Package: tabularise (via r-universe)

July 10, 2024

Type Package

Version 0.6.3

Title Create Tabular Outputs from R

Description Create rich-formatted tabular outputs from R that can be incorporated into R Markdown/Quarto documents with correct output at least in HTML, LaTeX/PDF, Word and PowerPoint formats for various R objects.

Maintainer Philippe Grosjean <phgrosjean@sciviews.org>

Depends R (>= 4.2.0)

Imports commonmark (>= 1.9.0), data.io (>= 1.5.0), equatiomatic (>= 0.3.1), flextable (>= 0.9.1), generics (>= 0.1.3), knitr (>= 1.42), officer (>= 0.6.2), rlang (>= 1.1.1), stats (>= 4.2.0), svMisc (>= 1.4.0), tinytable (>= 0.2.1), utils (>= 4.2.0)

Suggests equatags (>= 0.2.0), rmarkdown (>= 2.21), spelling (>= 2.2.1), testthat (>= 3.0.0)

Remotes SciViews/data.io, SciViews/svMisc

License MIT + file LICENSE

URL https://github.com/SciViews/tabularise, https://www.sciviews.org/tabularise/

BugReports https://github.com/SciViews/tabularise/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/tabularise

RemoteRef HEAD

RemoteSha e1e4215c8339b7554d8b1b0cf627f5a59832fced

Contents

`tabularise-package` *Create Tabular Outputs from R*

Description

Create rich-formatted tabular outputs from various R objects that can be incorporated into R Markdown/Quarto documents with correct output at least in HTML, LaTeX/PDF, Word and PowerPoint for various R objects.

Details

For an object class, there is a "default" method that creates the most obvious tabular form for this object, but there might be also other types provided with different tabular views of the same object. All types are accessible from the `tabularise()` function that accepts `type=` argument, or better, by using the compact and easier to read `tabularise$type()` form.

Important functions

- `tabularise()` constructs a table (**flextable** object by default, or **tinytable** object with `kind = "tt"`).
- Stb a collection of functions to manipulate and format a table (mostly functions from the {flextable} package.
- `colformat_sci()` and `colformat_md()` are additional columns formatters for **flextable** objects, respectively, scientific and markdown formats.

- `para_md()` creates {flextable} paragraphs with rich-formatting by converting Markdown strings into such {flextable} paragraphs.

`colformat_md` *Scientific format for columns in {flextable}*

Description

Scientific format for columns in {flextable}

Usage

```
colformat_md(
  x,
  i = NULL,
  j = NULL,
  h.sizes = c(15, 13, 9),
  h.colors = c("black", "darkgray"),
  strike = list(color = h.colors[2], underlined = FALSE),
  bullet = "·",
  code.font = "monospace",
  code.shading = "#f8f8f8",
  link.color = "blue",
  link.underline = TRUE,
  autolink = TRUE
)
```

Arguments

`x`	a **flextable** object
`i`	rows selection
`j`	columns selection
`h.sizes`	font sizes for titles
`h.colors`	font colors for titles
`strike`	a list with color and underlined = `TRUE` or `FALSE` for strikethrough text
`bullet`	bullet character used for lists
`code.font`	font used for code strings
`code.shading`	background color (shading) for code strings
`link.color`	color used for URL links
`link.underline`	are URL links underlined?
`autolink`	are links automatically constructed when there is an URL in the text?

Value

the **flextable** object with the selected region formatted as markdown strings.

