Package 'cursory'

August 22, 2019

Type Package

Title A Cursory Look at Variables

Version 1.0.0

Maintainer Andrew Redd <Andrew.Redd@hsc.utah.edu>

Description Provides functions for quickly looking at summary statistics for variables in a data set.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Language en-US

RoxygenNote 6.1.1

Imports dplyr, methods, pkgcond, purrr, rlang, tibble, tidymargins, tidyr, tidyselect

Suggests covr, forcats, datasets, dbplyr, testthat, testextra, DBI, RSQLite

URL https://github.com/halpo/cursory

BugReports https://github.com/halpo/cursory/issues

NeedsCompilation no

Author Andrew Redd [aut, cre] (<https://orcid.org/000-0002-6149-2438>)

Repository CRAN

Date/Publication 2019-08-22 08:40:02 UTC

R topics documented:

`add_class` *Add to object helpers*

Description

These function make using pipe functions easier when altering an object.

Usage

```
add_class(x, new)

set_class(x, new)

add_comment(x, new)

set_comment(x, new)
```

Arguments

`x`	object to alter
`new`	new characteristic

Functions

- `set_class`: Overwrite the class
- `add_comment`: Add a comment
- `set_comment`: Overwrite the comment

Examples

```
# set versions will replace anything already there.
a <- set_class(TRUE, 'example')
a <- set_comment(a, 'this is an example of setting a comment')

# Add version will add to `c()` anything there.
b <- add_class(a, 'ex123')
b <- add_comment(b, 'but you can also add more information')

class(a)
class(b)

comment(a)
comment(b)
```

ci *Confidence interval structure*

Description

Create a `confidence-interval` object.

Usage

```
ci(estimate, lower, upper, confidence = 0.95, ...)
```

Arguments

`estimate`	The Estimate
`lower`	Lower bound
`upper`	Upper bound.
`confidence`	confidence level
`...`	other information such as

Value

A `confidence-interval` object.

Examples

```
x <- ci(est=0, low=-1, upp=1)
format(x)
format(x, span='---')

y <- ci(1, 0, 2, span=',')
c(x,y)
```

cursory *Cursory Functions*

Description

Cursory functions act like the `dplyr` `summarize_(all|at|if)` functions with an important difference, they put the variable name in a column and for each function passed in it puts the value in it's own column.

- `cursory_all()` is the analog of `dplyr::summarize_all()`
- `cursory_at()` is the analog of `dplyr::summarize_at()`
- `cursory_if()` is the analog of `dplyr::summarize_if()`

Usage

```
cursory_all(.tbl, .funs, ..., var.name = "Variable")

cursory_at(.tbl, .vars, .funs, ..., var.name = "Variable")

cursory_if(.tbl, .predicate, .funs, ..., var.name = "Variable")
```

Arguments

`.tbl`	A `tbl` object.
`.funs`	A function `fun`, a quosure style lambda `~ fun(.)` or a list of either form.
`...`	Additional arguments for the function calls in `.funs`. These are evaluated only once, with tidy dots support.
`var.name`	Name of the column with variable names.
`.vars`	A list of columns generated by `vars()`, a character vector of column names, a numeric vector of column positions, or `NULL`.
`.predicate`	A predicate function to be applied to the columns or a logical vector. The variables for which `.predicate` is or returns `TRUE` are selected. This argument is passed to `rlang::as_function()` and thus supports quosure-style lambda functions and strings representing function names.

Value

A tibble with columns from the groups, the `var.name` column, and columns corresponding to each of function from `.funs`.

Examples

```
library(dplyr)
data(iris)

## basic summary statistics for each variable in a data frame.
cursory_all(group_by(iris, Species), lst(mean, sd))

## summary statistics for only numeric variables.
cursory_if(iris, is.numeric, lst(mean, sd))

## summary statistics for specific variables.
cursory_at(iris, vars(ends_with("Length")), lst(Variance = var))
```

dontrepeat *Indicate that when printing repeat values should be hidden.*

Description

Indicate that when printing repeat values should be hidden.

Usage

```
dontrepeat(x, replace.with = "")
```

Arguments

x	a vector
replace.with	what to replace the value with.

Examples

```
library(dplyr)
library(tibble)
x <- cursory_all(group_by(iris, Species), lst(mean, sd))
x <- as.data.frame(arrange(x, Species))
print(x)
x[[1]] <- dontrepeat(x[[1]], replace.with='"')
print(x)
```

Npct *Combine count with a percent*

Description

Combine count with a percent

Usage

```
Npct(x, n, ...)

## S4 method for signature 'logical,missing'
Npct(x, n, ...)

## S4 method for signature 'logical,logical'
Npct(x, n, ...)

## S4 method for signature 'integer,integer'
Npct(x, n, ...)

## S4 method for signature 'numeric,numeric'
Npct(x, n, ...)
```

Arguments

x	count or object to count
n	see methods.
...	formatting arguments for formatting the percent. See format.percent.

Value

A character vector formatted with number and percent of successes.

Methods (by class)

- x = logical,n = missing: Count and give percent of TRUE from a logical vector.
- x = logical,n = logical: Count and percent of a logical filtered by a second logical.
- x = integer,n = integer: Provided with count(s) of cases and total(s)
- x = numeric,n = numeric: Provided the actual count and the percent.

Examples

```
# Vector of cases only.
Npct(c(TRUE, FALSE, TRUE))

# Cases with indices
Npct( c(TRUE,FALSE,TRUE,FALSE,TRUE), c(TRUE,TRUE,TRUE,FALSE,FALSE))

# Successes/Total
Npct(2L, 3L)

# Count + percent directly, count must be integerish.
Npct(2, 2/3)
```

---

percent *Designate a numeric vector as a percent.*

Description

The percentage is stored as a formatted string with the original value as an attribute. The formatted value is what will most often be needed but but allows for the original value to be recovered when the attribute is not stripped off.

Usage

```
percent(x, ...)

pct(x, places = attr(x, "places") %||% getOption("percent::places", 2),
  threshold = attr(x, "threshold") %||% getOption("percent::threshold",
  1 * 10^-places), ...)
```

Arguments

x	a numeric object indicating a percentage.
...	additional formatting arguments.
places	Places to show after the decimal point.
threshold	The minimum absolute percentage to show.

Value

- For pct() a string formatted for a percent.
- For percent() the same as pct but is classed as a 'percent' and includes attributes for the raw value.

Functions

- pct: Format a number as a percent.

Examples

```
pct(2/3)           #<-- no class
(x<- percent(2/3)) #<-- has class
as.numeric(x)
```

---

table_1 *Create a Dataset Summary Table (I.E. Table 1)*

Description

This helps creates demographics or summary table for a dataset, the eponymous "table 1". Given a data set a key for columns, describe the differences across the provided factor variables between the levels of key.

Usage

```
table_1(.data, key, .vars = vars(everything()), ...)

table_1_dispatcher(.data, var, name, key)

table_1_summarise(.data, var, name, key)

table_1_summarize(.data, var, name, key)
```

Arguments

.data	a dataset
key	the comparison variable, such as case/control.
.vars	a lazy list of variables to include in the description.
...	passed on to other methods.
var	Variable identifier, used to dispatch
name	name of the variable

Details

table_1_summarise and table_1_dispatcher dispatch on the data type of the variable identified by var in .data

Value

The result is a table which includes:

- A 'Variable' column, can be renamed with the var.name argument

Examples

```
table_1(iris, Species)
table_1(CO2, Plant)
```

Index