Package: PupillometryR (via r-universe)

October 15, 2024

Type Package

Title A Unified Pipeline for Pupillometry Data

Version 0.0.5

Description Provides a unified pipeline to clean, prepare, plot, and
run basic analyses on pupillometry experiments.

BugReports https://github.com/samhforbes/PupillometryR/issues

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0), dplyr, ggplot2, rlang

Imports fda, itsadug, mgcv, signal, stats, stringr, tidyr, utils, zoo

RoxygenNote 7.2.3

Suggests knitr, rmarkdown

VignetteBuilder knitr

Repository https://samhforbes.r-universe.dev

RemoteUrl https://github.com/samhforbes/pupillometryr

RemoteRef HEAD

RemoteSha 278ef3235f533b65dbdeeec9d7128c2beeb850a6

Contents

baseline_data 2
calculate_mean_pupil_size 3
calculate_missing_data 4
clean_missing_data 4
create_difference_data 5
create_functional_data 6
create_time_windows 7
create_window_data 8
detect_blinks_by_column 9

detect_blinks_by_size 10
detect_blinks_by_velocity 11
downsample_time_data 12
filter_data 12
GeomFlatViolin 13
geom_flat_violin 14
gfv_helper1 15
interpolate_data 15
make_pupillometryr_data 16
mean2 17
plot.PupillometryR 17
plot.Pupil_difference_data 18
plot.Pupil_test_data 19
plot.Pupil_window_data 20
pupil_data 21
regress_data 22
replace_missing_data 22
run_functional_t_test 23
subset_data 24

Index **25**

`baseline_data` *Baseline pupil data to the average pupil size within a window*

Description

This function is for use with the PupillometryR package to baseline each participant's pupil size to the mean pupil size within a window. This may not be necessary if you are doing purely within-subject analyses, but it is convenient for comparison across subjects, and makes results more uniform.

Usage

```
baseline_data(data, pupil, start, stop)
```

Arguments

`data`	a PupillometryR dataframe
`pupil`	a column name denoting pupil data
`start`	start time of baseline window
`stop`	stop time of baseline window

Value

A PupillometryR dataframe, with baselined pupil

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                                 subject = ID,
                                 trial = Trial,
                                 time = Time,
                                 condition = Type)
regressed_data <- regress_data(data = Sdata, pupil1 = RPupil, pupil2 = LPupil)
mean_data <- calculate_mean_pupil_size(data = regressed_data,
pupil1 = RPupil, pupil2 = LPupil)
base_data <- baseline_data(data = mean_data, pupil = mean_pupil, start = 0, stop = 100)
```

`calculate_mean_pupil_size`

Calculate a mean size across two pupils over time

Description

This function is useful when you have left and right eye eyetracking data, and a mean of the two would be useful.

Usage

```
calculate_mean_pupil_size(data, pupil1, pupil2)
```

Arguments

`data`	a PupillometryR dataframe
`pupil1`	column name indicating pupil size
`pupil2`	column name indicating pupil size

Value

A PupillometryR dataframe with a mean pupil column

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                                 subject = ID,
                                 trial = Trial,
                                 time = Time,
                                 condition = Type)
regressed_data <- regress_data(data = Sdata, pupil1 = RPupil, pupil2 = LPupil)
mean_data <- calculate_mean_pupil_size(data = regressed_data, pupil1 = RPupil, pupil2 = LPupil)
```

`calculate_missing_data` *Calculate the missing data amount*

Description

This function can be used to assess the amount of samples that have problematic data from each trial, which helps assess cleaning parameters

Usage

```
calculate_missing_data(data, pupil)
```

Arguments

`data`	your data of class PupillometryR
`pupil`	a column name denoting pupil size

Value

A summary table with number of missing samples in each trial

Examples

```
data(pupil_data)
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)
new_data <- downsample_time_data(data = Sdata,
pupil = LPupil,
timebin_size = 50,
option = 'mean')
calculate_missing_data(data = new_data, pupil = LPupil)
```

`clean_missing_data` *Clean missing data above an acceptable threshold*

Description

This function can be used to remove trials and participants who do not meet the threshold for a study. Note that there are two parameters for cleaning, one to remove trials above a threshold, the second to remove participants who drop more than a certain amount of trials.

