Package 'pixiedust'

July 23, 2025

Title Tables so Beautifully Fine-Tuned You Will Believe It's Magic

Version 0.9.4

Description The introduction of the 'broom' package has made converting model objects into data frames as simple as a single function. While the 'broom' package focuses on providing tidy data frames that can be used in advanced analysis, it deliberately stops short of providing functionality for reporting models in publication-ready tables. 'pixiedust' provides this functionality with a programming interface intended to be similar to 'ggplot2's system of layers with fine tuned control over each cell of the table. Options for output include printing to the console and to the common markdown formats (markdown, HTML, and LaTeX). With a little 'pixiedust' (and happy thoughts) tables can really fly.

Depends R (>= 3.1.2)

Imports broom, checkmate (>= 1.8.0), htmltools, knitr, labelVector, magrittr, reshape2, scales

Suggests dplyr, rmarkdown, testthat

License GPL (>= 2)

VignetteBuilder knitr

URL https://github.com/nutterb/pixiedust

BugReports https://github.com/nutterb/pixiedust/issues

RoxygenNote 7.2.3

NeedsCompilation no

Author Benjamin Nutter [aut, cre], David Kretch [ctb]

Maintainer Benjamin Nutter <benjamin.nutter@gmail.com>

Repository CRAN

Date/Publication 2023-10-10 02:20:02 UTC

Contents

`as.data.frame.dust` *Convert* `dust` *Object to Data Frame*

Description

Sprinkles are applied to the `dust` object as if it were being prepared for printing to the console. However, instead of printing, the object is returned as a single data frame.

Usage

```
## S3 method for class 'dust'
as.data.frame(x, ..., sprinkled = TRUE)

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

Arguments

`x`	A `dust` object.
`...`	Arguments to be passed to other methods. Currently unused.
`sprinkled`	Logical. If `TRUE`, the sprinkles attached to the `dust` object are applied before returning the data frame. Sprinkles are applied via the same mechanism that prints to the console, so only sprinkles that are applicable to console output are used. When `FALSE`, `pixiedust` attempts to reconstruct the data frame (or tidied output from `broom::tidy` originally given to `dust`.

Details

In its current state, this can be a fairly inefficient function as the table, if the longtable option is in use, will be built in a `for` loop and bound together using `rbind`. This isn't really intended for large tables, but may be of assistance when there isn't a sprinkle that does what you want to do. (You can at least pull out the object as a data frame and do your own post processing).

Functional Requirements

1. Accepts an object of class `dust` or `dust_list`
2. Accepts a `logical(1)` indicating if the sprinkles should be applied to the data.
3. For a `dust` object, returns an object of class `data.frame`
4. For a `dust_list` object, returns a list of objects of class `data.frame`

Author(s)

Benjamin Nutter

Examples

```
fit <- lm(mpg ~ qsec + factor(am) + wt * factor(gear), data = mtcars)
Dust <- dust(fit) %>%
  sprinkle(cols = 2:4, round = 2) %>%
  sprinkle(cols = 5, fn = quote(pvalString(value))) %>%
  sprinkle(cols = 3, font_color = "#DA70D6") %>%
  sprinkle_print_method("html")

as.data.frame(Dust)
```

`dust` *Dust Table Construction*

Description

Dust tables consist of four primary components that are built together to create a full table. Namely, the `head`, the `body`, the `interfoot`, and the `foot`. Dust tables also contain a table-wide attributes `border_collapse` and `longtable` as well as a `print_method` element.

Usage

```
dust(object, ...)

## Default S3 method:
dust(
  object,
  ...,
  tidy_df = FALSE,
  keep_rownames = FALSE,
  glance_foot = FALSE,
  glance_stats = NULL,
  col_pairs = 2,
  byrow = FALSE,
  descriptors = "term",
  numeric_level = c("term", "term_plain", "label"),
```

```
  label = NULL,
  caption = NULL,
  caption_number = getOption("pixied_caption_number", TRUE),
  justify = getOption("pixie_justify", "center"),
  float = getOption("pixie_float", TRUE),
  longtable = getOption("pixie_longtable", FALSE),
  hhline = getOption("pixie_hhline", FALSE),
  bookdown = getOption("pixie_bookdown", FALSE),
  border_collapse = getOption("pixie_border_collapse", "collapse"),
  tabcolsep = getOption("pixie_tabcolsep", 6),
  fixed_header = getOption("pixie_fixed_header", FALSE),
  html_preserve = getOption("pixie_html_preserve", TRUE)
)

## S3 method for class 'grouped_df'
dust(object, ungroup = TRUE, ...)

## S3 method for class 'list'
dust(object, ...)

redust(x, table, part = c("head", "foot", "interfoot", "body"))

## Default S3 method:
redust(x, table, part = c("head", "foot", "interfoot", "body"))

## S3 method for class 'dust_list'
redust(x, table, part = c("head", "foot", "interfoot", "body"))
```

Arguments

object	An object that has a tidy method in broom
...	Additional arguments to pass to tidy
tidy_df	When object is an object that inherits the data.frame class, the default behavior is to assume that the object itself is the basis of the table. If the summarized table is desired, set to TRUE.
keep_rownames	When tidy_df is FALSE, setting keep_rownames binds the row names to the data frame as the first column, allowing them to be preserved in the tabulated output. This is only to data frame like objects, as the broom::tidy.matrix method performs this already.
glance_foot	Arrange the glance statistics for the foot of the table. (Not scheduled for implementation until version 0.4.0)
glance_stats	A character vector giving the names of the glance statistics to put in the output. When NULL, the default, all of the available statistics are retrieved. In addition to controlling which statistics are printed, this also controls the order in which they are printed.
col_pairs	An integer indicating the number of column-pairings for the glance output. This must be less than half the total number of columns, as each column-pairing in-

cludes a statistic name and value. See the full documentation for the unexported function glance_foot.

byrow — A logical, defaulting to FALSE, that indicates if the requested statistics are placed with priority to rows or columns. See the full documentation for the unexported function glance_foot.

descriptors — A character vector indicating the descriptors to be used in the table. Acceptable inputs are "term", "term_plain", "label", "level", and "level_detail". These may be used in any combination and any order, with the descriptors appearing in the table from left to right in the order given. The default, "term", returns only the term descriptor and is identical to the output provided by broom::tidy methods. See Details for a full explanation of each option and the Examples for sample output. See the full documentation for the unexported function tidy_levels_labels.

numeric_level — A character string that determines which descriptor is used for numeric variables in the "level_detail" descriptor when a numeric has an interaction with a factor. Acceptable inputs are "term", "term_plain", and "label". See the full documentation for the unexported function tidy_levels_labels.

label — character(1). An optional string for assigning labels with which tables can be referenced elsewhere in the document. If NULL, pixiedust attempts to name the label tab:[chunk-name], where [chunk-name] is the name of the knitr chunk. If this also resolves to NULL (for instance, when you aren't using knitr, the label tab:pixie-[n] is assigned, where [n] is the current value of options()$pixie_count. Note that rendering multiple tables in a chunk without specifying a label will result in label conflicts.

caption — A character string giving the caption for the table.

caption_number — logical(1). Should the table caption be prefixed with the table number?

justify — character(1). Specifies the justification of the table on the page. May be "center" (default), "left", or "right".

float — A logical used only in LaTeX output. When TRUE, the table is set within a table environment. The default is TRUE, as with xtable.

longtable — Allows the user to print a table in multiple sections. This is useful when a table has more rows than will fit on a printed page. Acceptable inputs are FALSE, indicating that only one table is printed (default); TRUE that the table should be split into multiple tables with the default number of rows per table (see "Longtable"); or a positive integer indicating how many rows per table to include. All other values are interpreted as FALSE. In LaTeX output, remember that after each section, a page break is forced. This setting may also be set from sprinkle.

hhline — Logical. When FALSE, the default, horizontal LaTeX cell borders are drawn using the \cline command. These don't necessarily play well with cell backgrounds, however. Using hhline = TRUE prints horizontal borders using the \hhline command. While the hhline output isn't disrupted by cell backgrounds, it may require more careful coding of the desired borders. In hhline, cells with adjoining borders tend to double up and look thicker than when using cline.

bookdown — Logical. When TRUE, bookdown style labels are generated. Defaults to FALSE.

border_collapse
: character(1). One of "collapse", "separate", "initial", or "inherit".

tabcolsep
: integerish(1). For LaTeX output, the distance in pt between columns of the table.

fixed_header
: logical(1). For HTML tables, should the header rows be fixed in place over a scrollable body.

html_preserve
: logical(1). When TRUE, HTML output is returned wrapped in htmltools::htmlPreserve. If using LaTeX style equations in an HTML table, it may be necessary to set this to FALSE. Do this at your own risk; this has not been thoroughly field tested.

ungroup
: Used when a grouped_df object is passed to dust. When TRUE (the default), the object is ungrouped and dusted as a single table. When FALSE, the object is split and each element is dusted separately.

x
: A dust object

table
: A data frame of similar dimensions of the part being replaced.

part
: The part of the table to replace with table

Details

The head object describes what each column of the table represents. By default, the head is a single row, but multi row headers may be provided. Note that multirow headers may not render in markdown or console output as intended, though rendering in HTML and LaTeX is fairly reliable. In longtables (tables broken over multiple pages), the head appears at the top of each table portion.

The body object gives the main body of information. In long tables, this section is broken into portions, ideally with one portion per page.

The interfoot object is an optional table to be placed at the bottom of longtable portions with the exception of the last portion. A well designed interfoot can convey to the user that the table continues on the next page.

The foot object is the table that appears at the end of the completed table. For model objects, it is recommended that the glance statistics be used to display model fit statistics.

The border_collapse object applies to an entire HTML table. It indicates if the borders should form a single line or distinct lines.

The longtable object determines how many rows per page are printed. By default, all content is printed as a single table. Using the longtable argument in the sprinkle function can change this setting.

The table_width element is specific to LaTeX tables. This is a reference value for when column widths are specified in terms of the % units. For example, a column width of 20% will be defined as table_width * .20. The value in table_width is assumed to be in inches and defaults to 6.

The tabcolsep object determines the spacing between columns in a LaTeX table in pt. By default, it is set at 6.

The print_method object determines how the table is rendered when the print method is invoked. The default is to print to the console.

Many of these options may be set globally. See pixiedust for a complete list of package options.

Value

Returns an object of class dust

Symbols and Greek Letters

When using markdown, math symbols and greek letters may be employed as they would within a markdown document. For example, "α" will render as the lower case Greek alpha. Math symbols may be rendered in the same manner.

Author(s)

Benjamin Nutter

See Also

tidy glance_foot tidy_levels_labels pixiedust

get_dust_part for extracting parts of the dust object in order to build custom headers and/or footers.

Examples

```
x <- dust(lm(mpg ~ qsec + factor(am), data = mtcars))
x
```

---

fixed_header_css *Generate CSS Code for Fixed Header Tables*

Description

Tables with a fixed header may be generated to permit the headings to remain visible with the data. The CSS is not difficult, but it not-trivial and requires some coordination across a few parts. This functions standardizes the generation of the CSS code using as few elements as possible. Note that there is potential for conflicts with existing CSS in this method.

Usage

```
fixed_header_css(
  fixed_header_class_name = "pixie-fixed",
  scroll_body_height = 300,
  scroll_body_height_units = "px",
  scroll_body_background_color = "white",
  fixed_header_height = 20,
  fixed_header_height_units = "px",
  fixed_header_text_height = fixed_header_height/2,
  fixed_header_text_height_units = "px",
  fixed_header_background_color = "white",
  pretty = TRUE
)
```

Arguments

`fixed_header_class_name`
: `character(1)`. When `include_fixed_header_css = FALSE`, this class name is used to reference CSS classes provided by the user to format the table correctly.

`scroll_body_height`
: `integerish(1)`. Sets the height of the scrollable table body.

`scroll_body_height_units`
: `character(1)`. Determines the units for the height of the scrollable table. Defaults to `"px"`. Must be one of `c("px", "pt", "%", "em")`.

`scroll_body_background_color`
: `character(1)`. The color of the background of the body. Must be a valid color. It defaults to white, which may override CSS settings provided by the user. If this needs to be avoided, you may use the `fixed_header_css` function to assist in generating CSS code to use to define the CSS. See Avoiding CSS Conflicts.

`fixed_header_height`
: `integerish(1)`. Sets the height of the header row.

`fixed_header_height_units`
: `character(1)`. Determines the units for the height of the header row. Defaults to `"px"`. Must be one of `c("px", "pt", "%", "em")`.

`fixed_header_text_height`
: `numeric(1)`. Sets the height at which the header text appears. By default it is set to half of the header height. This should be approximately centered, but you may alter this to get the precise look you want.

`fixed_header_text_height_units`
: `character(1)`. Determines the units for placing the header text. Defaults to `"px"`. Must be one of `c("px", "pt", "%", "em")`.

`fixed_header_background_color`
: `character(1)`. Sets the background color for the header row. This defaults to white and may override the user's CSS settings. See Avoiding CSS Conflicts.

`pretty`
: `logical(1)`. When `TRUE`, the result is printed to the console using `cat`, making it easy to copy and paste the code to another document. When `FALSE`, it is returned as a character string.

Details

CSS doesn't make this kind of table natural. The solution to generate the fixed headers used by `pixiedust` is probably not the best solution in terms of CSS design. It is, however, the most conducive to generating dynamically on the fly.

The fixed header table requires nesting several HTML elements.

1. a `div` tag is used to control the alignment of the table
2. a `section` tag is used to set up the header row that remains fixed.
3. a `div` that sets the height of the scrollable body
4. the `table` tag establishes the actual table.

5. The th tags inside the table are set to full transparency and the content of the headers is duplicated in a div within the th tag to display the content.

To accomplish these tasks, some CSS is exported with the table and placed in the document immediately before the table. Read further to understand the conflicts that may arise if you are using custom CSS specifications in your documents.

Avoiding CSS Conflicts

Because of all of the shenanigans involved, exporting the CSS with the tables may result in conflicts with your custom CSS. Most importantly, any CSS you have applied to the th or td tags may be overwritten. If you are using custom CSS, you may want to consider using include_fixed_header_css = FALSE and then utilizing fixed_header_css to generate CSS you can include in your CSS file to provide the fixed headers. The code generated by fixed_header_css ought to be placed before your definitions for td and th.

To get the same header design in the fixed table, you will want to modify the .th-pixie-fixed div definition in the CSS to match your desired th definition.

The code produced by fixed_header_css will include comments where there is potential for a CSS conflict.

Functional Requirements

1. If pretty = TRUE print results to the console.
2. If pretty = FALSE Return a character string of length 1.
3. Cast an error if scroll_body_height is not integerish(1)
4. Cast an error if scroll_body_height_units is not character(1)
5. Cast an error if scroll_body_background_color is not character(1)
6. Cast an error if scroll_body_background_color is not a valid color.
7. Cast an error if fixed_header_height is not integerish(1)
8. Cast an error if fixed_header_height_units is not character(1)
9. Cast an error if fixed_header_text_height is not numeric(1)
10. Cast an error if fixed_header_text_height_units is not character(1)
11. Cast an error if fixed_header_background_color is not character(1)
12. Cast an error if fixed_header_background_color is not a valid color.
13. Cast an error if pretty is not logical(1)

Source

Jonas Schubert Erlandsson. https://jsfiddle.net/dPixie/byB9d/3/

gaze *Mimic Stargazer Output to Display Multiple Models*

Description

Tidy multiple models and display coefficients and test statistics in a side-by-side format.

Usage

