Package: modelit (via r-universe)

August 12, 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 *Statistical Models for 'SciViews::R'*

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()`.

as.function.lm *Transform an lm or glm model into a 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

data	A **lm** or **glm** model.
type	The type of plot: "model", "resfitted", "qqplot", "scalelocation", "cooksd", "resleverage", "cookleverage", "reshist" or "resautocor". For chart(), can also be provided as chart$type(....). chart() also uses "residuals" that constructs a combined figure with resfitted, qqplot, scalelocation and resleverage.
...	Additional arguments passed to the chart.“
origdata	The original dataset this model was fitted to. Only required for type = model and in case untransformed X variable is not in the model.
title	A title for the plot. If not provided, a default title is computed.
labels	A vector of four character strings, one for each plot done with chart$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

```
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$scalelocation(trees_lm)
chart$cooksd(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)
```

---

equation.nls *Get a LaTeX equation from an nls or the summary of a nls models*

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

`object`	An **nls** or **summary.nls** object.
`ital_vars`	Logical, defaults to `FALSE`. Should the variable names not be wrapped in the `\operatorname` command?
`use_coefs`	Logical, defaults to `FALSE`. Should the actual model estimates be included in the equation instead of math symbols? If `TRUE`, `var_names=` is ignored.
`coef_digits`	Integer, defaults to 2. The number of decimal places to round to when displaying model estimates with `use_coefs = TRUE`.
`fix_signs`	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)`).
`var_names`	A named character vector as `c(old_var_name = "new name")`
`op_latex`	The LaTeX product operator character to use in fancy scientific notation, either `\\cdot` (by default), or `\\times`.
`...`	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 is. 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 `chart()`)
`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 `parent.frame()` 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 of 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

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

`data`	A **summary.glm** object
`...`	Additional arguments passed to `tabularise_tidy.glm()`
`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_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 `tabularise_tidy.lm()`
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`, `equation()` 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., `options(data.io_lang = "fr")` for French.
`show.signif.stars`	If `TRUE` (by default), add the significance stars to the table.
`...`	Additional arguments passed to `equation()`
`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

`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) automatically (from `origdata=`)
`origdata`	The original data set used for the ANOVA. By default it is `NULL`. Used to extract labels that are lost in the **anova** object.
`labs`	Labels to change the default names 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 is 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 taken from `getOption("show.signif.star`
`...`	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(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

data	A **glm** object
footer	If TRUE (by default), add a footer to the table
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 `tabularise_coef.glm()`
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 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(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 `tabularise_coef.lm()`
`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

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	Add equation of the model to the table. If TRUE, equation() is used. The equation can also be passed in the form of a character string (LaTeX equation).
footer	If TRUE (by default), add a footer to the table.
lang	The language to use. The default value can be set with, e.g., options(data.io_lang = "fr") for French.
...	Additional arguments. Not used.
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(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

`data`	A **summary.glm** object
`footer`	If `TRUE` (by default), add a footer to the table
`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 `tabularise_coef.summary.glm()`
`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 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

`data`	A **summary.lm** object
`footer`	If `TRUE` (by default), add a footer to the table
`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 `tabularise_coef.summary.lm()`
`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(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

`data`	A **summary.nls** 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`	Add equation of the model to the table. If `TRUE`, `equation()` is used. The equation can also be passed in the form of a character string (LaTeX equation).
`footer`	If `TRUE` (by default), add a footer to the table.
`lang`	The language to use. The default value can be set with, e.g. `options(data.io_lang = "fr")` for French.
`show.signif.stars`	If `TRUE` (by default), add the significance stars to the table.
`...`	Additional arguments (Not used).
`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))
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 tabularise::equation()
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 tabularise::equation()
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

`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`	Add equation of the model to the table. If `TRUE`, `equation()` is used. The equation can also be passed in the form of a character string (LaTeX).
`lang`	The language to use. The default value can be set with, e.g., `options(data.io_lang = "fr")` for French.
`...`	Additional arguments passed to `equation()`
`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_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., `options(data.io_lang = "fr")` for French.
show.signif.stars	If `TRUE`, add the significance stars to the table. The default is taken from `getOption("show.signif.star`
...	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 `tabularise_tidy.anova()`
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

`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 variables labels from this 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 nothing 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. Its value is obtained from `getOption("show.signif.star`
...	Additional arguments passed to `tabularise::equation()`
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, PowerPoint), 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 `tabularise::equation()`
`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 tabularise_coef.summary.nls()
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