

`labels.model_fit (fit_model)`, 10
`lm()`, 3

`mae (modelr-reexport)`, 13
`modelit-package`, 3
`modelr-reexport`, 13

`nls()`, 3
`nobs.model_fit (fit_model)`, 10

`plot.model_fit (fit_model)`, 10
`print()`, 23
`print.summary.nls()`, 27

`qae (modelr-reexport)`, 13
`qae()`, 13

`residuals.model_fit (fit_model)`, 10
`rmse (modelr-reexport)`, 13
`rmse()`, 13
`rsquare (modelr-reexport)`, 13
`rsquare()`, 13
`rstandard.model_fit (fit_model)`, 10

`stats::coef()`, 14–16, 19
`summary()`, 10, 18, 25
`summary.model_fit (fit_model)`, 10

`tabularise()`, 3, 23
`tabularise::equation()`, 29, 30, 35, 36
`tabularise::tabularise()`, 28, 37
`tabularise::tabularise_glance()`, 31
`tabularise::tabularise_tidy()`, 28, 37
`tabularise_coef()`, 27
`tabularise_coef.glm`, 14
`tabularise_coef.glm()`, 22
`tabularise_coef.lm`, 15
`tabularise_coef.lm()`, 23
`tabularise_coef.nls`, 16

tabularise_coef.summary.glm, 18
tabularise_coef.summary.glm(), 25
tabularise_coef.summary.lm, 18
tabularise_coef.summary.lm(), 26
tabularise_coef.summary.nls, 19
tabularise_coef.summary.nls(), 28, 31,
37
tabularise_default.anova, 20
tabularise_default.glm, 22
tabularise_default.lm, 23
tabularise_default.nls, 23
tabularise_default.summary.glm, 25
tabularise_default.summary.lm, 26
tabularise_default.summary.nls, 27
tabularise_glance.glm, 28
tabularise_glance.lm, 29
tabularise_glance.nls, 31
tabularise_tidy.anova, 32
tabularise_tidy.anova(), 33
tabularise_tidy.aov, 33
tabularise_tidy.glm, 34
tabularise_tidy.glm(), 18
tabularise_tidy.lm, 35
tabularise_tidy.lm(), 19
tabularise_tidy.nls, 36

variable.names.model_fit (fit_model), 10
vcov.model_fit (fit_model), 10