Usage

```
clean_missing_data(
  data,
  pupil,
  trial_threshold = 1,
  subject_trial_threshold = 1
)
```

Arguments

`data`	your data of class PupillometryR
`pupil`	a column name denoting pupil size
`trial_threshold`	a proportion of missing data over which a trial can be considered lost
`subject_trial_threshold`	a proportion of missing trials over which a participant can be considered lost.

Value

A cleaned PupillometryR dataframe

Examples

```
data(pupil_data)
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)
new_data <- downsample_time_data(data = Sdata,
pupil = LPupil,
timebin_size = 50,
option = 'mean')
calculate_missing_data(data = new_data, pupil = LPupil)
```

`create_difference_data` *Create a difference data frame when dealing with a condition column with 2 levels*

Description

The difference data frame is used when creating a dataframe to do the functional t-test analysis. This function would be the first step in that analysis, after doing the pre-processing. It creates a frame where it treats the condition data as level2 - level1. It will throw an error if there are more than two conditions.

Usage

```
create_difference_data(data, pupil)
```

Arguments

`data`	a PupillometryR dataframe
`pupil`	column name for pupil data

Value

A Pupil_difference_data data frame

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                               subject = ID,
                               trial = Trial,
                               time = Time,
                               condition = Type)
mean_data <- calculate_mean_pupil_size(data = Sdata,
pupil1 = RPupil, pupil2 = LPupil)
base_data <- baseline_data(data = mean_data, pupil = mean_pupil, start = 0, stop = 100)
differences <- create_difference_data(data = base_data, pupil = mean_pupil)
plot(differences, pupil = mean_pupil, geom = 'line')
```

`create_functional_data` *Makes a functional data with splines from a Pupil_difference_data dataframe.*

Description

This function turns difference data into fitted splines in order to carry out functional data analysis. Under the hood this passes basis and order to fda::Data2fd, and fda::create.bspline.basis, and is mandatory before running run_functional_t_test. It is recommended to read the documentation for package fda for further information.

Usage

```
create_functional_data(data, pupil, basis, order)
```

Arguments

`data`	a Pupil_difference_data dataframe
`pupil`	Column name indicating pupil data to fit
`basis`	Integer specifying number of basis functions to create a b-spline basis
`order`	Integer specifying order of b-splines (one higher than the degree)

Value

A Pupil_difference_data dataframe fitted with b-splines.

See Also

fda package

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                               subject = ID,
                               trial = Trial,
                               time = Time,
                               condition = Type)
regressed_data <- regress_data(data = Sdata, pupil1 = RPupil, pupil2 = LPupil)
mean_data <- calculate_mean_pupil_size(data = regressed_data, pupil1 = RPupil, pupil2 = LPupil)
base_data <- baseline_data(data = mean_data, pupil = mean_pupil, start = 0, stop = 100)
differences <- create_difference_data(data = base_data, pupil = mean_pupil)
spline_data <- create_functional_data(data = differences, pupil = mean_pupil, basis = 10, order = 4)
```

`create_time_windows` *Make PupillometryR dataframe into multiple time windows for easy analysis*

Description

This function creates a single collapsed data frame for easy analysis with an anova or model, per condition. By comparison create_window_data allows collapsing all into a single time window.