```
gaze(
  ...,
  include_glance = TRUE,
  glance_vars = c("adj.r.squared", "sigma", "AIC"),
  digits = 3
)
```

Arguments

`...`	models to be tidied. Arguments may be named or unnamed. For named arguments, the model will be identfied by the argument name; for unnamed arguments, the object name will be the identifier.
`include_glance`	`logical(1)` Determines if `glance` (fit) statistics are displayed under the models.
`glance_vars`	`character`. A vector of statistics returned by `glance` that are to be displayed for each model. Defaults are subject to change in future versions.
`digits`	`numeric(1)` The number of digits used for rounding.

Details

This function is still in development. Significant stars will be added in a future version. Note that function defaults may be subject to change.

Functional Requirements

1. Return a data frame object
2. Cast an error if `include_glance` is not `logical(1)`
3. Cast an error if `glance_vars` is not a `character` vector.
4. Cast an error if `digits` is not `"integerish(1)"`.

Examples

```
fit1 <- lm(mpg ~ qsec + am + wt + gear + factor(vs), data = mtcars)
fit2 <- lm(mpg ~ am + wt + gear + factor(vs), data = mtcars)

gaze(fit1, fit2)
gaze(with_qsec = fit1,
     without_qsec = fit2)
```

```
gaze(fit1, fit2, include_glance = FALSE)
gaze(fit1, fit2, glance_vars = c("AIC", "BIC"))
```

---

get_dust_part *Get a Portion of the Table Stored in a* dust *Object*

Description

Making customized table headers and footers requires a data frame be added to the dust object that has the same column dimension as the rest of the table. In order to reduce the inconvenience of counting columns, get_dust_part extracts the data frame portion currently in use. This ensures the column dimension is correct with the current values, and provides an object suitable for editing.

Usage

```
get_dust_part(x, part = c("head", "foot", "interfoot", "body"))
```

Arguments

x	An object of class dust
part	character(1), naming the part of the table to retrieve. May be one of "head", "foot", "interfoot", or "body".

Value

an object of class data.frame

Functional Requirements

1. Return, as a data frame, the part of the table requested in part
2. Cast an error if x is not a dust object.
3. Cast an error if part is not one of c("head", "foot", "interfoot", "body")

---

glance_foot *Prepare Glance Statistics for* pixiedust *Table Footer*

Description

Retrieves the broom::glance output for a model object and structures it into a table suitable to be placed in the footer. By default, the statistics are displayed in two column-pairings (see Details). This function is not exported but is documented to maintain clarity of its behavior. It is intended for use within dust, but may be useful elsewhere if used with caution.

Usage

```
glance_foot(fit, col_pairs, total_cols, glance_stats = NULL, byrow = FALSE)
```

Arguments

fit	A model object with a broom::glance method.
col_pairs	An integer indicating the number of column-pairings for the glance output. This must be less than half the total number of columns, as each column-pairing includes a statistic name and value.
total_cols	The total number of columns in the body of the pixiedust table
glance_stats	A character vector giving the names of the glance statistics to put in the output. When NULL, the default, all of the available statistics are retrieved. In addition to controlling which statistics are printed, this also controls the order in which they are printed.
byrow	A logical, defaulting to FALSE, that indicates if the requested statistics are placed with priority to rows or columns. See Details.

Details

Statistics are placed in column-pairings. Each column pair consists of two columns named stat_name_x and stat_value_x, where x is the integer index of the column pair. The column-pairings are used to allow the user to further customize the output, more-so than pasting the name and value together would allow. With this design, statistics can be rounded differently by applying sprinkles to the resulting table.

The total number of column-pairings must be less than or equal to half the number of total columns. This constraint prevents making glance tables that have more columns than the model table it accompanies.

When the total number of column-parings is strictly less than half the total number of columns, "filler" columns are placed between the column pairings. As much as possible, the filler columns are placed evenly between the column pairings, but when the number of filler columns is unequal between column-pairings, there will be more space placed on the left side. For example, if a table has 7 columns and 3 column-pairings, the order of placement would be column-pair-1, filler, column-pair-2, column-pair-3. Since there was only room for one column of filler, it was placed in the left most fill position.

The byrow arguments acts similarly to the byrow argument in the matrix function, but defaults to FALSE. If four statistics are requested and byrow = FALSE, the left column-pair will have statistics one and two, while the right column-pair will have statistics three and four. If byrow = TRUE, however, the left column-pair will have statistics one and three, while the right column-pair will have statistics two and four.

Author(s)

Benjamin Nutter

`index_to_sprinkle` *Determine the Indices to Sprinkle*

Description

The sprinkle methods accept the rows and columns that are to be modified as matrix coordinates. The `dust` object stores the table data in a long form. The tabular coordinates are translated into row indices using this function.

Usage

```
index_to_sprinkle(
  x,
  rows = NULL,
  cols = NULL,
  fixed = FALSE,
  part = c("body", "head", "foot", "interfoot"),
  recycle = c("none", "rows", "cols", "columns"),
  coll = NULL
)
```

Arguments

`x`	An object of class `dust`.
`rows`	Either a numeric vector of rows in the tabular object to be modified or an object of class `call`. When a `call`, generated by `quote(expression)`, the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to `TRUE`.
`cols`	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
`fixed`	`logical(1)` indicating if the values in `rows` and `cols` should be read as fixed coordinate pairs. See Details.
`part`	`character` string. Specifies if the sprinkles are being applied to the head, body, foot, or interfoot of the table. Partial matching is supported.
`recycle`	`character` string. Indicates how recycling is to be performed. Partial matching is supported. See Details.
`coll`	An optional `AssertCollection` object. When `NULL`, an `AssertCollection` object will be created and reported within the call to this function. When not `NULL`, any failed assertions will be added to the object in reported in the function that called `index_to_sprinkle`.

Details

When `fixed = FALSE`, sprinkles are applied at the intersection of `rows` and `cols`, meaning that the arguments do not have to share the same length. When `fixed = TRUE`, they must share the same length.

The value of `recycle` determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right). `"cols"` and `"columns"` have the same effect. The two choices to specify are motivated by the fact that I sometimes get confused about which it should be. :)

Functional Requirements

1. Return the indices of the intersection of `rows` and `cols`
2. If `rows = NULL`, assume all rows.
3. If `rows` is an expression where no values resolve to `TRUE`, return `x` unchanged.
4. If any value in `rows` is not a valid row in the table, cast an error.
5. If `cols = NULL`, assume all columns.
6. If any value in `cols` does not identify a column in the table, cast an error.
7. If `fixed = TRUE`, `length(rows)` (or `sum(rows), if an expression`) and `cols` must have the same length.
8. Cast an error if `fixed` is not a `logical(1)`
9. Cast an error if `part` is not one of `"body"`, `"head"`, `"foot"`, or `"interfoot"`.

Author(s)

Benjamin Nutter

See Also

`sprinkle`

`is_valid_color` *Test a Character String For Pixiedust Recognized Color Format*

Description

`pixiedust` recognizes colors as dvips names, `rgb(R,G,B)`, `rgba(R,G,B,A)`, `#RRGGBB`, or `#RRGGBBAA`. This code returns a logical indicating if the given character strings are valid.

Usage

```
is_valid_color(color)

is_valid_color_single(color)
```

Arguments

color	A character vector of color names.

Functional Requirements

1. Returns a logical vector correctly identifying valid color formats.
2. Casts an error if color is not a character object.

---

knit_print.dust *knitr Printing Function*

Description

Custom printing functions for displaying dust and dust_list objects in R Markdown documents.

Usage

```
## S3 method for class 'dust'
knit_print(x, options, ...)

## S3 method for class 'dust_list'
knit_print(x, options, ...)
```

Arguments

x	A dust object
options	A list of options received from the chunk options.
...	Additional arguments to pass to other methods.

---

medley *Sprinkle Medleys*

Description

pixiedust can get to be pretty verbose if you are doing a great deal of customization. Sprinkle medleys can take out some of that code by bundling much of the formatting sprinkling into a single function.

pixiedust comes with a couple very basic medleys that are mostly for illustration of how to write medleys. Once you get the hang of sprinkling, you need only bundle your most common sprinkles into a medley function of your own and cut down on some of the time coding your most basic formatting.

Usage

```
medley_bw(x)

medley_model(x, round = 2)
```

Arguments

x	a dust object.
round	A numerical value passed to the round sprinkle.

Author(s)

Benjamin Nutter

Examples

```
## Not run:
fit <- lm(mpg ~ qsec + factor(am) + wt * factor(gear), data = mtcars)

dust(fit) %>%
  medley_bw() %>%
  sprinkle_print_method("html")

dust(fit, glance_foot = TRUE) %>%
  medley_model() %>%
  sprinkle_print_method("html")

# Medleys are not generics and do not have methods.
# Using a medley on a dust_list object requires pixieply

library(dplyr)
mtcars %>%
  group_by(gear) %>%
  dust(ungroup = FALSE) %>%
  pixieply(medley_bw) %>%
  sprinkle_print_method("html")

## End(Not run)
```

---

medley_all_borders *Apply Cell Borders to All Cells in a Region*

Description

For most output, specifying a region of cells with borders on all sides is as simple as giving the sprinkle `border = "all"`. In LaTeX output, however, this can result in thicker than expected vertical borders. This medley provides a LaTeX save approach to drawing borders on all sides without getting the double vertical border effect.

Usage

```
medley_all_borders(
  x,
  rows = NULL,
  cols = NULL,
  horizontal = TRUE,
  vertical = TRUE,
  part = "body"
)
```

Arguments

x	An object of class dust
rows	The rows over which the borders are to be drawn.
cols	The cols over which the borders are to be drawn.
horizontal	Logical. Toggles horizontal borders.
vertical	Logical. Toggles vertical borders
part	A character vector. May contain any of "body","head", "interfoot", "foot", "table". When any element is "table", the borders are drawn in all parts of the table.

Author(s)

Benjamin Nutter

---

pixiedust_print_method

Determine the Current Print Method

Description

The user has the option of designating the print method to use, or allowing package to select one from the knitr settings. This function manages the logic of assigning the correct print method within the dust call.

Usage

```
pixiedust_print_method()
```

Details

The function pixiedust_print_method first uses getOption("pixiedust_print_method") to determine if the user has set a print method. If the user has not, it then looks to knitr::opts_knit$get("rmarkdown.pandoc
Finally, if this is also NULL, then the option is set to "console".

pixieply *Apply Functions Over 'dust_list' Objects*

Description

The sprinkle methods work with dust_list objects very naturally, but medleys pose a slightly more difficult problem. Medleys are intended to be predefined collections of sprinkles that reduce the time required to format a table with a particular look and style. It seems counter-productive to expect a user to define each of her or his medleys as a method that can work with both dust and dust_list objects. pixieply is a wrapper to lapply that preserves the dust_list class of the object.

pixiemap provides functionality to apply differing sprinkles over each element of a dust_list. The most common example is probably adding a unique caption to each table.

Usage

```
pixieply(X, FUN, ...)

pixiemap(X, FUN, ..., MoreArgs = NULL, SIMPLIFY = FALSE, USE.NAMES = TRUE)
```

Arguments

X	An object of class dust_list.
FUN	A function to apply to each element of X
...	Additional arguments to pass to FUN
MoreArgs	a list of other arguments to FUN
SIMPLIFY	logical or character string; attempt to reduce the result to a vector, matrix or higher dimensional array; see the simplify argument of sapply
USE.NAMES	logical; use names if the first ... argument has names, or if it is a character vector, use that character vector as the names.

Examples

```
## Not run:
#* This example will only display the last table
#* in the viewer pane.  To see the full output,
#* run this example in an Rmarkdown document.
x <- split(mtcars, list(mtcars$am, mtcars$vs))
dust(x) %>%
  sprinkle_print_method("html") %>%
  pixieply(medley_bw)

## End(Not run)

## Not run:
#* This is the full text of an RMarkdown script
```

```
#* for the previous example.
---
title: "Pixieply"
output: html_document
---

```{r}
library(pixiedust)
x <- dplyr::group_by(mtcars, am, vs)
dust(x, ungroup = FALSE) %>%
  sprinkle_print_method("html") %>%
    pixieply(medley_bw)
```

End(Not run)
```

`pixie_count` *Access and manipulate table numbers counters*

Description

While LaTeX provides the ability to automatically number tables, this functionality is not readily available with console, HTML, or Word output. By keep track of the number of (captioned) tables, we can mimic the behavior of LaTeX tables to provide (mostly) consistent table numbering between formats. The table numbering is stored in the `pixie_count` option.

Usage