Examples

```
dat <- data.frame(
  x = 1:3,
  y = noquote(c("#### text1", "text~2~", "text^3^")),
  names = noquote(c("*Iris setosa*", "Iris ~~virginica~~",
    "Iris **versicolor**")),
  id = c("*setosa*", "`virginica`", "**versicolor**"),
  factor = factor(c("*setosa*", "`virginica`", "**versicolor**"))
)
tabularise(dat)
tabularise(dat) |> colformat_md() |> Stb$autofit()
tabularise(dat) |> colformat_md(i = 2:3, j = 'names') |> Stb$autofit()
```

`colformat_sci` *Scientific format for columns in {flextable}*

Description

Scientific format for columns in {flextable}

Usage

```
colformat_sci(
  x,
  i = NULL,
  j = NULL,
  digits = 3,
  scipen = 0,
  lod = NULL,
  lod_str = paste("<", lod),
  fancy = TRUE,
  op = c("·", "×", "*", "x")
)
```

Arguments

`x`	a **flextable** object
`i`	rows selection
`j`	columns selection
`digits`	number of digits to display
`scipen`	penalty to use to decide if numbers are presented in decimal or scientific notation (generally use 0 or -1)
`lod`	value indicating the limit of detection, for which we should display something like '< lod_value' instead of the actual value; useful for p values (R often uses 2e-16 in that case), chemical measurements ...
`lod_str`	the string to use, by default, it is `< lod_value`.

`fancy`	use a perfect scientific notation (`TRUE`) or a simplified one like 1.34e-5 (`FALSE`).
`op`	the operator character to use in fancy scientific notation

Value

the **flextable** object with the selected region formatted as scientific numbers.

Examples

```
summ <- summary(lm(Volume ~ Girth + Height, data = trees))
tabularise(as.data.frame(summ$coefficients)) |>
  colformat_sci() |>
  colformat_sci(j = 'Pr(>|t|)', lod = 2e-16) |>
  Stb$autofit()
```

`equation` *Get a LaTeX equation from a model or from LaTeX code*

Description

Extract or create a LaTeX equation to describe a model, or directly from LaTeX code. All objects supported by `equatiomatic::extract_eq()` are supported by the default method description.

Usage

```
equation(object, ...)

## Default S3 method:
equation(
  object,
  auto.labs = TRUE,
  origdata = NULL,
  labs = NULL,
  swap_var_names = NULL,
  ...
)

## S3 method for class 'character'
equation(object, ...)

eq__(object, ...)

eq_(object, ...)

## S3 method for class 'inline_equation'
print(x, ...)
```

Arguments

`object`	An object with a model whose the equation is constructed. If a **character** object is provided, it is concatenated into a single character string and the **equation** class, otherwise non transformed (it is supposed to be a valid LaTeX equation). Remember that backslashes must be doubled in regular R strings, or use the `r"(...)"` notation, which may be more comfortable here, see examples.
`...`	Further parameters passed to `equatiomatic::extract_eq()` (see its man page).
`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.
`swap_var_names`	Change the variable names for these values, regardless of the values in `auto.labs=` or `labs=` that are ignored if this argument is used. Provide a named character string with name being the variables and strings the new names. You can use `^` or `_` to indicate next character, or next integer should be super) or subscript in the equation.
`x`	An **inline_equation** object generated by `eq_()`.‘

Details

There are slight differences between `equation()`, `eq_()` and `eq__()`:

- `equation()` returns a string with LaTeX code and prints LaTeX code at the R console.
- `eq_()` returns the LaTeX code surrounded by a dollar sign `$...$` and is suitable to build inline equations in R Markdown/Quarto documents by using inline R code. It prints the rendered inline equation in the RStudio Viewer or in the browser. So it is advised to rapidly preview the resulting equation.
- `eq__()` returns the LaTeX code *not* surrounded by dollar signs in a simple character string. It just prompts the LaTeX string in the R console. It should be used in an R inline chunk inside a `$$...$$` construct in a Markdown text. The result is a display equation that can also be cross referenced in Quarto in the usual way if you label it, e.g., you use `$$...$$ {#eq-label}`.

Value

An object of class `c("equation", "character")`.