Usage

```
create_time_windows(data, pupil, breaks)
```

Arguments

`data`	a PupillometryR dataframe
`pupil`	column name denoting pupil data to be used
`breaks`	a vector or numbers indicating start times for each window

Value

a Pupil_window_data dataframe

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                               subject = ID,
                               trial = Trial,
                               time = Time,
                               condition = Type)
regressed_data <- regress_data(data = Sdata, pupil1 = RPupil, pupil2 = LPupil)
mean_data <- calculate_mean_pupil_size(data = regressed_data,
pupil1 = RPupil, pupil2 = LPupil)
base_data <- baseline_data(data = mean_data, pupil = mean_pupil, start = 0, stop = 100)
time_window <- create_time_windows(data = base_data, pupil = mean_pupil,
breaks = c(1000, 2000))
```

`create_window_data` *Make PupillometryR dataframe into a single collapsed window for easy analysis*

Description

This function creates a single collapsed data frame for easy analysis with a t-test or anova, per condition. By comparison create_time_windows allows dividing it into multiple windows per time.

Usage

```
create_window_data(data, pupil)
```

Arguments

`data`	a PupillometryR dataframe
`pupil`	column name denoting pupil data to be used

Value

a Pupil_window_data dataframe

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                               subject = ID,
                               trial = Trial,
                               time = Time,
                               condition = Type)
regressed_data <- regress_data(data = Sdata, pupil1 = RPupil, pupil2 = LPupil)
mean_data <- calculate_mean_pupil_size(data = regressed_data,
pupil1 = RPupil, pupil2 = LPupil)
base_data <- baseline_data(data = mean_data, pupil = mean_pupil, start = 0, stop = 100)
window <- create_window_data(data = base_data, pupil = mean_pupil)
p <- plot(window, pupil = mean_pupil, windows = FALSE, geom = 'boxplot')
p
```

`detect_blinks_by_column`

detect blinks by a pre-existing labelled blink column that comes from the eyetracker

---

Description

This allows the user to remove anything classed as a blink as a result of eyetracker output.

Usage

```
detect_blinks_by_column(
  data,
  pupil,
  column,
  extend_forward = 0,
  extend_back = 0,
  .tag = 1
)
```

Arguments

`data`	dataset of class PupillometryR
`pupil`	column name for pupil data
`column`	column that refers to blinks
`extend_forward`	number of observations to remove forward of blink
`extend_back`	number of obervations to remove behind blink
`.tag`	the variable in the blink column that represents a blink

Value

returns dataframe with blinks removed including forward and back, and data in blink column.

Examples

```
## Not run:
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)

Sdata2 <- detect_blinks_by_column(data = Sdata,
pupil = LPupil,
column = data_in_blink,
extend_forward = 0,
extend_back = 0)
```

```
## End(Not run)
```

`detect_blinks_by_size` *detect blinks by a change in pupil size*

Description

This allows the user to set a threshold for pupil size and remove anything classed as a blink as a result

Usage

```
detect_blinks_by_size(
  data,
  pupil,
  threshold = 2.5,
  extend_forward = 0,
  extend_back = 0
)
```

Arguments

`data`	dataset of class PupillometryR
`pupil`	column name for pupil data
`threshold`	velocity threshold for blink detection
`extend_forward`	number of observations to remove forward of blink
`extend_back`	number of obervations to remove behind blink

Value

returns dataframe with blinks removed including forward and back, and data in blink column.

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)

Sdata2 <- detect_blinks_by_size(data = Sdata,
pupil = LPupil,
threshold = 2.5,
extend_forward = 0,
extend_back = 0)
```

`detect_blinks_by_velocity`

detect blinks by a change in velocity

Description

This allows the user to set a threshold for velocity and remove anything classed as a blink as a result

Usage

```
detect_blinks_by_velocity(
  data,
  pupil,
  threshold = 0.1,
  extend_forward = 0,
  extend_back = 0
)
```

Arguments

`data`	dataset of class PupillometryR
`pupil`	column name for pupil data
`threshold`	velocity threshold for blink detection
`extend_forward`	number of observations to remove forward of blink
`extend_back`	number of obervations to remove behind blink

Value

returns dataframe with blinks removed including forward and back, and data in blink column.

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)

Sdata2 <- detect_blinks_by_velocity(data = Sdata,
pupil = LPupil,
threshold = 0.1,
extend_forward = 0,
extend_back = 0)
```

`downsample_time_data` *Downsample frequency to reduce number of samples and data size*

Description

This function is useful if you were sampling at a very high frequency (eg 500Hz) causing the data size to be hard to manage, and high autocorrelation. Careful decisions should be made about the time bin size and appropriateness of this function, with respect to the data type.

