Package 'pixiedust'

April 13, 2016

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

Version 0.6.4

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 (>= 3.2.1)

Imports ArgumentCheck, broom, checkmate, dplyr, Hmisc (>= 3.14-6), htmltools, knitr, lazyWeave, magrittr, stringr, tidyr

Suggests rmarkdown, testthat

License MIT + file LICENSE

LazyData true

VignetteBuilder knitr

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

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

RoxygenNote 5.0.1

Author Benjamin Nutter [aut, cre]

Maintainer Benjamin Nutter <nutter@battelle.org>

RemoteType local

RemoteUrl C:/Users/Nutter/Documents/GitHub/pixiedust

RemoteUsername nutterb

RemoteRepo pixiedust

R topics documented:

`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 ineffcient 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).

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)
```

---

assert_match_arg *Checkmate Compatible Version of* match.arg

Description

Matches arguments as match.arg and returns the value of the selected argument.

Usage

```
assert_match_arg(x, choices, several_ok = FALSE, ...)
```

Arguments

x	A character vecotr (of length one unless several.ok is TRUE or NULL.
choices	A character vector of candidate values
several_ok	logical specifying if arg should be allowed to have more than one element.
...	Additional arguments to pass to either assertChoice or assertSubset. assertChoice is used when several_ok = FALSE, otherwise assertSubset is used.

See Also

match.arg, assertChoice, assertSubset

---

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,
  justify = "center", float = getOption("pixie_float", TRUE),
  longtable = getOption("pixie_longtable", FALSE),
  hhline = getOption("pixie_hhline", FALSE),
  bookdown = getOption("pixie_bookdown", FALSE))

## 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	Whe 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 includes 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.

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.

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.

Value

Returns an object of class dust

Upcoming Developments

- 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 glance_foot tidy_levels_labels

Examples

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

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

`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")

## 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 *Tables So Beautifully Fine-Tuned You Will Believe It's Magic.*

Description

The `pixiedust` mission is to provide a user friendly and flexible interface by which report-quality tables may be rendered in multiple output formats. Intially, `pixiedust` will support markdown, HTML, and LaTeX formats, as well as methods for console output.

Details

The advantage of `pixiedust` is that it gives you the control to alter the appearance of a table by as little as one cell at a time. This fine-tuned control gives you enormous flexibility in how the final table looks with minimal pre and post processing.

Additionally, `pixiedust` is largely built on top of the `broom` package, allowing for simple and fast generation of tables based on analytical results.

The chief disadvantage of `pixiedust` is that it can be extremely verbose. If you are applying many customizations, you will find yourself writing a great deal of code.

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.

Usage

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

Arguments

X	An object of class dust_list.
FUN	A function to apply to each element of X
...	Additional arguments to pass to FUN

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)

## 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.

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 rended 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))
```

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 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)

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"))

## 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.
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. 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. This feature is not yet implemented (2015-08-05) and may be removed, depending on how useful it turns out to be.

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"`

In `sprinkle`, when `part = "table"`, the attributes are assigned to the entire table. This is not yet active and may be removed entirely.

The `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 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		
	action	Modifies the background color of a cell.
	default	
	accepts	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		
	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
	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
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
	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"
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	
	accepts	numeric(1)
	console	Recognized
	markdown	Recognized
	html	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.

HTML Colors

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

HTML color names are not case sensitive, but the color names in LaTeX output are. If you desire to be able to toggle your output between HTML and LaTeX, it is recommended that you use the

color names under the dvips section of page 38 of the LaTeX package `xcolor` manual (https://www.ctan.org/pkg/xcolor.

LaTeX Colors

Use of color in LaTeX requirements requires that you have the LaTeX `color` package included in your document preamble (`\usepackage{color}`). Rmarkdown documents include the color package automatically. The standard colors available in LaTeX are "white", "black", "red", "green", "blue", "cyan", "magenta", and "yellow".

Additional colors may be made available using the LaTeX package `xcolor`. To be consistent with color names used in the HTML tables, it is recommended that you use the option `\usepackage[dvipsnames]{xcolor}` in your preamble. Please note that color names in LaTeX are case-sensitive, but the HTML names are not. If the ability to switch between output methods is something you desire, you should adopt the capitalization used in the dvips names (See page 38 of the `xcolor` manual; https://www.ctan.org/pkg/xcolor).

If desired, you may also use the `[x11names]` option to have the X11 color names available to you.

The LaTeX output will accept hexidecimal names (`"#DA70D6"`) and rgb names (`"rgb(218 112 214)"`), similar to the HTML colors described above. However, transparency is not supported. If the transparency value is provided, it is silently ignored.

Custom color definitions may also be defined by defining the color in the preamble. The process for color definitions is described in the `xcolor` documentation. Keep in mind that custom color designations in LaTeX output will not transfer the other output formats.

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. In my opinion, the lines drawn by `cline` have a slightly better appearance that `hhline`, but they overwrite horizontal cell borders. If using both backgrounds and borders, it is advantageous to use the `hhline` option.

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_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)
```

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 environment generated by ArgumentCheck::newArgCheck. Under normal circumstances, this is passed from dust. If NULL, as in the case it is run outside of dust, a new environment is created and the argument check is completed 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 Hmisc::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
Hmisc::label(mtcars2$mpg) <- "Gas Mileage"
Hmisc::label(mtcars2$qsec) <- "Quarter Mile Time"
Hmisc::label(mtcars2$am) <- "Transmission"
Hmisc::label(mtcars2$wt) <- "Weight"
Hmisc::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"))
Hmisc::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")
```

`%>%` *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

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

Description

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

Usage

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

Arguments

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

Index