Examples

```
iris_lm <- lm(data = iris, Petal.Length ~ Sepal.Length + Species)
summary(iris_lm)
equation(iris_lm)
# Providing directly the LaTeX code of the equation (variance of a sample)
equation("S^2_y = \\sum_{i=1}^{n-1} \\frac{(y_i - \\bar{Y})^2}{n}")
# Easier to write like this (avoiding the double backslashes):
eq1 <- equation(r"(S^2_y = \sum_{i=1}^{n-1} \frac{(y_i - \bar{Y})^2}{n})")
# Print raw text:
```

```
eq1
# Get a preview of the equation
eq__(eq1)
# The same function can be used inside a `$$...$$ {#eq-label}` construct in
# R Markdown or Quarto to calcule a display equation that is also recognized
# by the cross referencing system of Quarto.
# Get a string suitable for inclusion inline in R Markdown with `r eq_(eq1)`
# (inside Markdown text and without the dollar signs around it)
eq_(eq1)
```

para_md *Create a rich-formatted paragraph using markdown notation for flextable objects*

Description

Create a rich-formatted paragraph using markdown notation for flextable objects

Usage

```
para_md(
  ...,
  h.fonts = rep(flextable::get_flextable_defaults()$font.family, 6),
  h.sizes = c(15, 13, 9),
  h.colors = c("black", "darkgray"),
  strike.color = h.colors[2],
  strike.underline = NA,
  code.font = "inconsolata",
  code.shading = "#f8f8f8",
  link.color = "blue",
  link.underline = TRUE,
  bullet = "·",
  autolink = TRUE,
  smart = TRUE,
  debug = FALSE
)

## S3 method for class 'paragraph'
print(x, ...)
```

Arguments

`...`	the character strings with markdown formatting
`h.fonts`	fonts used for the titles (h1-h6)
`h.sizes`	font sizes for titles
`h.colors`	font colors for titles
`strike.color`	color for strikethrough text

`strike.underline`	is strikethrough text converted into underlined text?
`code.font`	font used for code strings
`code.shading`	background color (shading) for code strings
`link.color`	color used for URL links
`link.underline`	are URL links underlined?
`bullet`	bullet character used for lists
`autolink`	are links automatically constructed when there is an URL in the text?
`smart`	is smart punctuation detected and replaced?
`debug`	switch in debug mode
`x`	A **paragraph** object

Value

a flextable paragraph object with its content formatted according to markdown tags

Examples

```
md1 <- paste0("# Heading **1**\n## Heading **2**\n### Heading **3**\n",
  "#### Heading **4**\n##### Heading **5**\n###### Heading **6**")
md2 <- paste0("* List 1\n  1. List 1a\n  2. List 1b\n* List 2\n<br />\n",
  "*Some text* with super^script^, sub~script~ and ~~color~~{+#F50490}")
tabularise(head(iris)) |>
  Stb$add_footer_lines(para_md(md1, md2))
```

`Stb` *Tabularise set of function (mainly from {flextable})*

Description

This set provides all the functions you can use to manipulate `tabularise()` tables. They mostly contain the {flextable} API. You are supposed to use it like `Stb$verb(....)` where `verb` is one of the objects contained in the collection. Use `Stb` to list all objects in the set.

Usage

```
Stb
```

Format

An object of class `list` of length 147.

Value

When printing `Stb` alone, a list of all verbs and other objects provided in the set are returned.

Examples

```
# TODO...
```

`tabularise` *Tabularise an object (arrange or enter in tabular form)*

Description

Tabularise an object (arrange or enter in tabular form)

Usage

```
tabularise(data, ..., type = "default", kind = "ft", env = parent.frame())

tabularize(data, ..., type = "default", kind = "ft", env = parent.frame())
```

Arguments

`data`	An object
`...`	Further arguments (depending on the object class and on `type=`).
`type`	The type of table to produce.
`kind`	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
`env`	The environment where to evaluate formulas (you probably do not need to change the default).

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from set Stb$verb().