```
get_pixie_count()

set_pixie_count(value)

increment_pixie_count(increment = 1)
```

Arguments

`value`	The value at which to set the pixie counter.
`increment`	The value to add to the current pixie count. Defaults to 1.

Details

The pixie count is stored in the options and may also be accessed using `getOption("pixie_count")`.

`get_pixie_count` returns the current value of the counter.

`set_pixie_count` sets the value to the user-specification.

`increment_pixie_count` increments the pixie count, usually by 1. This is called within `print.dust` any time a `dust` object has a caption.

Author(s)

Benjamin Nutter

Source

The concept for these functions is loosely based on a hook meant to work with knitr to automatically number tables. http://stackoverflow.com/a/18672268/1017276

---

print.dust *Print A* dust *Table*

Description

Apply the formatting to a dust object and print the table.

Usage

```
## S3 method for class 'dust'
print(x, ..., asis = TRUE, linebreak_at_end = 2)

## S3 method for class 'dust_list'
print(x, ..., asis = TRUE)
```

Arguments

x	An object of class dust
...	Additional arguments to pass to the print method. Currently ignored.
asis	A logical value that controls if the output is printed using knitr::asis_output. See Details.
linebreak_at_end	Used only in HTML tables; defines the number of line break tags </br> appended to the end of the table in order to generate whitespace between then end of the table and the subsequent element. By default, two line breaks are used.

Details

The printing format is drawn from options()$dustpan_output and may take any of the values "console", "markdown", "html", or "latex"

The markdown, html, and latex output is returned via asis_output, which forces the output into the 'asis' environment. It is intended to work with Rmarkdown, and the tables will be rendered regardless of the chunk's results argument. Currently, there is no way to to capture the code for additional post processing.

When asis = TRUE (the default), the output is returned via knitr::asis_output, which renders the output as if the chunk options included results = 'asis'. Under this setting, the table will be rendered regardless of the value of the results option. Using asis = FALSE returns a character string with the code for the table. This may be rendered in a markdown document via cat(print(x, asis = FALSE)) with the chunk option results = 'asis'. (If working with an Rnw file, the chunk option is results = tex). The only way to use the asis argument is with an explicit call to print.dust.

Author(s)

Benjamin Nutter

Examples

```
dust(lm(mpg ~ qsec + factor(am), data = mtcars))
```

---

pval_string *Format P-values for Reports*

Description

Convert numeric p-values to character strings according to pre-defined formatting parameters. Additional formats may be added for required or desired reporting standards.

Usage

```
pval_string(p, format = c("default", "exact", "scientific"), digits = 3, ...)

pvalString(p, format = c("default", "exact", "scientific"), digits = 3, ...)
```

Arguments

p	a numeric vector of p-values.
format	A character string indicating the desired format for the p-values. See Details for full descriptions.
digits	For "exact" and "scientific"; indicates the number of digits to precede scientific notation.
...	Additional arguments to be passed to format

Details

When format = "default", p-values are formatted:

1. $p > 0.99$: "> 0.99"
2. $0.99 > p > 0.10$: Rounded to two digits
3. $0.10 > p > 0.001$: Rounded to three digits
4. $0.001 > p$: "< 0.001"

When format = "exact", the exact p-value is printed with the number of places after the deimal equal to digits. P-values smaller that 1*(10^-digits) are printed in scientific notation.

When format = "scientific", all values are printed in scientific notation with digits digits printed before the e.

Functional Requirements

1. When `format = "default"`, print p-values greater than 0.99 as "> 0.99"; greater than 0.10 with two digits; greater than 0.001 with three digits; and less than 0.001 as "< 0.001".
2. when `format = "exact"`, print the exact p-value out to at most `digits` places past the decimal place.
3. When `format = "scientific"`, print the p-value in scientific notation with up to `digits` values ahead of the `e`.
4. Cast an error if `p` is not numeric on the interval [0, 1]
5. Cast an error if format is not one of `c("default", "exact", "scientific")`.
6. Cast an error if `digits` is not `integerish(1)`.

Author(s)

Benjamin Nutter

Examples

```
p <- c(1, .999, .905, .505, .205, .125, .09531,
       .05493, .04532, .011234, .0003431, .000000342)
pvalString(p, format="default")
pvalString(p, format="exact", digits=3)
pvalString(p, format="exact", digits=2)
pvalString(p, format="scientific", digits=3)
pvalString(p, format="scientific", digits=4)
```

`rbind_internal` *Bind Rows in Base R*

Description

Stack data frames on top of each other. Data frames do not have to have all of the same columns.

Usage

```
.rbind_internal(..., deparse.level = 1)
```

Arguments

`...`	data frames
`deparse.level`	See `deparse.level` in `rbind`.

Author(s)

Benjamin Nutter

reshape_data_internal *Reshape data frames for Pixiedust*

Description

Pixiedust reshapes data to have one row per cell in the table. This permits adjustments to be made to individual cells. These internal functions are provided to simplify the reshaping process. It is slower than using the tidyr functions ‘gather‘ and ‘spread‘ (or whatever their newer counterparts are), but keeps me off of other people’s development schedules.

Usage

```
.make_dataframe_long(data)
```

Arguments

data	A data.frame

Details

No validations are performed in these functions, and it is assumed that the input data set has the components it needs.

Author(s)

Benjamin Nutter

sanitize_latex *Escape Characters for Printing in LaTeX Output*

Description

sanitize_latex translates particular items in character strings to LaTeX format, e.g., makes a^2 = a\$^2\$ for superscript within variable labels. LaTeX names of greek letters (e.g., "alpha") will have backslashes added if greek==TRUE. Math mode is inserted as needed. sanitize_latex assumes that input text always has matches, e.g. [) [] (] (), and that surrounding by \$\$ is OK.

Usage

```
sanitize_latex(
  object,
  inn = NULL,
  out = NULL,
  pb = FALSE,
  greek = FALSE,
  na = "",
  ...
)
```

Arguments

object	character vector of strings to translate. Any NAs are set to blank strings before conversion.
inn	character vector. Additional strings to translate.
out	character vector the same length as inn. This gives the translated value of the corresonding element in inn
pb	logical(1) If pb=TRUE, sanitize_latex also translates [()] to math mode using \left, \right.
greek	logical(1). set to TRUE to have sanitize_latex put names for greek letters in math mode and add backslashes.
na	character(1) Single character string to translate NA values to.
...	Additional arguments for other methods. Currently ignored.

Value

Vector of chracter strings.

Author(s)

This code is lifted from the Hmisc package in order to avoid depending on that package.

Frank E. Harrell Jr.
Department of Biostatistics,
Vanderbilt University,
f.harrell@vanderbilt.edu

Richard M. Heiberger,
Department of Statistics,
Temple University, Philadelphia, PA.
rmh@temple.edu

David R. Whiting,
School of Clinical Medical Sciences (Diabetes),
University of Newcastle upon Tyne, UK.
david.whiting@ncl.ac.uk

See Also

Hmisc::latexTranslate, Hmisc::sedit

Examples

```
sanitize_latex("75% of the cars were | more than $20,000 Delta = 1.30", greek = TRUE)
```

sprinkle *Define Customizations to a Table*

Description

Customizations to a dust table are added by "sprinkling" with a little extra pixie dust. Sprinkles are a collection of attributes to be applied over a subset of table cells. They may be added to any part of the table, or to the table as a whole.

Usage

```
sprinkle(
  x,
  rows = NULL,
  cols = NULL,
  ...,
  part = c("body", "head", "foot", "interfoot", "table")
)

## Default S3 method:
sprinkle(
  x,
  rows = NULL,
  cols = NULL,
  ...,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns")
)

## S3 method for class 'dust_list'
sprinkle(
  x,
  rows = NULL,
  cols = NULL,
  ...,
  part = c("body", "head", "foot", "interfoot", "table")
)

sprinkle_print_method(
  x,
  print_method = c("console", "markdown", "html", "latex")
)

## Default S3 method:
sprinkle_print_method(
  x,
```

```
  print_method = c("console", "markdown", "html", "latex", "docx")
)

## S3 method for class 'dust_list'
sprinkle_print_method(
  x,
  print_method = c("console", "markdown", "html", "latex")
)

sprinkle_table(x, cols = NULL, ..., part = "table")