Usage

```
downsample_time_data(data, pupil, timebin_size, option = c("mean", "median"))
```

Arguments

`data`	your data of class PupillometryR
`pupil`	a column name denoting pupil size
`timebin_size`	the size of the new timebin you wish to use
`option`	what should be calculated in each timebin - mean or median. Defaults to mean.

Value

A downsampled dataframe of class PupillometryR

Examples

```
data(pupil_data)
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)
new_data <- downsample_time_data(data = Sdata,
pupil = LPupil,
timebin_size = 50,
option = 'mean')
```

`filter_data` *Run a filter on the data to smooth it out.*

Description

filter_data allows three different options for filtering, a butterworth lowpass filter, a hanning filter, or a median filter. You can also set the degree of this filter; we recommend a default of 11. This filters on one pupil, it can be re-run on a second pupil if needed. Lowpass makes use of the butterworth filter and filtfilt from package signal, median makes use of runmed.

Usage

```
filter_data(
  data,
  pupil,
  filter = c("median", "hanning", "lowpass"),
  degree = 11
)
```

Arguments

`data`	a PupillometryR dataframe
`pupil`	column name for pupil data
`filter`	option for filtering the data
`degree`	filter degree

Value

filtered pupil data

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)
mean_data <- calculate_mean_pupil_size(data = Sdata,
pupil1 = RPupil, pupil2 = LPupil)
filtered_data <- filter_data(data = mean_data,
pupil = mean_pupil,
filter = 'hanning',
degree = 11)
```

`GeomFlatViolin` *geom_flat_violin_HELPER2*

Description

Borrowed from Ben Marwick. Original author David Robinson.

`geom_flat_violin` *ggplot Flat Violin*

Description

ggplot Flat Violin

Usage

```
geom_flat_violin(
  mapping = NULL,
  data = NULL,
  stat = "ydensity",
  position = "dodge",
  trim = TRUE,
  scale = "area",
  show.legend = NA,
  inherit.aes = TRUE,
  ...
)
```

Arguments

`mapping`	A value
`data`	A value
`stat`	A value
`position`	A value
`trim`	A value
`scale`	A value
`show.legend`	A value
`inherit.aes`	A value
`...`	A value

Details

Copy-pasted from https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce947837ef somewhat hackish solution to: https://twitter.com/EamonCaddigan/status/646759751242620928 based mostly on copy/pasting from ggplot2 geom_violin source: https://github.com/hadley/ggplot2/blob/master/R/geom-violin.r The original seems to be: sourced from: https://gist.github.com/dgrtwo/eb7750e74997891d7c20, Author is David Robinson. A key internal function for the raincloud plots used as a plotting option in this package. For information on raincloud plots see: Allen, M., Poggiali, D., Whitaker, K., Marshall, T. R., & Kievit, R. A. (2019). Raincloud plots: a multi-platform tool for robust data visualization. Wellcome open research, 4, 63. doi:10.12688/wellcomeopenres.15191.1

Examples

```
ggplot(diamonds, aes(cut, carat)) +
  geom_flat_violin() +
  coord_flip()
```

`gfv_helper1` *geom_flat_violin_HELPER1*

Description

Borrowed from https://gist.githubusercontent.com/benmarwick/2a1bb0133ff568cbe28d/raw/fb53bd97121f7f9ce947837ef1a
Original author David Robinson, from https://gist.github.com/dgrtwo/eb7750e74997891d7c20

`interpolate_data` *Interpolate across the gaps in data*

Description

Once data is smoothed, it is important to deal with missing observations, such as blinks. This allows simple interpolation over missing values, either linear, or cubic. Depending on the analysis planed, this may not be a necessary option, but it is strongly recommended for the functional analyses planned in this package.

Usage

