Package 'pixiedust'

August 3, 2015

Date 2015-08-03

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

Version 0.1.1

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 (>= 3.2.1)

Imports ArgumentCheck, broom, dplyr, knitr, lazyWeave, magrittr, tidyr

Suggests testthat

License MIT + file LICENSE

LazyData true

VignetteBuilder knitr

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

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

NeedsCompilation no

Author Benjamin Nutter [aut, cre]

Maintainer Benjamin Nutter <nutter@battelle.org>

Repository CRAN

Date/Publication 2015-08-03 18:52:58

R topics documented:

`add_sprinkles` *Modify Attributes of a* `dustpan` *table*

Description

Adds attributes of a plot to be rendered by the `print.dust` method.

Usage

```
## S3 method for class 'dust'
x + y
```

Arguments

`x`	a `dust` object
`y`	a `dust_bunny` object.

Author(s)

Benjamin Nutter

See Also

`dust`, `sprinkle`

`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_attributes` object and a `print_method` object.

Usage

```
dust(object, ..., glance_foot = TRUE, tidy_df = FALSE)
```

Arguments

object	An object that has a tidy method in broom
...	Additional arguments to pass to tidy
glance_foot	Arrange the glance statistics for the foot of the table. (Not scheduled for implementation until version 0.4.0)
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.

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 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 portion 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 table_attributes object stores information to apply to the entire table.

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

Value

Returns an object of class dust

Upcoming Developments

- inspect_dust Function to evaluate a dust object for things such as incompatible columns (the table head might have 7 columns while the body only has 6, for example); sprinkles not supported by the print method (colored text in the console); or sprinkle selections that may cause conflicts (hopefully this won't occur, but there is potential for problems in combining attributes in LaTeX).
- dust_part A wrapper for extracting objects from a dust object. This is intended to assist in building custom heads and feet.

Author(s)

Benjamin Nutter

See Also

tidy

Examples

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

`print.dust` *Print A* `dust` *Table*

Description

Print A `dust` Table

Usage

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

Arguments

`x`	An object of class `dust`
`...`	Additional arguments to pass to the print method. Currently ignored.

Details

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

Author(s)

Benjamin Nutter

Examples

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

`pvalString` *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.

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 significant digits 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.

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)
```

---

sprinkles *Define Customization to a Table*

Description

Customizations to a dust table are added by "sprinkling" with a little extra fairy 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(rows = NULL, cols = NULL, ..., part = c("body", "head", "foot",
  "interfoot", "table"))

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

Arguments

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.
print_method	A character string giving the print method for the table.

Details

Sprinkling is done over the intersection of rows and columns. 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`.

Whenever `part = "table"`, `rows` and `columns` are ignored and the attributes are applied to the entire table.

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.

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")`

Sprinkles

The following list 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. For a complete list of which sprinkles are recognized by each output type, see `vignette("sprinkles", package = "pixiedust")`.

- `bg` A character string denoting the color for the background. See "Colors".
- `bg_pattern` This is one of the few exceptions to the length 1 rule. This accepts a vector of any length. Background colors are recycled in a pattern. See "Colors". If left unspecified but `bg_pattern_by` is specified, this will default to `c("white", "gainsboro")`.
- `bg_pattern_by` A character string denoting if the background pattern is recycled over rows or columns. Accepts either `"rows"`, or `"columns"` with partial matching and defaults to `"rows"`. If `bg_pattern` is provided, `bg_pattern_by` is assumed, meaning it is not necessary to explicitly define `bg_pattern_by` unless changing an existing or default setting.

- bold Logical value. If TRUE, text is rendered in bold.
- border_collapse Logical. Defaults to TRUE. This element is only applicable to part = "table" and will be applied to the table regardless the value of part in the call.
- border This is one of the few exceptions to the length 1 rule. Accepts values "left", "right", "top", "bottom", and "all" with partial matching. The border will be added to the sides indicated.
- border_thickness A numeric value denoting the thickness of the border. Defaults to 1.
- border_units A character string taking any one of the values "px" or "pt" with partial matching. Defaults to "px".
- border_style A character string taking any one of the values "solid", "dashed", "dotted", "double", "groove", "ridge", "inset", "outset", "hidden", or "none". Defaults to "solid".
- border_color A character string denoting the color for the border. See "Colors".
- fn A function to apply to values in cells. The function should be an expression that acts on the variable value. For example, quote(round(value, 3)).
- font_color A character string denoting the color of the font. See "Colors".
- font_size A numeric value denoting the size of the font.
- font_size_units A character string giving the units of the font size. Accepts values "px", "pt", "%", and "em". Defaults to "px".
- halign A character string denoting the horizontal alignment. Accepts any one of the values "left", "center", or "right", with partial matching.
- height A numerical value giving the height of the cells.
- height_units A character string giving the units for the height argument. Accepts "px" and "%". Defaults to "px".
- italic Logical value. If TRUE, text is rendered in italics.
- pad A numerical value giving the cell padding in pixels.
- rotate_degree A numerical value that determines the angle of rotation in the clockwise direction. Use negative values to rotate counter clockwise.
- round A numerical value for the number of decimal places to which numerical values are rounded. This can also be accomplished through the fn argument, but this argument makes it a bit easier to do.
- valign A character string giving the vertical alignment for the cells. Accepts the values "top", "middle", or "bottom" with partial matching.
- width A numerical value giving the width of the cells.
- width_units A character string giving the units for the width argument. Accepts "px" and "%". Defaults to "px".

Colors

Color specifications accept X11 color names ("orchid"), hexidecimal names ("#DA70D6"), rgb names ("rgb(218 112 214)"), and rgba (rgb+alpha transparancy; "rgba(218, 112, 214, .75)"). Refer to https://en.wikipedia.org/wiki/Web_colors#X11_color_names.

Author(s)

Benjamin Nutter

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_colnames *Column Names for* dust *Tables*

Description

Assigns new column names to a table

Usage

```
sprinkle_colnames(...)
```

Arguments

...	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

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)
```

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

Description

This is a copy of the documentation for `%>%` in `dplyr`. The copy here is made to conform to CRAN requirements regarding documentation. Please see the `dplyr` documenation for the complete and current documentation.

Usage

```
lhs %>% rhs
```

Arguments

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

Index