Package 'metamer'

September 18, 2019

Title Create Data with Identical Statistics

Version 0.2.0

Description Creates data with identical statistics (metamers) using an iterative algorithm proposed by Matejka & Fitzmaurice (2017) <DOI:10.1145/3025453.3025912>.

URL https://github.com/eliocamp/metamer

BugReports https://github.com/eliocamp/metamer/issues

License GPL-3

Encoding UTF-8

ByteCompile yes

LazyData true

Language en-US

Depends R (>= 2.10)

Imports FNN, progress (>= 1.2.0), methods

Suggests shiny, miniUI, testthat (>= 2.1.0), data.table, covr

RoxygenNote 6.1.1

NeedsCompilation no

Author Elio Campitelli [cre, aut] (<https://orcid.org/0000-0002-7742-9230>)

Maintainer Elio Campitelli <elio.campitelli@cima.fcen.uba.ar>

Repository CRAN

Date/Publication 2019-09-18 18:40:02 UTC

R topics documented:

`delayed_with` *Apply expressions to data.frames*

Description

Creates a function that evaluates expressions in a future data.frame. Is like `with()`, but the data argument is passed at a later step.

Usage

```
delayed_with(...)
```

Arguments

`...`	Expressions that will be evaluated.

Details

Each expression in `...` must return numeric values. They can be named or return named vectors.

Value

A function that takes a `data.frame` and returns the expressions in `...` evaluated in an environment constructed from it.

See Also

Other helper functions: `densify`, `draw_data`, `mean_dist_to`, `mean_self_proximity`, `moments_n`

Examples

```
some_stats <- delayed_with(mean_x = mean(x), mean(y), sd(x), coef(lm(x ~ y)))
data <- data.frame(x = rnorm(20) , y = rnorm(20))
some_stats(data)
```

`densify` *Increase resolution of data*

Description

Interpolates between the output of `draw_data()` and increases the point density of each stroke.Useful for avoiding sparse targets that result in clumping of points when metamerizing. It only has an effect on strokes (made by double clicking).

Usage

```
densify(data, res = 2)
```

Arguments

`data`	A `data.frame` with columns `x`, `y` and `.group`.
`res`	A numeric indicating the multiplicative resolution (i.e. 2 = double resolution).

Value

A `data.frame` with the `x` and `y` values of your data and a `.group` column that identifies each stroke.

See Also

Other helper functions: `delayed_with`, `draw_data`, `mean_dist_to`, `mean_self_proximity`, `moments_n`

`draw_data` *Freehand drawing*

Description

Opens up a dialogue that lets you draw your data.

Usage

```
draw_data(data = NULL)
```

Arguments

`data`	Optional `data.frame` with x and y values that can used as background to guide your drawing.

Value

A `data.frame` with the `x` and `y` values of your data and a `.group` column that identifies each stroke.

See Also

Other helper functions: `delayed_with`, `densify`, `mean_dist_to`, `mean_self_proximity`, `moments_n`

`mean_dist_to` *Mean minimum distance*

Description

Creates a function to get the mean minimum distance between two sets of points.

Usage

```
mean_dist_to(target)
```

Arguments

`target`	A `data.frame` with all numeric columns.

Value

A function that takes a `data.frame` with the same number of columns as `target` and then returns the mean minimum distance between them.

See Also

Other helper functions: `delayed_with`, `densify`, `draw_data`, `mean_self_proximity`, `moments_n`

Examples

```
target <- data.frame(x = rnorm(100), y = rnorm(100))
data <- data.frame(x = rnorm(100), y = rnorm(100))
distance <- mean_dist_to(target)
distance(data)
```

`mean_self_proximity` *Inverse of the mean self distance*

Description

Returns the inverse of the mean minimum distance between different pairs of points. It's intended to be used as a minimizing function to, then, maximize the distance between points.

Usage

```
mean_self_proximity(data)
```

Arguments

`data`	a data.frame

See Also

Other helper functions: `delayed_with`, `densify`, `draw_data`, `mean_dist_to`, `moments_n`

`metamerize` *Create metamers*

Description

Produces very dissimilar datasets with the same statistical properties.