```
interpolate_data(data, pupil, type = c("linear", "cubic"))
```

Arguments

`data`	a PupillometryR dataframe
`pupil`	Column name for pupil data to be interpolated
`type`	string indicating linear or cubic interpolation to be performed.

Value

interpolated pupillometry data

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)
mean_data <- calculate_mean_pupil_size(data = Sdata,
pupil1 = RPupil, pupil2 = LPupil)
filtered_data <- filter_data(data = mean_data,
pupil = mean_pupil,
filter = 'hanning',
degree = 11)
int_data <- interpolate_data(data = filtered_data,
pupil = mean_pupil,
type = 'linear')
```

`make_pupillometryr_data` *Prepare data for pre-processing in PupillometryR*

Description

This should be the first function you run as part of using PupillometryR. This will make sure your data is in the right format for processing. This package is designed to deal with data at it comes out of the eyetracker in a long-form csv style format. Thus data input here would be a long dataframe, wherein each row is a single frame collected by the eyetracker.

Usage

```
make_pupillometryr_data(data, subject, trial, time, condition, other)
```

Arguments

`data`	a raw, long form dataframe organised by subject, trial, and time. if your data is not long form, look at tidyr for examples of conversion.
`subject`	column name indicating subject ID
`trial`	column name indicating trial ID. This should be unique for participants
`time`	column name indicating time column (should be numeric)
`condition`	column name indicating experimental condition
`other`	any other column you may wish to keep in the data frame for processing

Value

A dataframe ready to use in PupillometryR

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)
```

`mean2` *Helper function mean2*

Description

Somewhat useful function for ignoring NAs

Usage

```
mean2(x)
```

Arguments

`x`	the object

`plot.PupillometryR` *Pre-prepared plots of PupillometryR data*

Description

The plot functions are designed to run with just data and pupil selections, with some additional options for fun with plotting. This allows to see raw data as points, grouped by either subject or condition.

Usage

```
## S3 method for class 'PupillometryR'
plot(
  x,
  pupil,
  group = c("none", "condition", "subject"),
  geom = c("point", "line", "pointrange"),
  model = NULL,
  ...
)
```

Arguments

`x`	A PupillometryR dataframe
`pupil`	Column name of pupil data to be plotted
`group`	What to group the data by (none, condition, or subject)
`geom`	Geom to pass to ggplot. Either point, line, or pointrange.
`model`	Optional argument to plot agains a fitted model
`...`	Ignored

Value

A ggplot object

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)
Sdata2 <- downsample_time_data(data = Sdata,
pupil = LPupil,
timebin_size = 100,
option = 'median')
p <- plot(Sdata2, pupil = LPupil, group = 'subject')
p
```

`plot.Pupil_difference_data`

Pre-prepared plots of PupillometryR data

Description

The plot functions are designed to run with just data and pupil selections, with some additional options for fun with plotting. To see these plots, you must first use create_difference_data.

Usage

```
## S3 method for class 'Pupil_difference_data'
plot(x, pupil, geom = c("point", "line"), colour = "black", ...)
```

Arguments

`x`	A Pupil_difference_data dataframe
`pupil`	Column name of pupil data to be plotted
`geom`	string indicating whether made of connected points or a line
`colour`	string indicating colour of geom, passed to ggplot2
`...`	Ignored

Value

A ggplot object

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                               subject = ID,
                               trial = Trial,
                               time = Time,
                               condition = Type)
regressed_data <- regress_data(data = Sdata, pupil1 = RPupil, pupil2 = LPupil)
mean_data <- calculate_mean_pupil_size(data = regressed_data,
pupil1 = RPupil, pupil2 = LPupil)
base_data <- baseline_data(data = mean_data, pupil = mean_pupil, start = 0, stop = 100)
differences <- create_difference_data(data = base_data,
pupil = mean_pupil)
p <- plot(differences, pupil = mean_pupil, geom = 'line')
p
```

`plot.Pupil_test_data` *Pre-prepared plots of PupillometryR data*

Description

The plot functions are designed to run with just data and pupil selections, with some additional options for fun with plotting. To see these plots, you must first use one of the run_functional tests.

Usage