## Default S3 method:
sprinkle_table(x, cols = NULL, ..., part = "table")

## S3 method for class 'dust_list'
sprinkle_table(x, cols = NULL, ..., part = "table")
```

Arguments

x	A dust object
rows	A numeric vector specifying the rows of the table to sprinkle. See details for more about sprinkling.
cols	A numeric (or character) vector specifying the columns (or column names) to sprinkle. See details for more about sprinkling.
...	named arguments, each of length 1, defining the customizations for the given cells. See "Sprinkles" for a listing of these arguments.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
print_method	A character string giving the print method for the table. Note: "docx" is synonymous with "markdown".

Details

Sprinkling is done over the intersection of rows and columns (unless fixed = TRUE. If rows but no columns are specified, sprinkling is performed over all columns of the given given rows. The reverse is true for when columns but no rows are specified. If neither columns nor rows are specified, the attribute is applied over all of the cells in the table part denoted in part.

If at least one of border, border_thickness, border_units, border_style or border_color is specified, the remaining unspecified attributes assume their default values.

Other sprinkle pairings are `height` and `height_units`; `width` and `width_units`; `font_size` and `font_size_units`; `bg_pattern` and `bg_pattern_by`

The sprinkles `bg` and `bg_pattern` may not be used together.

A more detailed demonstration of the use of sprinkles is available in `vignette("pixiedust", package = "pixiedust")`

Using `sprinkle_table`, sprinkles may be applied to the columns of multiple tables. Table parts are required to have the same number of columns, but not necessarily the same number of rows, which is why the `rows` argument is not available for the `sprinkle_table`. In contrast to `sprinkle`, the `part` argument in `sprinkle_table` will accept multiple parts. If any of the named parts is `"table"`, the sprinkle will be applied to the columns of all of the parts.

Sprinkles

The following table describes the valid sprinkles that may be defined in the `...` dots argument. All sprinkles may be defined for any output type, but only sprinkles recognized by that output type will be applied when printed. A more readable format of this information is available in `vignette("sprinkles", package = "pixiedust")`.

bg		
	action	Modifies the background color of a cell.
	default	
	accepts	dvips color names; rgb(R,G,B); rgba(R,G,B,A); #RRGGBB; #RRGGBBAA. See the "Colors" section for further details.
	console	Not recognized
	markdown	Not recognized
	html	Accepts any of the listed formats; recognizes transparency
	latex	Accepts any of the listed formats, but ignores transparency
bg_pattern		
	action	Generates a pattern of background colors. Can be used to make striping by rows or by columns.
	default	c("#FFFFFF", "#DDDDDD")
	accepts	A vector of color names: dvips color names; rgb(R,G,B); rgba(R,G,B,A); #RRGGBB; #RRGGBBAA
	console	Not recognized
	markdown	Not recognized
	html	Accepts any of the listed formats; recognizes transparency
	latex	Accepts any of the listed formats, but ignores transparency
bg_pattern_by		
	action	Determines if a ‘bg_pattern‘ is patterned by row or by columns.
	default	"rows"

	accepts	"rows", "columns", "cols"
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
bold		
	action	Renders text within a cell in bold.
	default	FALSE
	accepts	logical(1)
	console	Recognized; rendered as double asterisks on either side of the text
	markdown	Recognized
	html	Recognized
	latex	Recognized
border_collapse		
	action	Sets the ‘border-collapse‘ property in an HTML table. The property sets whether the table borders are collapsed into a single border or detached as in standard HTML.
	default	TRUE
	accepts	logical(1)
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Not recognized
border		
	action	Sets a border on the specified side of a cell.
	default	
	accepts	Any combination of "all", "bottom", "left", "top", "right". Using "all" results in all borders being drawn, regardless of what other values are passed with it.
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
border_color		
	action	Sets the color of the borders specified for a cell.
	default	"Black"
	accepts	character(1) dvips color names; rgb(R,G,B); rgba(R,G,B,A); #RRGGBB; #RRGGBBAA. See the "Colors" section for further details.
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
border_style		

	action	Sets the border style for a specified cell
	default	"solid"
	accepts	character(1) "solid", "dashed", "dotted", "double", "groove", "ridge", "inset", "outset", "hidden", "none"
	console	Not recognized
	markdown	Not recognized
	html	Accepts any of the values listed.
	latex; hhline = FALSE	accepts "solid", "dashed", "dotted", "hidden", "none" "dotted" is silently changed to "dashed" "hidden" and "none" are equivalent.
	latex; hhline = TRUE	accepts "solid", "double", "hidden", "none" "hidden" and "none" are equivalent.
border_thickness		
	action	Sets the thickness of the specified border
	default	1
	accepts	numeric(1)
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
border_units		
	action	Sets the unit of measure for the specified border thickness
	default	"pt"
	accepts	"pt", "px"
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Silently changes "px" to "pt"
caption		
	action	Adds or alters the ‘caption‘ property
	default	
	accepts	character(1)
	console	Recognized
	markdown	Recognized
	html	Recognized
	latex	Recognized
discrete		
	action	Adds distinct background colors based on discrete values in the selected region. May not be used concurrently with bg. `"font"` is an alias for `"font_color"` and `"border"` is an alias for all borders.
	default	"bg"
	accepts	"bg", "font", "font_color", "border",

		"left_border", "top_border", "right_border", "bottom_border"
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
discrete_color		
	action	Sets the color palette from which `discrete` selects background colors. If `NULL` colors are automatically selected using the `scales` package.
	default	`getOption("pixie_discrete_pal", NULL)`
	accepts	character
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
float		
	action	Sets the 'float' property
	default	TRUE
	accepts	logical(1)
	console	Not recognized
	markdown	Not recognized
	html	Not recognized
	latex	Recognized
fn		
	action	Applies a function to the value of a cell. The function should be an expression that acts on the variable 'value'. For example, `quote(format(value, nsmall = 3))`
	default	
	accepts	call
	console	Recognized
	markdown	Recognized
	html	Recognized
	latex	Recognized
font_color		
	action	Sets the color of the cell text
	default	Black
	accepts	dvips color names; rgb(R,G,B); rgba(R,G,B,A); #RRGGBB; #RRGGBBAA. See the "Colors" section for further details.
	console	Not recognized
	markdown	Not recognized
	html	Recognized; transparency recognized
	latex	Recognized; transparency ignored
font_family		
	action	Sets the font for the text

	default	Times New Roman
	accepts	character(1) http://www.w3schools.com/cssref/css_websafe_fonts.asp
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Not recognized
font_size		
	action	Sets the size of the font in the cell
	default	
	accepts	numeric(1)
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
font_size_units		
	action	Determines the units in which ‘font_size‘ is measured
	default	"px"
	accepts	"px", "pt", "%", "em"
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Only recognizes "pt" and "em". All others are coerced to "pt"
gradient		
	action	Adds distinct background colors based on progressively increasing values in the selected region. May not be used concurrently with bg. `"font"` is an alias for `"font_color"` and `"border"` is an alias for all borders.
	default	"bg"
	accepts	"bg", "font", "font_color", "border", "left_border", "top_border", "right_border", "bottom_border"
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
gradient_colors		
	action	Provides the colors between which to shade gradients.
	default	`getOptions("pixie_gradient_pal", NULL)`
	accepts	character
	console	Not recognized
	markdown	Not recognized

	html	Recognized
	latex	Recognized
gradient_cut		
	action	Determines the breaks points for the gradient shading. When `NULL` equally spaced quantiles are used, the number of which are determined by `gradient_n`.
	default	NULL
	accepts	numeric
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
gradient_n		
	action	Determines the number of shades to use between the colors in `gradient_colors`.
	default	10
	accepts	numeric
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
gradient_na		
	action	Sets the color of NA values when gradients are shaded.
	default	grey
	accepts	character(1)
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
halign		
	action	Sets the horizontal alignment of the text in the cell
	default	
	accepts	"left", "center", "right"
	console	Not recognized
	markdown	Recognized; numeric values will auto align to the right if no value given.
	html	Recognized. Does not currently employ auto alignment of numeric values, but this may change.
	latex	Recognized; numeric values will auto align to the right if no value given.
height		
	action	Sets the height of the cell
	default	
	accepts	numeric(1)

	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
height_units		
	action	Determines the units in which ‘height‘ is measured
	default	"pt"
	accepts	"px", "pt", "cm", "in", "%"
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized; "px" is coerced to "pt"
hhline		
	action	Toggles the option for cell border drawing with the ‘hhline‘ LaTeX package
	default	FALSE
	accepts	logical(1)
	console	Not recognized
	markdown	Not recognized
	html	Not recognized
	latex	Recognized. When ‘FALSE‘ double borders are not available. When ‘TRUE‘, colored and dashed borders are not available. This is usually the better option when using colored backgrounds in table cells.
italic		
	action	Renders the text in the cell in italic
	default	FALSE
	accepts	logical(1)
	console	Recognized; rendered as an underscore on either side of the text.
	markdown	Recognized
	html	Recognized
	latex	Recognized
justify		
	action	Justifies the entire table on the page.
	default	"center"
	accepts	character(1)
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognizes "center", but both "left" and "right" are rendered as left justified. This may change if a satisfactory solution is found. Usually, tables are best left centered.
longtable		
	action	Toggles the use of the LaTeX ‘longtable‘ style tables, namely allowing long tables to be broken

		into multiple sections. The table header appears at the top of each section. The table interfoot appears at the bottom of each section, except for the last. The table foot appears at the bottom of the last section. May accept either a logical or a numerical value. If numerical, each section will have the specified number of rows.
	default	FALSE
	accepts	logical(1); numeric(1)
	console	Recognized; when 'TRUE', defaults to 25 rows per section.
	markdown	Recognized; when 'TRUE', defaults to 25 rows per section.
	html	Recognized; when 'TRUE', defaults to 25 rows per section.
	latex	Recognized; when 'TRUE', 'longtable''s own algorithm will determine the number of rows per section. When numeric, breaks are forced at the specified number of rows.
merge		
	action	Merges cells in the specified range into a single cell. In cases where either 'merge_rowval' or 'merge_colval' is specified, they will only be honored if 'merge = TRUE'. You must opt in to this action.
	default	FALSE
	accepts	logical(1)
	console	Recognized
	markdown	Recognized
	html	Recognized
	latex	Recognized
merge_rowval		
	action	Specifies the row value of the merged range to print in the table
	default	minimum row value of the merged range
	accepts	numeric(1)
	console	Recognized
	markdown	Recognized
	html	Recognized
	latex	Recognized
merge_colval		
	action	Specifies the column value of the merged range to print in the table
	default	minimum col value of the merged range
	accepts	numeric(1)

	console	Recognized
	markdown	Recognized
	html	Recognized
	latex	Recognized
na_string		
	action	Designates the character string to use in place of missing values
	default	NA
	accepts	character(1)
	console	Recognized
	markdown	Recognized
	html	Recognized
	latex	Recognized
pad		
	action	Designates the padding to place between cell text and boundaries Measured in pixels.
	default	0
	accepts	numeric(1)
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Not recognized
replace		
	action	Replaces existing cell values with user-specified content. Replacement occurs moving down columns from left to right.
	default	
	accepts	character vector of the same length as the number of cells being replaced.
	console	Recognized
	markdown	Recognized
	html	Recognized
	latex	Recognized
rotate_degree		
	action	Rotates text in cells by the designated angle in degrees
	default	
	accepts	numeric(1)
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
round		
	action	Applies the ‘round‘ function to values in the cell. Skips any character values it encounters.
	default	`getOption("digits")`
	accepts	numeric(1)

	console	Recognized
	markdown	Recognized
	html	Recognized
	latex	Recognized
sanitize		
	action	Sanitizes character values that may cause difficulties for the rendered format.
	default	FALSE
	accepts	logical(1)
	console	Not recognized
	markdown	Not recognized
	html	Not recognized
	latex	Recognized. Sanitization is performed using `latexTranslate`
sanitize_args		
	action	Passes additional arguments to `latexTranslate`
	default	`list()`
	accepts	list. See documentation for `latexTranslate` for details
	console	Not recognized
	markdown	Not recognized
	html	Not recognized
	latex	Recognized
tabcolsep		
	action	Modifies the LaTeX ‘tabcolsep‘ parameter of tables This is similar to ‘pad‘ for HTML tables, but only affects the space between columns. Measured in "pt"
	default	6
	accepts	numeric(1)
	console	Not recognized
	markdown	Not recognized
	html	Not recognized
	latex	Recognized
valign		
	action	Designates the vertical alignment of a cell.
	default	
	accepts	"top", "middle", "bottom"
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized
width		
	action	Sets the width of the cell
	default	
	accepts	numeric(1)
	console	Not recognized
	markdown	Not recognized
	html	Recognized

	latex	Recognized
width_units		
	action	Determines the units in which ‘width‘ is measured
	default	"pt"
	accepts	"px", "pt", "cm", "in", "%"
	console	Not recognized
	markdown	Not recognized
	html	Recognized
	latex	Recognized; "px" is coerced to "pt"

Longtable

The `longtable` feature is named for the LaTeX package used to break very large tables into multiple pages.

When using the `longtable=TRUE` option, the default number of rows per table is 25 for console, HTML, and markdown output. For LaTeX output, the number of rows is determined by the LaTeX `longtable` package's algorithm. The number of rows per table only considers the content in the body of the table. Consideration for the number of rows in the head and foot are the responsibility of the user.

Whenever a table is broken into multiple parts, each part retains the table head. If any `interfoot` is provided, it is appended to the bottom of each section, with the exception of the last section. The last section has the `foot` appended.

Colors

Colors may be declared as any of the color names in `colors()`, as rgb character strings such as `"rgb(rrr,ggg,bbb)"` or as hexadecimal character strings such as `"#rrggbb"`.

Transparency is also recognized by HTML output, and may be indicated in the rgba format `"rgba(rrr,ggg,bbb,aa)"`, where `aa` is a number between 0 and 1, inclusive. Alternative, transparency may be given as `"#rrggbbAA"`, where `AA` is a hexadecimal representation of transparency with "00" being completely transparent and "FF" being completely opaque.

LaTeX output does not recognize transparency and will quietly drop the transparency parameter.

All colors are internally translated into rgb format and are case insensitive.

Required LaTeX Packages

If you will be using the LaTeX output, some sprinkles will require you to include additional LaTeX packages in your document preamble. In `.Rnw` files, additional packages can be included with the `\usepackage{[package]}` syntax. In markdown, additional packages are included using `header-includes:` in the YAML front matter with a line of the format `\usepackage{[package]}` for each package to be used. Sprinkles that require additional packages, and the LaTeX packages required, are listed below:

Sprinkle	LaTeX Package(s)
`font_color`	`\usepackage[dvipsnames]{xcolor}`
`bg, bg_pattern`	`\usepackage[dvipsnames,table]{xcolor}`

border_style	\usepackage{arydshln}
	\usepackage{amssymb}
	\usepackage{hhline}
(with vertical dashed lines)	\usepackage{graphicx}
	\makeatletter
	\newcommand*\vdashline{\rotatebox[origin=c]{90}{\$\dabar@\dabar@\dabar@\$}}
	\makeatother
longtable	\usepackage{longtable}
	(Must be loaded before arydshln)
merge	\usepackage{multirow}
captions for non floats	\usepackage{caption}

Note that hhline is used to make horizontal lines when options(pixiedust_latex_hhline = TRUE) (the package default is FALSE), otherwise the cline command is used.

Use of cline permits colored borders and dashed borders, but borders around cells with background colors are sometimes (often) lost.

Use of hhline preserves borders around cells with background colors and permits double borders, but colored and dashed borders are not available.

In order to ensure all features are available, the recommended code block (accounting for the proper order to load packages) is:

```
header-includes:
- \usepackage{amssymb}
- \usepackage{arydshln}
- \usepackage{caption}
- \usepackage{graphicx}
- \usepackage{hhline}
- \usepackage{longtable}
- \usepackage{multirow}
- \usepackage[dvipsnames,table]{xcolor}
- \makeatletter
- \newcommand*\vdashline{\rotatebox[origin=c]{90}{\$\dabar@\dabar@\dabar@\$}}
- \makeatother
```

Author(s)

Benjamin Nutter

Source

Altering the number of rows in a LaTeX longtable
http://tex.stackexchange.com/questions/19710/how-can-i-set-the-maximum-number-of-rows-in-a-page-for-longtable

Vertical dashed cell borders in LaTeX table
http://www.latex-community.org/forum/viewtopic.php?f=45&t=3149

Colored Cell border
http://tex.stackexchange.com/questions/40666/how-to-change-line-color-in-tabular

See Also

sprinkle_colnames for changing column names in a table.

Examples

```
x <- dust(lm(mpg ~ qsec + factor(am), data = mtcars))
x %>% sprinkle(cols = 2:4, round = 3) %>%
  sprinkle(cols = 5, fn = quote(pvalString(value))) %>%
  sprinkle(rows = 2, bold = TRUE)
```

---

sprinkle_align *Sprinkle Alignment of Table Cells*

Description

The alignment refers to the positioning of the text within a cell. Alignment may be given relative to the left, center, or right of a cell, and the top, middle, or bottom of the cell.

Usage

```
sprinkle_align(
  x,
  rows = NULL,
  cols = NULL,
  halign = NULL,
  valign = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_align(
  x,
  rows = NULL,
  cols = NULL,
  halign = NULL,
  valign = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
```

```
sprinkle_align(
  x,
  rows = NULL,
  cols = NULL,
  halign = NULL,
  valign = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
halign	character One of "left", "center", or "right". Defaults to NULL, for no change to the current value.
valign	character One of "top", "middle", or "bottom". Defaults to NULL, for no change to the current value.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Functional Requirements

1. Correctly reassigns the appropriate elements of halign and valign columns in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if halign is not a character
4. Casts an error if part is not one of "body", "head", "foot", or "interfoot"
5. Casts an error if fixed is not a logical(1)
6. Casts an error if recycle is not one of "none", "rows", or "cols"

7. Casts an error if `valign` is not a `character`
8. Cast an error if `recycle = "none"` and `halign` does not have length 1.
9. Cast an error if `recycle = "none"` and `valign` does not have length 1.
10. Cast an error if `halign` is not one of `c("left", "center", "right")`
11. Cast an error if `valign` is not one of `c("top", "middle", "bottom")`

The functional behavior of the `fixed` and `recycle` arguments is not tested for this function. It is tested and validated in the tests for `index_to_sprinkle`.

See Also

`sprinkle`, `index_to_sprinkle`

`sprinkle_bg` *Sprinkle the Background Color of a Cell*

Description

Background colors may be used to highlight the contents of cells, rows, or columns. Most commonly, backgrounds are used to provide row discrimination; the `sprinkle_bg_pattern` function is better suited to that purpose.

Usage

```
sprinkle_bg(
  x,
  rows = NULL,
  cols = NULL,
  bg = "",
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_bg(
  x,
  rows = NULL,
  cols = NULL,
  bg = "",
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

```
## S3 method for class 'dust_list'
sprinkle_bg(
  x,
  rows = NULL,
  cols = NULL,
  bg = "",
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

sprinkle_background(
  x,
  rows = NULL,
  cols = NULL,
  bg = "",
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
bg	character(1) A character string giving a color for the background of the chosen cells.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Details

Colors may be a dvips color name, or in the rgb(R, G, B), rgba(R, G, B, A), #RRGGBB, or #RRGGBBAA formats.

This sprinkle is ignored in console and markdown outputs. HTML output will accept any of the color formats and recognize transparency. LaTeX output will accept any of the color formats but ignore transparency.

As long as `bg` is required to be a `character(1)`, the `recycle` argument is kind of useless. It is included to maintain consistency with the `index_to_sprinkle` function. Future development may permit a character vector of colors.

Functional Requirements

1. Correctly reassigns the appropriate elements `bg` column in the table part.
2. Casts an error if `x` is not a `dust` object.
3. Casts an error if `bg` is not a `character(1)`
4. Casts an error if `bg` is not a valid color format.
5. Casts an error if `part` is not one of `"body"`, `"head"`, `"foot"`, or `"interfoot"`
6. Casts an error if `fixed` is not a `logical(1)`
7. Casts an error if `recycle` is not one of `"none"`, `"rows"`, or `"cols"`
8. Casts an error if `recycle = "none"` and `bg` does not have length 1.

The functional behavior of the `fixed` and `recycle` arguments is not tested for this function. It is tested and validated in the tests for `index_to_sprinkle`.

Author(s)

Benjamin Nutter

See Also

`sprinkle`, `sprinkle_bg_pattern`, `index_to_sprinkle`

---

sprinkle_bg_pattern *Row and Column Background Striping*

Description

Provides background color striping based on row or column. Striping may be done with any number of colors. The most common use of striping is to provide row discrimination in tables.