Usage

```
metamerize(data, preserve, minimize = NULL, change = colnames(data),
  signif = 2, N = 100, trim = N, annealing = TRUE,
  perturbation = 0.08, name = NULL, verbose = interactive())
```

Arguments

`data`	A `data.frame` with the starting data or a `metamer_list` object returned by a previous call to the function.
`preserve`	A function whose result must be kept exactly the same. Must take the data as argument and return a numeric vector.
`minimize`	An optional function to minimize in the process. Must take the data as argument and return a single numeric.
`change`	A character vector with the names of the columns that need to be changed.
`signif`	The number of significant digits of `preserve` that need to be preserved.
`N`	Number of iterations.
`trim`	Max number of metamers to return.
`annealing`	Logical indicating whether to perform annealing.
`perturbation`	Numeric with the magnitude of the random perturbations. Can be of length 1 or `length(change)`.
`name`	Character for naming the metamers.
`verbose`	Logical indicating whether to show a progress bar.

Details

It follows Matejka & Fitzmaurice (2017) method of constructing metamers. Beginning from a starting dataset, it iteratively adds a small perturbation, checks if `preserve` returns the same value (up to `signif` significant digits) and if `minimize` has been lowered, and accepts the solution for the next round. If `annealing` is `TRUE`, it also accepts solutions with bigger `minimize` with an ever decreasing probability to help the algorithm avoid local minimums.

If `data` is a `metamer_list`, the function will start the algorithm from the last metamer of the list. Furthermore, if `preserve` and/or `minimize` are missing, the previous functions will be carried over from the previous call.

`minimize` can be also a *vector* of functions. In that case, the process minimizes the product of the functions applied to the data.

Value

A `metamer_list` object (a list of data.frames).

References

Matejka, J., & Fitzmaurice, G. (2017). Same Stats, Different Graphs. Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17, 1290–1294. https://doi.org/10.1145/3025453.3025912

See Also

`delayed_with()` for a convenient way of making functions suitable for `preserve`, `mean_dist_to()` for a convenient way of minimizing the distance to a known target in `minimize`, `mean_self_proximity()` for maximizing the "self distance" to prevent data clumping.

Examples

```
data(cars)
# Metamers of `cars` with the same mean speed and dist, and correlation
# between the two.
means_and_cor <- delayed_with(mean_speed = mean(speed),
                              mean_dist = mean(dist),
                              cor = cor(speed, dist))
set.seed(42)  # for reproducibility.
metamers <- metamerize(cars,
                       preserve = means_and_cor,
                       signif = 3,
                       N = 1000)
print(metamers)

last <- metamers[[length(metamers)]]

# Confirm that the statistics are the same
cbind(original = means_and_cor(cars),
      metamer = means_and_cor(last))

# Visualize
plot(metamers[[length(metamers)]])
```

```
points(cars, col = "red")
```

`moments_n` *Compute moments*

Description

Returns a function that will return uncentered moments

Usage

```
moments_n(orders, cols = NULL)
```

Arguments

`orders`	Numeric with the order of the uncentered moments that will be computed.
`cols`	Character vector with the name of the columns of the data for which moments will be computed. If `NULL`, will use all columns.

Value

A function that takes a `data.frame` and return a named numeric vector of the uncentered moments of the columns.

See Also

Other helper functions: `delayed_with`, `densify`, `draw_data`, `mean_dist_to`, `mean_self_proximity`

Examples

```
data <- data.frame(x = rnorm(100), y = rnorm(100))
moments_3 <- moments_n(1:3)

moments_3(data)

moments_3 <- moments_n(1:3, "x")
moments_3(data)
```

`set_minimize` *Set* `metamer_list` *attributes*

Description

Set attributes of `metamer_lists` that will be used as function arguments in `metamerize()`.

Usage

```
set_minimize(object, minimize)

get_minimize(object)

set_preserve(object, preserve)

get_preserve(object)
```

Arguments

`object`	A `metamer_list` object.
`minimize, preserve`	Minimize and preserve functions as defined in `metamerize()`.

`trim` *Trim a* `metamer_list`

Description

When creating metamers, `metamerize()` can produce thousands of very similar metamers. This function is intended to keep only a subset of them for easier and faster handling and plotting.

Usage

```
trim(object, n = length(object))
```

Arguments

`object`	A `metamer_list` object returned by `metamerize()`
`n`	The number of metamers to keep.

Value

A `metamer_list` object with `n` equally spaced metamers.

Index