See Also

`tabularise_default()`, `tabularise_headtail()`, `tabularise_coef()`, `tabularise_tidy()`, `tabularise_glance()`

Examples

```
tabularise(iris)
```

tabularise_coef *"Coef" type tabularise generic (tabularise$coef)*

Description

The "coef" type of `tabularise()` extracts and formats a table of coefficients from an object, similar to `stats::coef()` applied to the same object, but in a rich-formatted form.

Usage

```
tabularise_coef(data, ..., kind = "ft", env = env)

## Default S3 method:
tabularise_coef(data, ..., kind = "ft", env = env)
```

Arguments

`data`	An object
`...`	Further arguments (depending on the object class).
`kind`	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
`env`	The environment where to evaluate formulas (you probably do not need to change the default).

Details

No useful method for this type is defined in the {tabularise} package, but additional packages might define some.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from set Stb$verb().

See Also

`tabularise()`, `stats::coef()`

`tabularise_confint` *"Confint" type" tabularise generic (tabularise$confint)*

Description

The "confint" type of `tabularise()` presents a table of confidence intervals for an objects (e.g., confidence intervals on parameters of a model). This is similar to the output of `stats::confint()` generic function on the same object. The nicely formatted table obtained here is (almost) publication-ready (good for informal reports, notebooks, etc).

Usage

```
tabularise_confint(data, ..., kind = "ft", env = env)

## Default S3 method:
tabularise_confint(data, ..., kind = "ft", env = env)
```

Arguments

`data`	An object
`...`	Further arguments (depending on the object class).
`kind`	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
`env`	The environment where to evaluate formulas (you probably do not need to change the default).

Details

#' No useful method for this type is defined in the {tabularise} package, but additional packages might define some.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from set Stb$verb().

See Also

`tabularise()`, `stats::confint()`

tabularise_default *Default type tabularise generic (tabularise$default)*

Description

The "default" type is the most obvious tabular representation for an object. For data frames, it tabularises the first few rows and columns (so, in case of a very large object, output remains limited). See also the "headtail" type that creates a table with the few first and last rows of the table (see `tabularise_headtail()`).

Usage

```
tabularise_default(data, ..., kind = "ft", env = parent.frame())

## Default S3 method:
tabularise_default(data, ..., kind = "ft", env = parent.frame())

## S3 method for class 'data.frame'
tabularise_default(
  data,
  formula = NULL,
  col_keys = names(data),
  cwidth = 0.75,
  cheight = 0.25,
  max.rows = 50,
  max.cols = 15,
  auto.labs = TRUE,
  ...,
  env = parent.frame()
)

## S3 method for class 'matrix'
tabularise_default(
  data,
  col_keys = colnames(data),
  rownames = " ",
  cwidth = 0.75,
  cheight = 0.25,
  ...,
  env = parent.frame()
)

## S3 method for class 'Correlation'
tabularise_default(
  data,
  col_keys = colnames(data),
  rownames = " ",
  header = TRUE,
  title = header,
  footer = TRUE,
  cwidth = 0.75,
  cheight = 0.25,
  lang = getOption("data.io_lang", "en"),
  ...,
  env = parent.frame()
)
```

Arguments

`data`	An object
`...`	Further arguments (depending on the object class).
`kind`	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
`env`	The environment where to evaluate formulas (you probably do not need to change the default).
`formula`	A formula to create a table using the {tables} syntax
`col_keys`	The names/keys to use for the table columns
`cwidth`	Initial width for cell sizes in inches
`cheight`	Initial height for cell sizes in inches
`max.rows`	The maximum number of rows to display in the table
`max.cols`	The maximum number of columns to display in the table
`auto.labs`	Are labels automatically used for names of table columns?
`rownames`	col_keys to use for row names. If `FALSE`, do not add row names. If a string, a first column is added in the table with that string as label
`header`	do we add a header?
`title`	do we add a title?
`footer`	do we add a footer?
`lang`	the natural language to use. The default value can be set with, e.g., `options(data.io_lang = "fr")` for French.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from set Stb$verb().

