

Package ‘pixiedust’

August 29, 2016

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

Version 0.7.4

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

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

NeedsCompilation no

Author Benjamin Nutter [aut, cre]

Maintainer Benjamin Nutter <nutter@battelle.org>

Repository CRAN

Date/Publication 2016-06-17 21:46:30

R topics documented:

as.data.frame.dust	2
assert_match_arg	3
dust	4
glance_foot	7
medley	8
medley_all_borders	9
pixiedust	10
pixieply	10
pixie_count	11
print.dust	12
pvalString	13
sprinkle	14
sprinkle_colnames	26
tidy_levels_labels	27
%>%	30
%<>%	30
Index	31

as.data.frame.dust	<i>Convert dust Object to Data Frame</i>
--------------------	--

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).

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,
  .var.name = checkmate::vname(x), add = NULL)
```

Arguments

x	A character vector (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.
.var.name	character(1), name of the checked object to print in assertions. Defaults to the heuristic implemented in vname
add	AssertCollection, Collection to store assertion messages. See link[checkmate]{AssertCollection}
...	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](#)

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),
      hhrline = getOption("pixie_hhrline", 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

<code>object</code>	An object that has a <code>tidy</code> method in <code>broom</code>
<code>...</code>	Additional arguments to pass to <code>tidy</code>
<code>tidy_df</code>	When <code>object</code> is an object that inherits the <code>data.frame</code> 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 <code>TRUE</code> .

keep_rownames	When <code>tidy_df</code> is <code>FALSE</code> , setting <code>keep_rownames</code> 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 <code>broom::tidy.matrix</code> 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 <code>NULL</code> , 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 <code>glance_foot</code> .
byrow	A logical, defaulting to <code>FALSE</code> , that indicates if the requested statistics are placed with priority to rows or columns. See the full documentation for the unexported function <code>glance_foot</code> .
descriptors	A character vector indicating the descriptors to be used in the table. Acceptable inputs are <code>"term"</code> , <code>"term_plain"</code> , <code>"label"</code> , <code>"level"</code> , and <code>"level_detail"</code> . 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, <code>"term"</code> , returns only the term descriptor and is identical to the output provided by <code>broom::tidy</code> methods. See Details for a full explanation of each option and the Examples for sample output. See the full documentation for the unexported function <code>tidy_levels_labels</code> .
numeric_level	A character string that determines which descriptor is used for numeric variables in the <code>"level_detail"</code> descriptor when a numeric has an interaction with a factor. Acceptable inputs are <code>"term"</code> , <code>"term_plain"</code> , and <code>"label"</code> . See the full documentation for the unexported function <code>tidy_levels_labels</code> .
label	<code>character(1)</code> . An optional string for assigning labels with which tables can be referenced elsewhere in the document. If <code>NULL</code> , <code>pixiedust</code> attempts to name the label <code>tab:[chunk-name]</code> , where <code>[chunk-name]</code> is the name of the knitr chunk. If this also resolves to <code>NULL</code> (for instance, when you aren't using knitr, the label <code>tab:pixie-[n]</code> is assigned, where <code>[n]</code> is the current value of <code>options()\$pixie_count</code> . 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	<code>character(1)</code> . Specifies the justification of the table on the page. May be <code>"center"</code> (default), <code>"left"</code> , or <code>"right"</code> .
float	A logical used only in LaTeX output. When <code>TRUE</code> , the table is set within a table environment. The default is <code>TRUE</code> , as with <code>xtable</code> .
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 <code>FALSE</code> , indicating that only one table is printed (default); <code>TRUE</code> that the table should be split into multiple tables with the default number of rows per table (see <code>"Longtable"</code>);

	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 <code>sprinkle</code> .
<code>hhline</code>	Logical. When FALSE, the default, horizontal LaTeX cell borders are drawn using the <code>\cline</code> command. These don't necessarily play well with cell backgrounds, however. Using <code>hhline = TRUE</code> prints horizontal borders using the <code>\hhline</code> command. While the <code>hhline</code> output isn't disrupted by cell backgrounds, it may require more careful coding of the desired borders. In <code>hhline</code> , cells with adjoining borders tend to double up and look thicker than when using <code>cline</code> .
<code>bookdown</code>	Logical. When TRUE, bookdown style labels are generated. Defaults to FALSE.
<code>ungroup</code>	Used when a <code>grouped_df</code> object is passed to <code>dust</code> . 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.
<code>x</code>	A dust object
<code>table</code>	A data frame of similar dimensions of the part being replaced.
<code>part</code>	The part of the table to replace with <code>table</code>

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

<code>fit</code>	A model object with a broom: <code>:glance</code> method.
<code>col_pairs</code>	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.
<code>total_cols</code>	The total number of columns in the body of the pixiedust table
<code>glance_stats</code>	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.
<code>byrow</code>	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-pairings 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` argument 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. Initially, `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

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

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 | <i>Print A dust Table</i> |
|------------|---------------------------|

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 <code>knitr::asis_output</code> . See Details. |