Usage

```
sprinkle_bg_pattern(
  x,
  rows = NULL,
  cols = NULL,
  bg_pattern = c("transparent", "#DCDCDC"),
  bg_pattern_by = c("rows", "cols"),
  ...,
  part = c("body", "head", "foot", "interoot", "table")
)

## Default S3 method:
sprinkle_bg_pattern(
  x,
  rows = NULL,
  cols = NULL,
  bg_pattern = c("transparent", "#DCDCDC"),
  bg_pattern_by = c("rows", "cols"),
  ...,
  part = c("body", "head", "foot", "interfoot", "table")
)

## S3 method for class 'dust_list'
sprinkle_bg_pattern(
  x,
  rows = NULL,
  cols = NULL,
  bg_pattern = c("transparent", "#DCDCDC"),
  bg_pattern_by = c("rows", "cols"),
  ...,
  part = c("body", "head", "foot", "interfoot", "table")
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
bg_pattern	A character vector giving the colors to be iterated in the pattern.
bg_pattern_by	A subset of c("rows", "cols"), with partial matching accepted. Only the first value is used, and determines the direction of the pattern.
...	Additional arguments to pass to other methods. Currently ignored.

part A character string denoting which part of the table to modify.

Functional Requirements

1. Correctly reassigns the appropriate elements bg column in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if bg_pattern is not a character vector.
4. Casts an error if any element in bg_pattern is not a valid color name.
5. Casts an error if bg_pattern_by is not a subset of c("rows", "columns") (with partial matching).
6. Casts an error if part is not one of "body", "head", "foot", or "interfoot"

This is a rare sprinkle that doesn't use the fixed and recycle arguments. They are assumed to be FALSE and "none", respectively, in order to pass through index_to_sprinkle.

See Also

sprinkle_bg, sprinkle, index_to_sprinkle

---

sprinkle_bookdown *Change the Bookdown Property in a Dust Table*

Description

Tables built for the bookdown package can be referenced in a manner that is consistent between HTML and LaTeX documents.

Usage

```
sprinkle_bookdown(x, bookdown = getOption("pixie_bookdown", FALSE), ...)

## Default S3 method:
sprinkle_bookdown(x, bookdown = getOption("pixie_bookdown", FALSE), ...)

## S3 method for class 'dust_list'
sprinkle_bookdown(x, bookdown = getOption("pixie_bookdown", FALSE), ...)
```

Arguments

x	An object of class dust
bookdown	logical(1) indicating if the table is being produced in a bookdown document.
...	Additional arguments to pass to other methods. Currently ignored.

Details

bookdown is a package that facilitates the writing of books. One of the advantages of bookdown is the ability to reference tables in a manner similar to LaTeX. The key difference in how pixiedust handles output is the reference specification. See https://bookdown.org/yihui/bookdown/tables.html for details on how bookdown uses labels and references.

Functional Requirements

1. Change the bookdown attribute of the dust object.
2. Cast an error if x is not a dust object.
3. Cast an error if bookdown is not a logical object.
4. Cast an error if bookdown has length greater than 1.

Author(s)

Benjamin Nutter

Source

https://bookdown.org/yihui/bookdown/tables.html

See Also

dust, sprinkle

---

sprinkle_border *Sprinkle Changes to Cell Borders*

Description

Cell borders may be used to give visual structure to a table. Borders may generate distinction between sets of results, groups, or types of output.

Usage

```
sprinkle_border(
  x,
  rows,
  cols,
  border = c("all", "bottom", "left", "top", "right"),
  border_color = "black",
  border_style = "solid",
  border_thickness = 1,
  border_units = c("pt", "px"),
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
```

```
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_border(
  x,
  rows = NULL,
  cols = NULL,
  border = c("all", "bottom", "left", "top", "right"),
  border_color = "black",
  border_style = "solid",
  border_thickness = 1,
  border_units = c("pt", "px"),
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_border(
  x,
  rows = NULL,
  cols = NULL,
  border = c("all", "bottom", "left", "top", "right"),
  border_color = "black",
  border_style = "solid",
  border_thickness = 1,
  border_units = c("pt", "px"),
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
border	One or more of "all", "bottom", "left", "top", or "right". Partial matching is supported. Designates the side of the chosen cells for which borders should

be modified.

`border_color` `character(1)` A character string giving a color for the background of the chosen cells. `NULL` makes no change to the current value.

`border_style` `character(1)` setting the border style for the cell. One of `"solid"`, `"dashed"`, `"dotted"`, `"double"`, `"groove"`, `"ridge"`, `"inset"`, `"outset"`, `"hidden"`, or `"none"`. `NULL` makes no change to the current value.

`border_thickness`
`numeric(1)`. Sets the thickness of the border. `NULL` makes no change to the current value.

`border_units` `character(1)`. Sets the unit of measure for the border thickness. May be either `"pt"`, `"px"`. `NULL` makes no change to the current value.

`part` A character string denoting which part of the table to modify.

`fixed` `logical(1)` indicating if the values in `rows` and `cols` should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of `rows` and `cols`, meaning that the arguments do not have to share the same length. When `fixed = TRUE`, they must share the same length.

`recycle` A `character` one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).

`...` Additional arguments to pass to other methods. Currently ignored.

Details

This sprinkle has no effect on console and markdown output.

HTML output accepts all of the possible values of `border_style`.

For LaTeX output, when `hhline = FALSE`, `"solid"`, `"dashed"`, `"dotted"`, `"hidden"`, and `"none"` are accepted. `"dotted"` will silently be treated as `"dashed"`, and `"hidden"` is the equivalent of `"none"`.

For LaTeX output when `hhline = TRUE`, `"solid"`, `"double"`, `"hidden"`, and `"none"` are accepted. `"hidden"` is the equivalent of `"none"`.

When a value of `border_style` is not recognized by an output format, it is silently ignored.

Functional Requirements

1. Correctly reassigns the `left_border`, `right_border`, `top_border` and `bottom_border` columns in the table part.
2. Casts an error if `x` is not a `dust` object.
3. Casts an error if any element of `border` is not one of `"all"`, `"bottom"`, `"left"`, `"top"`, or `"right"`.
4. Casts an error if `border_color` is not a `character(1)`
5. Casts an error if `border_color` is not a valid color format.
6. Casts an error if `border_style` is not one of `"solid"`, `"dashed"`, `"dotted"`, `"double"`, `"groove"`, `"ridge"`, `"inset"`, `"outset"`, `"hidden"`, `"none"`

7. Casts an error if `border_thickness` is not a `numeric(1)`.
8. Casts an error if `border_units` is not one of `"pt"` or `"px"`.
9. Casts an error if `part` is not one of `"body"`, `"head"`, `"foot"`, or `"interfoot"`
10. Casts an error if `fixed` is not a `logical(1)`
11. Casts an error if `recycle` is not one of `"none"`, `"rows"`, or `"cols"`
12. Cast an error if `recycle = "none"` and `border_color` does not have length 1.
13. Cast an error if `recycle = "none"` and `border_style` does not have length 1.
14. Cast an error if `recycle = "none"` and `border_thickness` does not have length 1.
15. Quietly restrict `border_units` to just the first element if is has length > 1 and `recycle = "none"`.

Author(s)

Benjamin Nutter

See Also

`sprinkle`, `index_to_sprinkle`

`sprinkle_border_collapse` *Change the Border Collapse Property in a Dust Table*

Description

The `border_collapse` property controls the appearance of cell borders in HTML tables. Be default, `pixiedust` collapses the borders so that the adjoining border of two cells appear as a single border.

Usage

```
sprinkle_border_collapse(
  x,
  border_collapse = getOption("pixie_border_collapse", "collapse"),
  ...
)

## Default S3 method:
sprinkle_border_collapse(
  x,
  border_collapse = getOption("pixie_border_collapse", "collapse"),
  ...
)

## S3 method for class 'dust_list'
```

```
sprinkle_border_collapse(
  x,
  border_collapse = getOption("pixie_border_collapse", "collapse"),
  ...
)
```

Arguments

x	An object of class dust
border_collapse	character(1). Defaults to "collapse", and may accept any of "collapse", "separate", "initial", or "inherit".
...	Additional arguments to pass to other methods. Currently ignored.

Details

See https://www.w3schools.com/cssref/pr_border-collapse.asp for details on how each option affects the appearance of a table.

This property has no effect on non-HTML output.

Functional Requirements

1. Change the border_collapse attribute of the dust object.
2. Cast an error if x is not a dust object.
3. Cast an error if border_collapse is not one of "collapse", "separate", "initial", "inherit".

Author(s)

Benjamin Nutter

Source

https://www.w3schools.com/cssref/pr_border-collapse.asp

See Also

dust, sprinkle

`sprinkle_caption` *Change the Caption in a Dust Table*

Description

The table caption is often used as a brief title, but may also be used to provide a longer statement explaining how to interpret the table results.

Usage

```
sprinkle_caption(x, caption, ...)

## Default S3 method:
sprinkle_caption(x, caption, ...)

## S3 method for class 'dust_list'
sprinkle_caption(x, caption, ...)
```

Arguments

`x`	An object of class `dust`
`caption`	`character(1)` giving the new caption for the table.
`...`	Additional arguments to pass to other methods. Currently ignored.

Details

The caption may be set during the initial `dust` call. This method allows for modification afterward, such as in the case of when a `dust` object is loaded from memory and the initial call cannot be accessed.

Functional Requirements

1. Change the `caption` attribute of the `dust` object.
2. Cast an error if `x` is not a `dust` object.
3. Cast an error if `caption` is not a character object.
4. Cast an error if `caption` has length greater than 1.

Author(s)

Benjamin Nutter

See Also

`dust`, `sprinkle`

sprinkle_caption_number

Change the Caption in a Dust Table

Description

The table caption is often used as a brief title, but may also be used to provide a longer statement explaining how to interpret the table results.

Usage

```
sprinkle_caption_number(x, caption_number, ...)

## Default S3 method:
sprinkle_caption_number(
  x,
  caption_number = getOption("pixie_caption_number", TRUE),
  ...
)

## S3 method for class 'dust_list'
sprinkle_caption_number(
  x,
  caption_number = getOption("pixie_caption_number", TRUE),
  ...
)
```

Arguments

x	An object of class dust
caption_number	logical(1) When TRUE, the table caption is prefixed with "Table #". Table numbering is suppressed when FALSE. When numbering is suppressed, the table number counter will not increment.
...	Additional arguments to pass to other methods. Currently ignored.

Details

Table numbering makes it possible to reference tables within a document. In some cases, the numbering is not desired. Suppressing numbering may restrict the ability to make reference to the table.

Functional Requirements

1. Change the caption_number attribute of the dust object.
2. Cast an error if x is not a dust object.
3. Cast an error if caption_number is not a logical object.
4. Cast an error if caption_number has length greater than 1.

Author(s)

Benjamin Nutter

See Also

dust, sprinkle

---

sprinkle_colnames *Column Names for* dust *Tables*

Description

Assigns new column names to a table

Usage

```
sprinkle_colnames(x, ...)

## Default S3 method:
sprinkle_colnames(x, ...)

## S3 method for class 'dust_list'
sprinkle_colnames(x, ...)
```

Arguments

x	A dust object.
...	Column names for the table. See 'Input Formats'

Input Formats

- named arguments Using `dust_colnames(term = "Term", estimate = "Estimate")`, column names may be passed for all or a subset of the columns. The existing column name will be matched against the argument name.
- unnamed arguments Using `dust_colnames("Term", "Estimate", "SE", ...)`, column names may be passed for all of the columns. If the arguments are unnamed, the number of arguments passed must match the number of columns in the table.

When using named arguments (or a named vector), you may not mix named and unnamed elements. In other words, if one element is named, they must all be named. Unnamed elements are assigned to columns in sequential order.

Author(s)

Benjamin Nutter

See Also

sprinkle

Examples

```
x <- dust(lm(mpg ~ qsec + factor(am), data = mtcars))
x
x %>% sprinkle_colnames(term = "Term", statistic = "T")
x %>% sprinkle_colnames("Term", "Estimate", "SE", "T-statistic", "p-value")
## Not run:
# Causes an error due to too few unnamed arguments
x %>% sprinkle_colnames("Term", "Estimate")

## End(Not run)
```

---

sprinkle_discrete *Change Color Features by Discrete Values*

Description

Distinct values within a range will be assigned a color and the designated attribute of the table will be modified accordingly.