```
## S3 method for class 'Pupil_test_data'
plot(x, show_divergence = TRUE, colour = "black", fill = "grey", ...)
```

Arguments

`x`	A Pupil_test_data dataframe
`show_divergence`	logical indicating whether divergences are to be highlighted
`colour`	string indicating colour of geom_line, passed to ggplot2
`fill`	string indicating fill hue of divergence highlights, passed to ggplot2
`...`	Ignored

Value

A ggplot object

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                                 subject = ID,
                                 trial = Trial,
                                 time = Time,
                                 condition = Type)
regressed_data <- regress_data(data = Sdata, pupil1 = RPupil, pupil2 = LPupil)
mean_data <- calculate_mean_pupil_size(data = regressed_data,
pupil1 = RPupil, pupil2 = LPupil)
base_data <- baseline_data(data = mean_data, pupil = mean_pupil, start = 0, stop = 100)
differences <- create_difference_data(data = base_data,
pupil = mean_pupil)
spline_data <- create_functional_data(data = differences, pupil = mean_pupil, basis = 10, order = 4)
ft_data <- run_functional_t_test(data = spline_data,
pupil = mean_pupil)
p <- plot(ft_data, show_divergence = TRUE, colour = 'red', fill = 'orange')
p
```

`plot.Pupil_window_data` *Pre-prepared plots of PupillometryR data*

Description

The plot functions are designed to run with just data and pupil selections, with some additional options for fun with plotting. To see these plots, you must first use create_window_data.

Usage

```
## S3 method for class 'Pupil_window_data'
plot(
  x,
  pupil,
  windows = c(FALSE, TRUE),
  geom = c("raincloud", "violin", "boxplot"),
  ...
)
```

Arguments

`x`	A Pupil_window_data dataframe
`pupil`	Column name of pupil data to be plotted
`windows`	Whether you want to include time windows in the plot - logical
`geom`	violin plots or boxplots. The newest version adds raincloud plots using Ben Marwick's flat violin plot.
`...`	Ignored

Value

A ggplot object

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                               subject = ID,
                               trial = Trial,
                               time = Time,
                               condition = Type)
regressed_data <- regress_data(data = Sdata, pupil1 = RPupil, pupil2 = LPupil)
mean_data <- calculate_mean_pupil_size(data = regressed_data,
pupil1 = RPupil, pupil2 = LPupil)
base_data <- baseline_data(data = mean_data, pupil = mean_pupil, start = 0, stop = 100)
window <- create_window_data(data = base_data,pupil = mean_pupil)
p <-plot(window, pupil = mean_pupil, windows = FALSE, geom = 'boxplot')
p
```

`pupil_data` *Data collected in a pupillometry study by Sylvain Sirois*

Description

Data from a simple study measuring pupil dilation as participants answer hard or easy maths problems. Original data sourced and reformatted from Sylvain Sirois' Pupillometry tutorial available at https://oraprdnt.uqtr.uquebec.ca/pls/public/gscw031?owa_no_site=314&owa_no_fiche=3&owa_bottin=)

Usage

```
pupil_data
```

Format

A data frame with 28800 rows and 7 variables:

ID Uniaue participant ID

Trial Unique trial code (also unique for each participant)

RPupil Right pupil size

LPupil Left Pupil Size

Timebin Ordered timebin within each trial

Time Elapsed time within trial

Type Hard or easy trial?...

Source

(https://oraprdnt.uqtr.uquebec.ca/pls/public/gscw031?owa_no_site=314&owa_no_fiche=3&owa_bottin=)

`regress_data` *Regress one pupil against another for extra smoothing*

Description

regress_data runs a simple linear regression of pupil1 against pupil2 and the reverse. This can help to account for small amount of bumpiness in the data. The regression runs over each participant and each trial, per time.

Usage

```
regress_data(data, pupil1, pupil2)
```

Arguments

`data`	a PupillometryR dataframe
`pupil1`	Column name for first pupil data
`pupil2`	Column name for second pupil data

Value

a PupillometryR dataframe with smoothed pupil values

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)
regressed_data <- regress_data(data = Sdata,
pupil1 = RPupil,
pupil2 = LPupil)
mean_data <- calculate_mean_pupil_size(data = regressed_data,
pupil1 = RPupil, pupil2 = LPupil)
```

`replace_missing_data` *replaces missing observations if you have some degree of incomplete observations*

Description

This is a useful function if you have a dataset where certain timepoints have been removed for whatever reason, but you want continuous time data. This will make assumptions about trials being the same length though, so may not be appropriate for all data types. This should only be run after running make_pupillometry_data.