See Also

`tabularise()`, `tabularise_headtail()`

Examples

```
tabularise$default(iris)
# Same as simply:
tabularise(iris)
```

tabularise_glance *"Glance" type" tabularise generic (tabularise$glance)*

Description

The "glance" type of `tabularise()` usually presents a very short (mostly single row) summary of an object in a rich-formatted table. The table presents usually the same or very similar information to what would be obtained with the `generics::glance()` generic function on the same object. The nicely formatted table obtained here is (almost) publication-ready (good for informal reports, notebooks, etc).

Usage

```
tabularise_glance(data, ..., kind = "ft", env = env)

## Default S3 method:
tabularise_glance(data, ..., kind = "ft", env = env)
```

Arguments

`data`	An object
`...`	Further arguments (depending on the object class).
`kind`	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
`env`	The environment where to evaluate formulas (you probably do not need to change the default).

Details

#' No useful method for this type is defined in the {tabularise} package, but additional packages might define some.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from set Stb$verb().

See Also

`tabularise()`, `generics::glance()`

`tabularise_headtail` *Head and tail type tabularise generic (tabularise$headtail)*

Description

The "headtail" type for tabularise presents the first (head) and last (tail) few lines of a table. This is useful for a long table and often more useful than just displaying only the first few lines of the same table (that you got with the default type, see `tabularise_default()`).

Usage

```
tabularise_headtail(data, n = 10, ..., kind = "ft", env = env)

## Default S3 method:
tabularise_headtail(data, n = 10, ..., kind = "ft", env = env)

## S3 method for class 'data.frame'
tabularise_headtail(
  data,
  n = 10,
  auto.labs = TRUE,
  sep = "...",
  ...,
  lang = getOption("data.io_lang", "en"),
  kind = "ft",
  env = env
)
```

Arguments

`data`	An object
`n`	The number of lines to display in the (truncated) table.
`...`	Further arguments (depending on the object class).
`kind`	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
`env`	The environment where to evaluate formulas (you probably do not need to change the default).
`auto.labs`	Are labels automatically used for names of table columns?
`sep`	The separator between the first and last lines of a table. By default, the vertical ellipse shape is used.
`lang`	the natural language to use. The default value can be set with, e.g., `options(data.io_lang = "fr")` for French.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from set Stb$verb().

See Also

`tabularise()`, `head()`, `tail()`, `tabularise_default()`

Examples

```
tabularise$headtail(iris)
```

`tabularise_tidy` *"Tidy" type tabularise generic (tabularise$tidy)*

Description

The "tidy" type of `tabularise()` presents a tidy version of an object, as it could be obtained by `generics::tidy()` whose goal is to turn information contained in an object into a rectangular table. Here, the table is nicely formatted as an (almost) publication-ready form (good for informal reports, notebooks, etc).

Usage

```
tabularise_tidy(data, ..., kind = "ft", env = env)

## Default S3 method:
tabularise_tidy(data, ..., kind = "ft", env = env)
```

Arguments

`data`	An object
`...`	Further arguments (depending on the object class).
`kind`	The kind of table to produce: "tt" for tinytable, or "ft" for flextable (default).
`env`	The environment where to evaluate formulas (you probably do not need to change the default).

Details

No useful method for this type is defined in the {tabularise} package, but additional packages might define some.

Value

A **flextable** object you can print in different form or rearrange with the {flextable} functions from set Stb$verb().

See Also

`tabularise()`, `generics::tidy()`

`theme_tt_sciviews` *Default SciViews theme for tinytables*

Description

This theme is applied by default to {tinytable} output. One can use a different one with `tinytable::theme_tt()`.

Usage

```
theme_tt_sciviews(x, ...)
```

Arguments

`x`	A **tinytable** object
`...`	Additional arguments (not used)

Value

A **tinytable** object with the default SciViews theme applied

Index