Usage

```
sprinkle_discrete(
  x,
  rows = NULL,
  cols = NULL,
  discrete = "bg",
  discrete_colors = getOption("pixie_discrete_pal", NULL),
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_discrete(
  x,
  rows = NULL,
  cols = NULL,
  discrete = "bg",
  discrete_colors = getOption("pixie_discrete_pal", NULL),
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
```

```
  ...
)

## S3 method for class 'dust_list'
sprinkle_discrete(
  x,
  rows = NULL,
  cols = NULL,
  discrete = "bg",
  discrete_colors = getOption("pixie_discrete_pal", NULL),
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
discrete	character. A subset of c("bg", "font","font_color", "border", "left_border", "top_border", "right_border","bottom_border").
discrete_colors	character. Gives the color palette to be used. Each value must be a valid color. Defaults to evenly spaced colors over the color space.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Details

This sprinkle is only recognized by HTML and LaTeX. All of the height_units values are recognized by HTML. For LaTeX, "px" is converted to "pt".

"font" and "font_color" both change the font color.

"border" is a shortcut to specify all borders.

Functional Requirements

1. Correctly reassigns the appropriate elements of the bg, font_color, left_border, top_border, right_border, or bottom_border column in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if discrete is not a subset of c("bg", "font", "font_color", "border", "left_border", "right_border", "top_border", "bottom_border")
4. Casts an error if discrete_colors is not a character value.
5. Casts an error if any value of discrete_colors is not a recognized color value.
6. Casts an error if part is not one of "body", "head", "foot", or "interfoot"
7. Casts an error if fixed is not a logical(1)
8. Casts an error if recycle is not one of "none", "rows", or "cols"

The functional behavior of the fixed and recycle arguments is not tested for this function. It is tested and validated in the tests for index_to_sprinkle.

See Also

sprinkle, index_to_sprinkle

---

sprinkle_fixed_header *Assign a Fixed Header to an HTML Table*

Description

Long tables to be displayed on-screen may benefit by keeping the header fixed in position while scrolling through the body of the table. This allows the user to maintain visual contact between the column name and the data.

Usage

```
sprinkle_fixed_header(
  x,
  fixed_header = TRUE,
  include_fixed_header_css = TRUE,
  fixed_header_class_name = "pixie-fixed",
  scroll_body_height = 300,
  scroll_body_height_units = "px",
  scroll_body_background_color = "white",
  fixed_header_height = 20,
  fixed_header_height_units = "px",
  fixed_header_text_height = fixed_header_height/2,
  fixed_header_text_height_units = "px",
```

```
  fixed_header_background_color = "white",
  ...
)

## Default S3 method:
sprinkle_fixed_header(
  x,
  fixed_header = TRUE,
  include_fixed_header_css = TRUE,
  fixed_header_class_name = "pixie-fixed",
  scroll_body_height = 300,
  scroll_body_height_units = "px",
  scroll_body_background_color = "white",
  fixed_header_height = 20,
  fixed_header_height_units = "px",
  fixed_header_text_height = fixed_header_height/2,
  fixed_header_text_height_units = "px",
  fixed_header_background_color = "white",
  ...
)

## S3 method for class 'dust_list'
sprinkle_fixed_header(
  x,
  fixed_header = TRUE,
  include_fixed_header_css = TRUE,
  fixed_header_class_name = "pixie-fixed",
  scroll_body_height = 300,
  scroll_body_height_units = "px",
  scroll_body_background_color = "white",
  fixed_header_height = 20,
  fixed_header_height_units = "px",
  fixed_header_text_height = fixed_header_height/2,
  fixed_header_text_height_units = "px",
  fixed_header_background_color = "white",
  ...
)
```

Arguments

x	An object of class dust
fixed_header	logical(1). When TRUE, HTML output will produce a table with a fixed header and a scrollable body.
include_fixed_header_css	logical(1). When TRUE, the CSS code to produce the table is inserted directly ahead of the HTML code for the table. When FALSE, the CSS is omitted and assumed to be provided by the user. This may be beneficial if the user has defined CSS styles for their tables. In this case, the user will need to add CSS

classes to their customized CSS to accomodate the fixed headers. See Avoiding CSS Conflicts.

`fixed_header_class_name`
: `character(1)`. When `include_fixed_header_css = FALSE`, this class name is used to reference CSS classes provided by the user to format the table correctly.

`scroll_body_height`
: `integerish(1)`. Sets the height of the scrollable table body.

`scroll_body_height_units`
: `character(1)`. Determines the units for the height of the scrollable table. Defaults to `"px"`. Must be one of `c("px", "pt", "%", "em")`.

`scroll_body_background_color`
: `character(1)`. The color of the background of the body. Must be a valid color. It defaults to white, which may override CSS settings provided by the user. If this needs to be avoided, you may use the `fixed_header_css` function to assist in generating CSS code to use to define the CSS. See Avoiding CSS Conflicts.

`fixed_header_height`
: `integerish(1)`. Sets the height of the header row.

`fixed_header_height_units`
: `character(1)`. Determines the units for the height of the header row. Defaults to `"px"`. Must be one of `c("px", "pt", "%", "em")`.

`fixed_header_text_height`
: `numeric(1)`. Sets the height at which the header text appears. By default it is set to half of the header height. This should be approximately centered, but you may alter this to get the precise look you want.

`fixed_header_text_height_units`
: `character(1)`. Determines the units for placing the header text. Defaults to `"px"`. Must be one of `c("px", "pt", "%", "em")`.

`fixed_header_background_color`
: `character(1)`. Sets the background color for the header row. This defaults to white and may override the user's CSS settings. See Avoiding CSS Conflicts.

`...`
: Arguments to pass to other methods.

Details

CSS doesn't make this kind of table natural. The solution to generate the fixed headers used by `pixiedust` is probably not the best solution in terms of CSS design. It is, however, the most conducive to generating dynamically on the fly.

The fixed header table requires nesting several HTML elements.

1. a `div` tag is used to control the alignment of the table
2. a `section` tag is used to set up the header row that remains fixed.
3. a `div` that sets the height of the scrollable body
4. the `table` tag establishes the actual table.
5. The `th` tags inside the table are set to full transparency and the content of the headers is duplicated in a `div` within the `th` tag to display the content.

To accomplish these tasks, some CSS is exported with the table and placed in the document immediately before the table. Read further to understand the conflicts that may arise if you are using custom CSS specifications in your documents.

Avoiding CSS Conflicts

Because of all of the shenanigans involved, exporting the CSS with the tables may result in conflicts with your custom CSS. Most importantly, any CSS you have applied to the `th` or `td` tags may be overwritten. If you are using custom CSS, you may want to consider using `include_fixed_header_css = FALSE` and then utilizing `fixed_header_css` to generate CSS you can include in your CSS file to provide the fixed headers. The code generated by `fixed_header_css` ought to be placed before your definitions for `td` and `th`.

To get the same header design in the fixed table, you will want to modify the `.th-pixie-fixed div` definition in the CSS to match your desired `th` definition.

The code produced by `fixed_header_css` will include comments where there is potential for a CSS conflict.

Functional Requirements

1. Set the `fixed_header` element of the `dust` object correctly.
2. Set the `include_fixed_header_css` element of the `dust` object correctly.
3. Set the `fixed_header_param` element of the `dust` object correctly.
4. Cast an error if `x` does not inherit class `dust`
5. Cast an error if `scroll_body_height` is not `integerish(1)`
6. Cast an error if `scroll_body_height_units` is not `character(1)`
7. Cast an error if `scroll_body_background_color` is not `character(1)`
8. Cast an error if `scroll_body_background_color` is not a valid color.
9. Cast an error if `fixed_header_height` is not `integerish(1)`
10. Cast an error if `fixed_header_height_units` is not `character(1)`
11. Cast an error if `fixed_header_text_height` is not `numeric(1)`
12. Cast an error if `fixed_header_text_height_units` is not `character(1)`
13. Cast an error if `fixed_header_background_color` is not `character(1)`
14. Cast an error if `fixed_header_background_color` is not a valid color.
15. Cast an error if `include_fixed_header_css` is not `logical(1)`
16. Cast an error if `fixed_header_class_name` is not `character(1)`

sprinkle_float *Change the float Property in a Dust Table*

Description

Alter the floating behavior of tables rendered in LaTeX documents. Floating tables are moved to a position deemed ideal by the typesetter. Setting `float = FALSE` causes the table to be rendered in the position in which it is generated in the code.

Usage

```
sprinkle_float(x, float = getOption("pixie_float", FALSE), ...)

## Default S3 method:
sprinkle_float(x, float = getOption("pixie_float", FALSE), ...)

## S3 method for class 'dust_list'
sprinkle_float(x, float = getOption("pixie_float", FALSE), ...)
```

Arguments

`x`	An object of class `dust`
`float`	`logical(1)` indicating if the table should be placed in a floating environment.
`...`	Additional arguments to pass to other methods. Currently ignored.

Details

See https://en.wikibooks.org/wiki/LaTeX/Floats,_Figures_and_Captions for more about floating environments in LaTeX.

This property has no effect on non-LaTeX output.

Functional Requirements

1. Change the `float` attribute of the `dust` object.
2. Cast an error if `x` is not a `dust` object.
3. Cast an error if `float` is not logical or length 1.

Author(s)

Benjamin Nutter

Source

https://en.wikibooks.org/wiki/LaTeX/Floats,_Figures_and_Captions

See Also

dust, sprinkle

sprinkle_fn *Apply a function to a selection of cells*

Description

The pre-defined sprinkles do not always provide the desired impact on the tables. Applying a function allows for highly customized output without having to pre-process that data frame.

Usage

```
sprinkle_fn(
  x,
  rows = NULL,
  cols = NULL,
  fn = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols"),
  ...
)

## Default S3 method:
sprinkle_fn(
  x,
  rows = NULL,
  cols = NULL,
  fn = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_fn(
  x,
  rows = NULL,
  cols = NULL,
  fn = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust

rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
fn	An object of class call. The function should act on an object value (which is an internal column in the dust object). It is recommend to wrap the function call in quote. For example, quote(pvalString(value)) or quote(format(value, nsmall = 3)).
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Details

dust objects transform tabular objects so that each cell in the table comprises one row in the data frame of cell attributes. The function to be applied needs to act on the value column of that data frame.

Functional Requirements

1. Correctly reassigns the appropriate elements fn column in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if fn is not a call object.
4. Casts an error if part is not one of "body", "head", "foot", or "interfoot"
5. Casts an error if fixed is not a logical(1)
6. Casts an error if recycle is not one of "none", "rows", or "cols"

Author(s)

Benjamin Nutter

sprinkle_font *Sprinkle the Characteristics of Text in a Cell*

Description

Text can be made to stand out (or fade away) by using font features such as bold and italic text, color, size, or different fonts.

Usage

```
sprinkle_font(
  x,
  rows = NULL,
  cols = NULL,
  bold = NULL,
  italic = NULL,
  font_size = NULL,
  font_size_units = NULL,
  font_color = NULL,
  font_family = NULL,
  ...,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = "none"
)

## Default S3 method:
sprinkle_font(
  x,
  rows = NULL,
  cols = NULL,
  bold = NULL,
  italic = NULL,
  font_size = NULL,
  font_size_units = NULL,
  font_color = NULL,
  font_family = NULL,
  ...,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = "none"
)

## S3 method for class 'dust_list'
sprinkle_font(
  x,
  rows = NULL,
```

```
  cols = NULL,
  bold = NULL,
  italic = NULL,
  font_size = NULL,
  font_size_units = NULL,
  font_color = NULL,
  font_family = NULL,
  ...,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = "none"
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
bold	logical(1) indicating if the text in the selected cells should be made bold.
italic	logical(1) indicating if the text in the selected cells should be made italic.
font_size	numeric(1) giving the font size.
font_size_units	character(1) giving the units of the font size. May be any of c("px", "pt", "%", "em"). LaTeX output only recognizes "pt" and "em". For LaTeX output, "px" is quietly changed to "pt" when printing.
font_color	character(1) giving a valid color name for the text.
font_family	character(1) giving the font name for the text. This is only recognized in HTML output.
...	Additional arguments to pass to other methods. Currently ignored.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).

Details

The `bold` and `italic` features are recognized by all formats.

Font size features are recognized by HTML and LaTeX. LaTeX only recognizes the font size unit options of `"pt"` and `"em"`, but will quietly change `"px"` to `"pt"` when printing.

Font color features are recognized by HTML and LaTeX.

Font family is only recognized by HTML.

Functional Requirements

1. Correctly change the `bold` column of the table part for the selected cells.
2. Correctly change the `italic` column of the table part for the selected cells.
3. Correctly change the `font_size` column of the table part for the selected cells.
4. Correctly change the `font_size_units` column of the table part for the selected cells.
5. Correctly chagne the `font_color` column of the table part for the selected cells.
6. Correctly change the `font_family` column of the table part for the selected cells.
7. Cast an error if `x` is not a dust object.
8. Cast an error if `bold` is not `logical(1)`
9. Cast an error if `italic` is not `logical(1)`
10. Cast an error if `font_size` is not `numeric(1)`
11. Cast an error if `font_size_units` is not `character(1)`
12. Cast an error if `font_size_units` is not one of px, pt, em, or
13. Cast an error if `font_color` is not `character(1)`
14. Cast an error if `font_family` is not `character(1)`
15. Cast an error if `part` is not a subset of `c("body", "head", "foot", "interfoot")`
16. Cast an error if `recycle = "none"` and `bold` does not have length 1.
17. Cast an error if `recycle = "none"` and `italic` does not have length 1.
18. Cast an error if `recycle = "none"` and `font_size` does not have length 1.
19. Cast an error if `recycle = "none"` and `font_size_units` does not have length 1.
20. Cast an error if `recycle = "none"` and `font_color` does not have length 1.
21. Cast an error if `recycle = "none"` and `font_family` does not have length 1.

The functional behavior of the `fixed` and `recycle` arguments is not tested for this function. It is tested and validated in the tests for `index_to_sprinkle`.

Author(s)

Benjamin Nutter

See Also

`sprinkle`

sprinkle_gradient *Change Color Features by Binning Numeric Values*

Description

Numeric values within a range of cells are binned and colors assigned to show gradual increases in the numeric value.

Usage

```
sprinkle_gradient(
  x,
  rows = NULL,
  cols = NULL,
  gradient = "bg",
  gradient_colors = getOption("pixie_gradient_pal", NULL),
  gradient_cut = NULL,
  gradient_n = 10,
  gradient_na = "grey",
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_gradient(
  x,
  rows = NULL,
  cols = NULL,
  gradient = "bg",
  gradient_colors = getOption("pixie_gradient_pal", c("#132B43", "#56B1F7")),
  gradient_cut = NULL,
  gradient_n = 10,
  gradient_na = "grey",
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_gradient(
  x,
  rows = NULL,
  cols = NULL,
  gradient = "bg",
```

```
  gradient_colors = getOption("pixie_gradient_pal", c("#132B43", "#56B1F7")),
  gradient_cut = NULL,
  gradient_n = 10,
  gradient_na = "grey",
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
gradient	character. A subset of c("bg", "font","font_color", "border", "left_border", "top_border", "right_border","bottom_border").
gradient_colors	character(2). Gives the colors between which to shared gradients.
gradient_cut	numeric. Determines the breaks points for the gradient shading. When NULL equally spaced quantiles are used, the number of which are determined by gradient_n.
gradient_n	numeric(1). Determines the number of shades to use between the colors in gradient_colors
gradient_na	character(1) A valid color that sets the color of NA values when shading a numeric range.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Details

This sprinkle is only recognized by HTML and LaTeX. All of the height_units values are recognized by HTML. For LaTeX, "px" is converted to "pt".

"font" and "font_color" both change the font color.

"border" is a shortcut to specify all borders.