Usage

```
replace_missing_data(data)
```

Arguments

`data`	your data of class pupillometryR

Value

A time-stepped data frame

Examples

```
data(pupil_data)
Sdata <- make_pupillometryr_data(data = pupil_data,
subject = ID,
trial = Trial,
time = Time,
condition = Type)
new_data <- replace_missing_data(data = Sdata)
```

`run_functional_t_test` *Run a functional t-test on a dataframe previously fitted with b-splines.*

Description

This allows running of a functional t-test for a given alpha on pupil data that has been fitted with b-splines. This is only appropriate for functional difference data, as it assumes we are dealing with condition A - condition B.

Usage

```
run_functional_t_test(data, pupil, alpha = 0.05)
```

Arguments

`data`	a Pupil_difference_data fitted with b-splines
`pupil`	column name indicating pupil data to test
`alpha`	an alpha level to be set for the t-test

Value

A Pupil_test_data dataframe

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                                 subject = ID,
                                 trial = Trial,
                                 time = Time,
                                 condition = Type)
regressed_data <- regress_data(data = Sdata, pupil1 = RPupil, pupil2 = LPupil)
mean_data <- calculate_mean_pupil_size(data = regressed_data, pupil1 = RPupil, pupil2 = LPupil)
base_data <- baseline_data(data = mean_data, pupil = mean_pupil, start = 0, stop = 100)
differences <- create_difference_data(data = base_data, pupil = mean_pupil)
spline_data <- create_functional_data(data = differences, pupil = mean_pupil, basis = 10, order = 4)
ft_data <- run_functional_t_test(data = spline_data, pupil = mean_pupil, alpha = 0.05)
```

`subset_data` *Subset data to provide start and finish time windows*

Description

subset_data can be used on a PupillometryR dataframe to subset the time into relevant chunks. This, ideally should be one of the first runctions run, before anything analytical. Use this to indicate a start and stop time to create a new resized dataframe.

Usage

```
subset_data(data, start = NULL, stop = NULL, rezero = T, remove = T)
```

Arguments

`data`	a PupillometryR dataframe
`start`	a single number indicating start time of new dataframe
`stop`	a single number indicating end time of new dataframe
`rezero`	logical, whether time should start from zero
`remove`	logical, remove observations outside of start and stop

Value

a subsetted PupillometryR dataframe

Examples

```
Sdata <- make_pupillometryr_data(data = pupil_data,
                                 subject = ID,
                                 trial = Trial,
                                 time = Time,
                                 condition = Type)
subset_data(Sdata, start = 100, stop = 10000, rezero = TRUE, remove = TRUE)
```

Index

∗ **datasets**
    GeomFlatViolin, 13
    pupil_data, 21

baseline_data, 2

calculate_mean_pupil_size, 3
calculate_missing_data, 4
clean_missing_data, 4
create_difference_data, 5
create_functional_data, 6
create_time_windows, 7
create_window_data, 8

detect_blinks_by_column, 9
detect_blinks_by_size, 10
detect_blinks_by_velocity, 11
downsample_time_data, 12

filter_data, 12

geom_flat_violin, 14
GeomFlatViolin, 13
gfv_helper1, 15

interpolate_data, 15

make_pupillometryr_data, 16
mean2, 17

plot.Pupil_difference_data, 18
plot.Pupil_test_data, 19
plot.Pupil_window_data, 20
plot.PupillometryR, 17
pupil_data, 21

regress_data, 22
replace_missing_data, 22
run_functional_t_test, 23

subset_data, 24