| linebreak_at_end | Used only in HTML tables; defines the number of line break tags <code></br></code> 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 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

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

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 than $1 \times 10^{-\text{digits}}$ 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"), 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

| | |
|---------------------------|--|
| <code>x</code> | A dust object |
| <code>rows</code> | A numeric vector specifying the rows of the table to sprinkle. See details for more about sprinkling. |
| <code>cols</code> | A numeric (or character) vector specifying the columns (or column names) to sprinkle. See details for more about sprinkling. |
| <code>...</code> | named arguments, each of length 1, defining the customizations for the given cells. See "Sprinkles" for a listing of these arguments. |
| <code>part</code> | A character string denoting which part of the table to modify. |
| <code>fixed</code> | <code>logical(1)</code> indicating if the values in <code>rows</code> and <code>cols</code> 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 <code>fixed = TRUE</code> , they must share the same length. |
| <code>recycle</code> | 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). |
| <code>print_method</code> | 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 |
| | console | Not recognized |
| | markdown | Not recognized |
| bg_pattern | html | Accepts any of the listed formats; recognizes transparency |
| | latex | Accepts any of the listed formats, but ignores transparency |
| | action | Generates a pattern of background colors. Can be used to make striping by rows or by columns. |
| | default | <code>c("#FFFFFF", "#DDDDDD")</code> |
| | accepts | A vector of color names: dvips color names; rgb(R,G,B); rgba(R,G,B,A); #RRGGBB; #RRGGBBAA |
| bg_pattern_by | 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 |
| | action | Determines if a 'bg_pattern' is patterned by row or by columns. |
| bold | default | "rows" |
| | accepts | "rows", "columns", "cols" |
| | console | Not recognized |
| | markdown | Not recognized |
| | html | Recognized |
| | latex | Recognized |

| | | |
|-----------------|----------|---|
| | 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; hhtmline = FALSE | accepts "solid", "dashed", "dotted", "hidden", "none"
"dotted" is silently changed to "dashed"
"hidden" and "none" are equivalent. |
| | latex; hhtmline = 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 |

| | | |
|--------------|----------|--|
| | latex | alignment of numeric values, but this may change. 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 |
| height_units | latex | Recognized |
| | action | Determines the units in which 'height' is measured |
| | default | "pt" |
| | accepts | "px", "pt", "cm", "in", "%" |
| | console | Not recognized |
| | markdown | Not recognized |
| hhline | html | Recognized |
| | latex | Recognized; "px" is coerced to "pt" |
| | action | Toggles the option for cell border drawing with the 'hhline' LaTeX package |
| | default | FALSE |
| | accepts | logical(1) |
| | console | Not recognized |
| hhline | 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. |
| | action | Renders the text in the cell in italic |
| | default | FALSE |
| | accepts | logical(1) |
| italic | console | Recognized; rendered as an underscore on either side of the text. |
| | markdown | Recognized |
| | html | Recognized |
| | latex | Recognized |
| | action | Justifies the entire table on the page. |
| | default | "center" |
| justify | 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 |
| width_units | latex | Recognized |
| | 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) |
|------------------------------|---|
| <code>font_color</code> | <code>\usepackage[dvipsnames]{xcolor}</code> |
| <code>bg, bg_pattern</code> | <code>\usepackage[dvipsnames, table]{xcolor}</code> |
| <code>border_style</code> | <code>\usepackage{arydshln}</code> |
| | <code>\usepackage{amssymb}</code> |
| | <code>\usepackage{hhline}</code> |
| (with vertical dashed lines) | <code>\usepackage{graphicx}</code> |
| | <code>\makeatletter</code> |
| | <code>\newcommand*\vdashline{\rotatebox[origin=c]{90}{\$\dabar\dabar\dabar\$}}</code> |
| | <code>\makeatother</code> |
| <code>longtable</code> | <code>\usepackage{longtable}</code> |
| | (Must be loaded before <code>arydshln</code>) |
| <code>merge</code> | <code>\usepackage{multirow}</code> |
| captions for non floats | <code>\usepackage{caption}</code> |

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_colnames | <i>Column Names for dust Tables</i> |
|-------------------|-------------------------------------|

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 <code>model.frame</code> method. It is unclear at the moment (18 Sept. 2015) what will happen if an object is passed that does not have a <code>model.frame</code> 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 <code>broom::tidy</code> 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 <code>checkmate::makeAssertCollection</code> . Under normal circumstances, this is passed from <code>dust</code> . If NULL, as in the case it is run outside of <code>dust</code> , a new collection is created and the assertions are reported within <code>tidy_levels_labels</code> . |

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 documentation 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 documentation for the complete and current documentation.

Usage

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

Arguments

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

Index

`%<>%`, 30

`%>%`, 30

`as.data.frame.dust`, 2

`as.data.frame.dust_list`
(`as.data.frame.dust`), 2

`asis_output`, 13

`assert_match_arg`, 3

`assertChoice`, 3

`assertSubset`, 3

`dust`, 4

`get_pixie_count (pixie_count)`, 11

`glance`, 6

`glance_foot`, 5, 7, 7

`increment_pixie_count (pixie_count)`, 11

`latexTranslate`, 23

`match.arg`, 3

`medley`, 8

`medley_all_borders`, 9

`medley_bw (medley)`, 8

`medley_model (medley)`, 8

`pixie_count`, 11

`pixiedust`, 10

`pixiedust-package (pixiedust)`, 10

`pixieply`, 10

`print.dust`, 12

`print.dust_list (print.dust)`, 12

`pvalString`, 13

`redust (dust)`, 4

`set_pixie_count (pixie_count)`, 11

`sprinkle`, 6, 14, 27

`sprinkle_colnames`, 26, 26

`sprinkle_print_method (sprinkle)`, 14

`sprinkle_table (sprinkle)`, 14

`tidy`, 7

`tidy_levels_labels`, 5, 7, 27

`vname`, 3