Functional Requirements

1. Correctly reassigns the appropriate elements of the bg, `font_color`, `left_border`, `top_border`, `right_border`, or `bottom_border` column in the table part.
2. Casts an error if `x` is not a `dust` object.
3. Casts an error if `gradient` is not a subset of `c("bg", "font", "font_color", "border", "left_border", "right_border", "top_border", "bottom_border")`
4. Casts an error if `gradient_colors` is not a `character(2)` value.
5. Casts an error if any value of `gradient_colors` is not a recognized color value.
6. Casts an error if `gradient_cut` is not `numeric`.
7. Casts an error if `gradient_n` is not `numeric(1)`.
8. Casts an error if `gradient_na` is not `character(1)`.
9. Casts an error if `gradient_na` is not a valid color.
10. Casts an error if `part` is not one of `"body"`, `"head"`, `"foot"`, or `"interfoot"`
11. Casts an error if `fixed` is not a `logical(1)`
12. Casts an error if `recycle` is not one of `"none"`, `"rows"`, or `"cols"`

The functional behavior of the `fixed` and `recycle` arguments is not tested for this function. It is tested and validated in the tests for `index_to_sprinkle`.

See Also

`sprinkle`, `index_to_sprinkle`

---

sprinkle_height *Adjust Table Cell Height*

Description

Customize the height of a cell in a table. This may be done to improve the appearance of cells with long text.

Usage

```
sprinkle_height(
  x,
  rows = NULL,
  cols = NULL,
  height = NULL,
  height_units = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

```
## Default S3 method:
sprinkle_height(
  x,
  rows = NULL,
  cols = NULL,
  height = NULL,
  height_units = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_height(
  x,
  rows = NULL,
  cols = NULL,
  height = NULL,
  height_units = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
height	numeric(1). Gives the height of the cell.
height_units	character(1). Gives the units for height. One of c("pt", "px", "cm", "in", "%")
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recy-

cling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).

... Additional arguments to pass to other methods. Currently ignored.

Details

This sprinkle is only recognized by HTML and LaTeX. All of the `height_units` values are recognized by HTML. For LaTeX, `"px"` is converted to `"pt"`.

Functional Requirements

1. Correctly reassigns the appropriate elements of `height` and `height_units` columns in the table part.
2. Casts an error if `x` is not a `dust` object.
3. Casts an error if `height` is not a `numeric(1)`
4. Casts an error if `height_units` is not a `character(1)`
5. Casts an error if `part` is not one of `"body"`, `"head"`, `"foot"`, or `"interfoot"`
6. Casts an error if `fixed` is not a `logical(1)`
7. Casts an error if `recycle` is not one of `"none"`, `"rows"`, or `"cols"`
8. Cast an error if `recycle = "none"` and `height` does not have length 1.
9. When `recycle = "none"`, quietly coerce `height_units` to just the first element given.

The functional behavior of the `fixed` and `recycle` arguments is not tested for this function. It is tested and validated in the tests for `index_to_sprinkle`.

See Also

`sprinkle`, `index_to_sprinkle`

---

sprinkle_hhline *Change the hhline Property in a Dust Table*

Description

The `hhline` property controls the appearance of cell borders in LaTeX tables. There is a known limitation in the LaTeX `colortbl` package where cell borders can be hidden if the cell has a background color. If using both cell borders and background colors, it is recommended that you use the `hhline` property to make cell borders appear as desired.

Usage

```
sprinkle_hhline(x, hhline = getOption("pixie_hhline", FALSE), ...)

## Default S3 method:
sprinkle_hhline(x, hhline = getOption("pixie_hhline", FALSE), ...)

## S3 method for class 'dust_list'
sprinkle_hhline(x, hhline = getOption("pixie_hhline", FALSE), ...)
```

Arguments

x	An object of class dust
hhline	logical(1). When TRUE, the LaTeX hhline package will be used for cell borderes.
...	Additional arguments to pass to other methods. Currently ignored.

Details

When hhline = TRUE, borders will be solid; dashed and dotted borders are unsupported by hhline.

This property has no effect on non-LaTeX output.

Functional Requirements

1. Change the hhline attribute of the dust object.
2. Cast an error if x is not a dust object.
3. Cast an error if hhline is not logical and length 1.

Author(s)

Benjamin Nutter

Source

https://www.ctan.org/pkg/hhline?lang=en

See Also

dust, sprinkle

---

sprinkle_html_preserve

Change the HTML Preserve Property in a Dust Table

Description

By default pixiedust makes use of htmltools::htmlPreserve to prevent certain symbols from rendering in unintended ways based on some not-very-well-understood-by-the-author issues. This property controls whether the preservation is used or not.

Usage

```
sprinkle_html_preserve(
  x,
  html_preserve = getOption("pixie_html_preserve", TRUE),
  ...
)

## Default S3 method:
sprinkle_html_preserve(
  x,
  html_preserve = getOption("pixie_html_preserve", TRUE),
  ...
)

## S3 method for class 'dust_list'
sprinkle_html_preserve(
  x,
  html_preserve = getOption("pixie_html_preserve", TRUE),
  ...
)
```

Arguments

x	An object of class dust
html_preserve	logical(1) indicating if the table is being produced in a htmltools::htmlPreserve environment.
...	Additional arguments to pass to other methods. Currently ignored.

Functional Requirements

1. Change the html_preserve attribute of the dust object.
2. Cast an error if x is not a dust object.
3. Cast an error if html_preserve is not logical(1).

Author(s)

Benjamin Nutter

See Also

dust, sprinkle, htmlPreserve

sprinkle_justify *Change the Caption in a Dust Table*

Description

The justification of the table determines the horizontal placing of the table on the page.

Usage

```
sprinkle_justify(x, justify = getOption("pixie_justify", "center"), ...)

## Default S3 method:
sprinkle_justify(x, justify = getOption("pixie_justify", "center"), ...)

## S3 method for class 'dust_list'
sprinkle_justify(x, justify = getOption("pixie_justify", "center"), ...)
```

Arguments

x	An object of class dust
justify	character string giving the justification of the entire table on the page. May be any one of "center", "left", or "right".
...	Additional arguments to pass to other methods. Currently ignored.

Details

For HTML tables, the values "center", "left", and "right" all justify the table as expected. It is important to note, however, that "left" and "right" will cause subsequent elements to be rendered next to the table, not below it. To render the table with left alignment without this side effect, use "none".

In LaTeX output, both "right" and "left" justify to the left. This may change in the future if I find a resolution. Using "none" also results in left justification.

Functional Requirements

1. Change the justify attribute of the dust object.
2. Cast an error if x is not a dust object.
3. Cast an error if justify is not one of "center", "none", "left", or "right".
4. Ignore capitalization of the justify argument.

Author(s)

Benjamin Nutter

See Also

dust, sprinkle

sprinkle_label *Change the Border Collapse Property in a Dust Table*

Description

The label property is used to make references to a table. Labels may be used in LaTeX documents, or in both LaTeX and HTML documents when using bookdown.

Usage

```
sprinkle_label(x, label = NULL, ...)

## Default S3 method:
sprinkle_label(x, label = NULL, ...)

## S3 method for class 'dust_list'
sprinkle_label(x, label = NULL, ...)
```

Arguments

x	An object of class dust
label	character(1) or NULL for no label.
...	Additional arguments to pass to other methods. Currently ignored.

Details

For details about using labels in LaTeX documents, see https://en.wikibooks.org/wiki/LaTeX/Labels_and_Cross-referencing.

For details about using labels in bookdown documents, see https://bookdown.org/yihui/bookdown/tables.html

Functional Requirements

1. Change the label attribute of the dust object.
2. Cast an error if x is not a dust object.
3. Cast an error if label is not a character(1).

Author(s)

Benjamin Nutter

Source

https://en.wikibooks.org/wiki/LaTeX/Labels_and_Cross-referencing

https://bookdown.org/yihui/bookdown/tables.html

See Also

dust, sprinkle

---

sprinkle_longtable *Change the Longtable Property in a Dust Table*

Description

The LaTeX longtable package allows for long tables to be broken into multiple parts to be displayed on separate pages. pixiedust will mimic this behavior for other output types.

Usage

```
sprinkle_longtable(x, longtable = getOption("pixie_longtable", FALSE), ...)

## Default S3 method:
sprinkle_longtable(x, longtable = getOption("pixie_longtable", FALSE), ...)

## S3 method for class 'dust_list'
sprinkle_longtable(x, longtable = getOption("pixie_longtable", FALSE), ...)
```

Arguments

x	An object of class dust
longtable	Either a logical(1) or an numeric(1) integer-like value. See Details.
...	Additional arguments to pass to other methods. Currently ignored.

Details

When longtable = TRUE, LaTeX tables will be divided according to the LaTeX document settings. In other table outputs, the default is to use 25 rows per table.

When longtable is an integer (or integer-like) value, the table is divided into that many rows per section. This applies to all output.

Functional Requirements

1. Change the longtable attribute of the dust object.
2. Cast an error if x is not a dust object.
3. Cast an error if longtable is logical and has length not equal to 1.
4. when longtable is not logical, cast an error if it is not-integerish and has length not equal to 1.

Author(s)

Benjamin Nutter

See Also

dust, sprinkle

---

sprinkle_merge *Sprinkle Table Cells to Merge*

Description

Merging cells creates more space for values to be displayed without disrupting the appearance of other cells in the same row or column. The downside is that the content from only one of the cells in the merge range will be displayed.

Usage

```
sprinkle_merge(
  x,
  rows = NULL,
  cols = NULL,
  merge = FALSE,
  merge_rowval = NULL,
  merge_colval = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_merge(
  x,
  rows = NULL,
  cols = NULL,
  merge = FALSE,
  merge_rowval = NULL,
  merge_colval = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_merge(
  x,
  rows = NULL,
  cols = NULL,
```

```
  merge = FALSE,
  merge_rowval = NULL,
  merge_colval = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
merge	logical Defaults to FALSE, prompting no merging action.
merge_rowval	The row position of the cell whose content will be displayed. Defaults to the minimum of rows.
merge_colval	The column position of the cell whose content will be displayed. Deafults to the minimum of cols.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Functional Requirements

1. Correctly reassigns the appropriate elements of merge, merge_rowval and merge_colval columns in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if merge is not a logical(1)
4. Casts an error if merge_rowval is not a numeric(1)
5. Casts an error if merge_colval is not a numeric(1)
6. Casts an error if part is not one of "body", "head", "foot", or "interfoot"

7. Casts an error if `fixed` is not a `logical(1)`
8. Casts an error if `recycle` is not one of `"none"`, `"rows"`, or `"cols"`

The functional behavior of the `fixed` and `recycle` arguments is not tested for this function. It is tested and validated in the tests for `index_to_sprinkle`.

See Also

`sprinkle`, `index_to_sprinkle`

`sprinkle_na_string` *Sprinkle Appearance of NA's*

Description

The appearance of `NA` values in a table may be dependent on the context. `pixiedust` uses the `na_string` sprinkle to guide the appearance of missing values in the table.

Usage

```
sprinkle_na_string(
  x,
  rows = NULL,
  cols = NULL,
  na_string = getOption("pixie_na_string", NA),
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_na_string(
  x,
  rows = NULL,
  cols = NULL,
  na_string = getOption("pixie_na_string", NA),
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_na_string(
  x,
  rows = NULL,
```

```
  cols = NULL,
  na_string = getOption("pixie_na_string", NA),
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
na_string	character(1) A character string giving desired replacement for NA values in the selected cells.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Functional Requirements

1. Correctly reassigns the appropriate elements na_string column in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if bg is not a character(1)
4. Casts an error if part is not one of "body", "head", "foot", or "interfoot"
5. Casts an error if fixed is not a logical(1)
6. Casts an error if recycle is not one of "none", "rows", or "cols"
7. Cast an error if recycle = "none" and na_string does not have length 1.

The functional behavior of the fixed and recycle arguments is not tested for this function. It is tested and validated in the tests for index_to_sprinkle.

See Also

sprinkle, index_to_sprinkle

sprinkle_pad *Sprinkle the Padding of a Cell*

Description

Padding for HTML tables indicates how many pixels should be placed between the cell's content and the outside border.

Usage

```
sprinkle_pad(
  x,
  rows = NULL,
  cols = NULL,
  pad = 0,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_pad(
  x,
  rows = NULL,
  cols = NULL,
  pad = 0,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_pad(
  x,
  rows = NULL,
  cols = NULL,
  pad = 0,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust

rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
pad	numeric(1) A character string giving a color for the background of the chosen cells.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Details

Colors may be a dvips color name, or in the rgb(R, G, B), rgba(R, G, B, A), #RRGGBB, or #RRGGBBAA formats.

This sprinkle is ignored in console and markdown outputs. HTML output will accept any of the color formats and recognize transparency. LaTeX output will accept any of the color formats but ignore transparency.

As long as pad is required to be a numeric(1), the recycle argument is kind of useless. It is included to maintain consistency with the index_to_sprinkle function. Future development may permit a character vector of colors.

Functional Requirements

1. Correctly reassigns the appropriate elements pad column in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if pad is not a numeric(1)
4. Casts an error if part is not one of "body", "head", "foot", or "interfoot"
5. Casts an error if fixed is not a logical(1)
6. Casts an error if recycle is not one of "none", "rows", or "cols"
7. Cast an error if recycle = "none" and pad does not have length 1.

The functional behavior of the fixed and recycle arguments is not tested for this function. It is tested and validated in the tests for index_to_sprinkle.

Author(s)

Benjamin Nutter

See Also

sprinkle, index_to_sprinkle

---

sprinkle_replace *Replace Contents of Selected Cells*

Description

At times it may be necessary to replace the contents of a cell with user-supplied values.

Usage

```
sprinkle_replace(
  x,
  rows = NULL,
  cols = NULL,
  replace,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_replace(
  x,
  rows = NULL,
  cols = NULL,
  replace,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_replace(
  x,
  rows = NULL,
  cols = NULL,
  replace,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
replace	character A character vector giving the desired content for the selected cells.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Functional Requirements

1. Correctly reassigns the appropriate elements replace column in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if replace is not a vector
4. Casts an warning if the number of indices to replace is not a multiple of replace
5. Casts an error if length(replace) is greater than the number of cells to replace.
6. Casts an error if part is not one of "body", "head", "foot", or "interfoot"
7. Casts an error if fixed is not a logical(1)
8. Casts an error if recycle is not one of "none", "rows", or "cols"

The functional behavior of the fixed and recycle arguments is not tested for this function. It is tested and validated in the tests for index_to_sprinkle.

See Also

sprinkle, index_to_sprinkle

sprinkle_rotate_degree *Sprinkle Appearance of NA's*

Description

The content of cells may be rotated when it is desired to save space (such as long table column names), or to draw attention to the cells.

Usage

```
sprinkle_rotate_degree(
  x,
  rows = NULL,
  cols = NULL,
  rotate_degree = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_rotate_degree(
  x,
  rows = NULL,
  cols = NULL,
  rotate_degree = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_rotate_degree(
  x,
  rows = NULL,
  cols = NULL,
  rotate_degree = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
rotate_degree	numeric(1) Indicates how much to rotate the cell text in degrees.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Functional Requirements

1. Correctly reassigns the appropriate elements rotate_degree column in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if rotate_degree is not a numeric(1)
4. Casts an error if part is not one of "body", "head", "foot", or "interfoot"
5. Casts an error if fixed is not a logical(1)
6. Casts an error if recycle is not one of "none", "rows", or "cols"
7. Cast an error if recycle = "none" and rotate_degree does not have length 1.

The functional behavior of the fixed and recycle arguments is not tested for this function. It is tested and validated in the tests for index_to_sprinkle.

See Also

sprinkle, index_to_sprinkle

sprinkle_round *Sprinkle Appearance of NA's*

Description

The appearance of NA values in a table may be dependent on the context. pixiedust uses the round sprinkle to guide the appearance of missing values in the table.

Usage

```
sprinkle_round(
  x,
  rows = NULL,
  cols = NULL,
  round = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_round(
  x,
  rows = NULL,
  cols = NULL,
  round = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_round(
  x,
  rows = NULL,
  cols = NULL,
  round = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust

rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
round	numeric(1) A value to pass to the digits argument of round.
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Functional Requirements

1. Correctly reassigns the appropriate elements round column in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if round is not a numeric(1)
4. Casts an error if part is not one of "body", "head", "foot", or "interfoot"
5. Casts an error if fixed is not a logical(1)
6. Casts an error if recycle is not one of "none", "rows", or "cols"
7. Cast an error if recycle = "none" and round does not have length 1.

The functional behavior of the fixed and recycle arguments is not tested for this function. It is tested and validated in the tests for index_to_sprinkle.

See Also

sprinkle, index_to_sprinkle

---

sprinkle_sanitize *Sanitize Characters for LaTeX Outputs*

Description

Certain characters in LaTeX code need to be escaped to prevent errors during processing. For example, % is the comment character in LaTeX, and needs to be escaped in order to render correctly.

Usage

```
sprinkle_sanitize(
  x,
  rows = NULL,
  cols = NULL,
  sanitize = NULL,
  sanitize_args = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_sanitize(
  x,
  rows = NULL,
  cols = NULL,
  sanitize = NULL,
  sanitize_args = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_sanitize(
  x,
  rows = NULL,
  cols = NULL,
  sanitize = NULL,
  sanitize_args = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices

	is permissible.
sanitize	logical(1). Should the code for the cell be sanitized.
sanitize_args	A list of arguments to pass to Hmisc::latexTranslate
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Details

This sprinkle is only recognized by LaTeX output. See latexTranslate for more details.

Functional Requirements

1. Correctly reassigns the appropriate elements of sanitize and sanitize_args columns in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if sanitize is not a logical(1)
4. Casts an error if sanitize_args is not a list
5. Casts an error if part is not one of "body", "head", "foot", or "interfoot"
6. Casts an error if fixed is not a logical(1)
7. Casts an error if recycle is not one of "none", "rows", or "cols"

The functional behavior of the fixed and recycle arguments is not tested for this function. It is tested and validated in the tests for index_to_sprinkle.

See Also

sprinkle, index_to_sprinkle

---

sprinkle_tabcolsep *Change the tabcolsep Property in a Dust Table*

Description

The tabcolsep property controls the space between columns in LaTeX output. By default, it is set to 6 pt.

Usage

```
sprinkle_tabcolsep(x, tabcolsep = getOption("pixie_tabcolsep", 6), ...)

## Default S3 method:
sprinkle_tabcolsep(x, tabcolsep = getOption("pixie_tabcolsep", 6), ...)

## S3 method for class 'dust_list'
sprinkle_tabcolsep(x, tabcolsep = getOption("pixie_tabcolsep", 6), ...)
```

Arguments

x	An object of class dust
tabcolsep	numeric(1), integer-like value.
...	Additional arguments to pass to other methods. Currently ignored.

Details

Reading on the details of tabcolsep may be done by searching "latex tabcolsep" on the internet.

This property has no effect on non-LaTeX output.

Functional Requirements

1. Change the tabcolsep attribute of the dust object.
2. Cast an error if x is not a dust object.
3. Cast an error if tabcolsep is not integerish and length 1.

Author(s)

Benjamin Nutter

Source

https://www.google.com/webhp?sourceid=chrome-instant&rlz=1C1CHBF_enUS706US706&ion=1&espv=2&ie=UTF-8#q=latex+tabcolsep&*

See Also

dust, sprinkle

sprinkle_width *Adjust Table Cell Width*

Description

Customize the width of a cell in a table. This may be done to improve the appearance of cells with long text.

Usage

```
sprinkle_width(
  x,
  rows = NULL,
  cols = NULL,
  width = NULL,
  width_units = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## Default S3 method:
sprinkle_width(
  x,
  rows = NULL,
  cols = NULL,
  width = NULL,
  width_units = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)

## S3 method for class 'dust_list'
sprinkle_width(
  x,
  rows = NULL,
  cols = NULL,
  width = NULL,
  width_units = NULL,
  part = c("body", "head", "foot", "interfoot", "table"),
  fixed = FALSE,
  recycle = c("none", "rows", "cols", "columns"),
  ...
)
```

Arguments

x	An object of class dust
rows	Either a numeric vector of rows in the tabular object to be modified or an object of class call. When a call, generated by quote(expression), the expression resolves to a logical vector the same length as the number of rows in the table. Sprinkles are applied to where the expression resolves to TRUE.
cols	Either a numeric vector of columns in the tabular object to be modified, or a character vector of column names. A mixture of character and numeric indices is permissible.
width	numeric(1). Gives the width of the cell.
width_units	character(1). Gives the units for width. One of c("pt", "px", "cm", "in", "%")
part	A character string denoting which part of the table to modify.
fixed	logical(1) indicating if the values in rows and cols should be read as fixed coordinate pairs. By default, sprinkles are applied at the intersection of rows and cols, meaning that the arguments do not have to share the same length. When fixed = TRUE, they must share the same length.
recycle	A character one that determines how sprinkles are managed when the sprinkle input doesn't match the length of the region to be sprinkled. By default, recycling is turned off. Recycling may be performed across rows first (left to right, top to bottom), or down columns first (top to bottom, left to right).
...	Additional arguments to pass to other methods. Currently ignored.

Details

This sprinkle is only recognized by HTML and LaTeX. All of the width_units values are recognized by HTML. For LaTeX, "px" is converted to "pt".

Functional Requirements

1. Correctly reassigns the appropriate elements of width and width_units columns in the table part.
2. Casts an error if x is not a dust object.
3. Casts an error if width is not numeric
4. Casts an error if width_units is not one of c("px", "pt", "in", "cm", "%").
5. Casts an error if part is not one of "body", "head", "foot", or "interfoot"
6. Casts an error if fixed is not a logical(1)
7. Casts an error if recycle is not one of "none", "rows", or "cols"
8. Casts an error if recycle = "none" and width does not have length 1.
9. Correctly assigns values when recycle is not "none" and multiple values are given.
10. Quietly accepts only the first value in width_units when recycle = "none".

The functional behavior of the fixed and recycle arguments is not tested for this function. It is tested and validated in the tests for index_to_sprinkle.

See Also

sprinkle, index_to_sprinkle

`str_extract_base` *Extract Patterns from Character Strings*

Description

This is a utility function that follow the pattern of `stringr::str_extract_all`. It is provided to avoid the dependency on the `stringr` package.

Usage

```
str_extract_base(x, pattern)

str_split_fixed_base(x, pattern, n)
```

Arguments

`x`	`character` vector.
`pattern`	`character(1)` of the pattern to find in `x`
`n`	The number of splits.

Source

https://stackoverflow.com/a/27274231/1017276

See Also

`stringr::str_extract_all`

`tidy_levels_labels` *Term and Level Descriptions for* `pixiedust` *Tables*

Description

Default model objects identify rows of results with appropriate term name. More often than not, the term name is not suitable for formally reported output. `tidy_levels_labels` performs some basic work to quickly provide more readable descriptors for cases where they can easily be obtained. These descriptors are retrieved from the data, however, so the utility is determined by the user's habits in providing term labels and meaningful factor levels.

Due to the complexity of the terms that could be used for a model, it isn't practical to attempt to recover human-ready descriptors for every conceivable term. This would require recovering variable names for any number of functions. `pixiedust` only goes after the easiest to obtain. Replacements no managed by `tidy_levels_labels` may still be made with the `replace` sprinkle.

Usage

```
tidy_levels_labels(
  object,
  descriptors = "term",
  numeric_level = c("term", "term_plain", "label"),
  argcheck = NULL
)
```

Arguments

object	A model object, ideally with a model.frame method. It is unclear at the moment (18 Sept. 2015) what will happen if an object is passed that does not have a model.frame method.
descriptors	A character vector indicating the descriptors to be used in the table. Acceptable inputs are "term", "term_plain", "label", "level", and "level_detail". These may be used in any combination and any order, with the descriptors appearing in the table from left to right in the order given. The default, "term", returns only the term descriptor and is identical to the output provided by broom::tidy methods. See Details for a full explanation of each option and the Examples for sample output.
numeric_level	A character string that determines which descriptor is used for numeric variables in the "level_detail" descriptor when a numeric has an interaction with a factor. Acceptable inputs are "term", "term_plain", and "label".
argcheck	An assert collection created by checkmate::makeAssertCollection. Under normal circumstances, this is passed from dust. If NULL, as in the case it is run outside of dust, a new collection is created and the assertions are reported within tidy_levels_labels.

Details

The user may select up to five columns of descriptors, although doing so would certainly create some ambiguity. See the Examples for sample output.

- "term" The term name used in the R model summary
- "term_plain" The term name used in the formula. For variables that produce multiple term names (such as factors), the plain term name may be duplicated. For example, a factor that has term names FctrB and FctrC, indicating rows for levels B and C of the variable Fctr, will have two rows of "term_plain" of just Fctr.
- "label" Provides the label attached to the data using labelVector::get_label. When a term is not associated with a label, the value of term_plain is returned instead. Note that, variable names will disassociate with a label if they are used in a function (such as factor(x) or x^2.
- "level" Indicates the level being compared within a factor (or an interaction involving a factor), otherwise it returns NA. It may also be said that this value is the appendix to a factor name. For the term FctrB, this would just be B.

- "level_detail" Gives additional information to level by including the reference level of the factor. For the term FctrB, this would return "B vs A". When an interaction with a numeric variable is present, the level for the numeric may be either term_plain or label, the choice being controlled by the level_detail argument.

Restrictions

The descriptors, other than "term", generally don't make sense for data frame objects. The use of tidy_levels_labels is not permitted within the dust function, but is allowed if you really want it by pixiedust:::tidy_levels_labels.

Other special cases noted in future uses will be documented here, but in general, if it isn't a model object, you probably don't really want to use this.

Author(s)

Benjamin Nutter

Examples

```
#* Descriptors for lm output with no interactions
mtcars2 <- mtcars
mtcars2$mpg <- labelVector::set_label(mtcars2$mpg, "Gas Mileage")
mtcars2$qsec <-  labelVector::set_label(mtcars2$qsec, "Quarter Mile Time")
mtcars2$am <-  labelVector::set_label(mtcars2$am, "Transmission")
mtcars2$wt <-  labelVector::set_label(mtcars2$wt, "Weight")
mtcars2$gear <-  labelVector::set_label(mtcars2$gear, "Gears")

#* Basic Output for a model with no interactions
#* Note: numeric_level has no impact as there are no
#*       interactions involving numeric variables.

fit <- lm(mpg ~ qsec + factor(am) + wt + factor(gear), data = mtcars2)

pixiedust:::tidy_levels_labels(fit,
  descriptors = c("term", "term_plain", "label", "level", "level_detail"),
  numeric_level = "term")

#* Assign factors ahead of the model. This allows
#* the user to determine the levels that display.
#* Compare the output for 'am' with the output for 'gear'

mtcars2$am <- factor(mtcars2$am, 0:1, c("Automatic", "Manual"))
mtcars2$am <-  labelVector::set_label(mtcars2$am, "Transmission")
    # Label was lost in variable conversion
fit <- lm(mpg ~ qsec + am + wt + factor(gear), data = mtcars2)
pixiedust:::tidy_levels_labels(fit,
  descriptors = c("term", "term_plain", "label", "level", "level_detail"),
  numeric_level = "term")


#* Include an interaction between a factor and numeric.
```

```
fit <- lm(mpg ~ qsec + am * wt + factor(gear), data = mtcars2)
pixiedust:::tidy_levels_labels(fit,
  descriptors = c("term", "term_plain", "label", "level", "level_detail"),
  numeric_level = "term")

#* Now observe how 'level' and 'level_detail' change
#* in the interaction terms as we choose different
#* values for 'numeric_level'

pixiedust:::tidy_levels_labels(fit,
  descriptors = c("term", "term_plain", "label", "level", "level_detail"),
  numeric_level = "term_plain")

pixiedust:::tidy_levels_labels(fit,
  descriptors = c("term", "term_plain", "label", "level", "level_detail"),
  numeric_level = "label")
```

`%>%` *magrittr forward-pipe operator*

Description

Pipe an object forward into a function or call expression

Usage

```
lhs %>% rhs
```

Arguments

`lhs`, `rhs`	A dataset and function to apply to it

`%<>%` *Chain together multiple operations*

Description

Chain together multiple operations and save to the object at the start of the chain. See ‘magrittr‘ documentation for details.

Usage

```
lhs %<>% rhs
```

Arguments

`lhs`, `rhs`	A data set and function to apply it to

Index