Package 'PupilPre'

December 20, 2025

Type Package

Title Preprocessing Pupil Size Data

Version 0.6.3

Date 2025-12-19

Author Aki-Juhani Kyröläinen [aut, cre],
Vincent Porretta [aut],
Jacolien van Rij [ctb],
Juhani Järvikivi [ctb]

Maintainer Aki-Juhani Kyröläinen <akkyro@gmail.com>

Description Pupillometric data collected using SR Research Eyelink eye trackers requires significant preprocessing. This package contains functions for preparing pupil dilation data for visualization and statistical analysis. Specifically, it provides a pipeline of functions which aid in data validation, the removal of blinks/artifacts, downsampling, and baselining, among others. Additionally, plotting functions for creating grand average and conditional average plots are provided. See the vignette for samples of the functionality. The package is designed for handling data collected with SR Research Eyelink eye trackers using Sample Reports created in SR Research Data Viewer.

Depends R (>= 3.5.0), dplyr (>= 0.8.0), rlang (>= 0.1.1), VWPre (>= 1.2.0)

Imports ggplot2 (>= 2.2.0), mgcv (>= 1.8-16), shiny (>= 0.14.2), tidyr (>= 0.6.0), stats (>= 3.3.2), robustbase (>= 0.93-3), zoo (>= 1.8-4), signal (>= 0.7-6)

License GPL-3

LazyData true

Suggests knitr, rmarkdown, gridExtra

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation no

Repository CRAN

Date/Publication 2025-12-20 14:50:07 UTC

Contents

apply_butter *Applies a Butterworth filter to each event.*

Description

apply_butter applies a Butterworth filter to the pupil size data.

Usage

```
apply_butter(data = data, n = NULL, W = NULL, type = NULL, plane = "z")
```

Arguments

data	A data table object.
n	A number specifying the filter order (as described in butter).
W	The critical frequencies of the filter (as described in butter). W must be a scalar for low-pass or high-pass filters. W must be a two-element vector c(low, high) specifying the lower and upper bands for stop-band or band-pass filters. For digital filters, W must be between 0 and 1 where 1 is the Nyquist frequency.
type	The filter type (as described in butter), one of "low" for a low-pass filter, "high" for a high-pass filter, "stop" for a stop-band (band-reject) filter, or "pass" for a pass-band filter.
plane	A character string (as described in butter), "z" for a digital filter or "s" for an analog filter.

Value

An object of type data table as described in tibble.

Examples

```
# Load example data
data("Pupilex5")

dat <- apply_butter(Pupilex5, n = 1, W = 0.1,
                    type = "low", plane = "z")

# Please see the vignettes for detailed example usage.
vignette("PupilPre_Interpolation_and_Filtering", package="PupilPre")
```

apply_cleanup_change *Applies user-selected changes to auto cleanup*

Description

apply_cleanup_change applies to each event the user-selected changes to the automatic cleanup based on information stored in the RDS file using verify_cleanup_app which was created using either blink_cleanup or artifact_cleanup.

Usage

```
apply_cleanup_change(data = data, LogFile = NULL)
```

Arguments

data	A data frame object created from downsample.
LogFile	A character string indicating the name (and location) of the log file.

Value

An object of type data table as described in tibble.

Examples

```
if (interactive()) {

# Load example data
data("Pupilex3")

# Ensure the log file exists by running cleanup
# Writing log file to temporary folder for the example
dat <- clean_blink(Pupilex3, BlinkPadding = c(100, 100), Delta = 5,
                   MaxValueRun = 5, NAsAroundRun = c(2,2),
                   LogFile = paste0(tempdir(),"/BlinkCleanupLog.rds"))

# Read log file from temporary folder
verify_cleanup_app(dat, LogFile = paste0(tempdir(),"/BlinkCleanupLog.rds"))

# Make verification via the app interface

# Read log file from the temporary folder
dat <- apply_user_cleanup(dat,
                          LogFile = paste0(tempdir(),"/BlinkCleanupLog.rds"))
}

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Cleanup", package="PupilPre")
```

apply_user_cleanup *Applies manual cleanup to the data*

Description

apply_user_cleanup applies to each event the manual cleanup based on data points stored in the RDS file created using user_cleanup_app. The Identified datapoints will be changed to NA.

Usage

```
apply_user_cleanup(data = data, LogFile = NULL)
```

Arguments

data	A data frame object created from downsample.
LogFile	A character string indicating the name (and location) of the log file.

Value

An object of type data table as described in tibble.

Examples

```
if (interactive()) {

# Load example data
data("Pupilex4")

# Ensure log file exists by using the cleanup app
# Writing log file to temporary folder for the example
user_cleanup_app(Pupilex4, LogFile = paste0(tempdir(),"/UserCleanupLog.rds"))

# Make cleanup via the app interface

# Read log file from the temporary folder
dat <- apply_user_cleanup(Pupilex4,
                          LogFile = paste0(tempdir(),"/UserCleanupLog.rds"))
}

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Cleanup", package="PupilPre")
```

baseline *Baseline correct the data*

Description

baseline calculates the average pupil value for the window of Time provided in BaselineWindow. The baseline value is then used in one of three different calculation types that must be specified (Subtraction, Division, or Normalization). Baselining is carried out separately for each event.

Usage

```
baseline(
  data = data,
  BaselineWindow = 0,
  BaselineType = NULL,
  DiffBaseOverride = FALSE
)
```

Arguments

data	A data frame object created from downsample.
BaselineWindow	A numeric vector of length 1 or 2 specifying the time points of the baseline window to be examined. Providing two values indicates the start time and the end time of the baseline, respectively. Providing a single value (i.e., time point) assumes that every preceding time point is part of the baseline (N.B. trials may vary in the size of the baseline window and will result in an error).
BaselineType	A character string specifying Subtraction, Division, or Normalization. Subtraction subtracts the average baseline value; Division divides by the average baseline value; and Normalization subtracts and divides by the average baseline value.
DiffBaseOverride	A logical value indicating whether or not to override the error produced when baseline windows differ in size.

Value

An object of type data table as described in tibble.

Examples

```
# Load example data
data("Pupilex4")

dat <- baseline(Pupilex4, BaselineWindow = c(-500, 0),
                BaselineType = "Subtraction")

# Please see the vignettes for detailed example usage.
```

```
vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

`blink_summary` *Check blinks*

Description

`blink_summary` summarizes Eyelink marked blinks by Event, Subject, or Item.

Usage

```
blink_summary(data, Summary = "Event", ReturnData = FALSE)
```

Arguments

`data`	A data table object output by `create_time_series`.
`Summary`	A character string indicating the type of summary.
`ReturnData`	A logical indicating whether to return a data table containing Start Time information for each event.

Value

Summary information

Examples

```
# Load example data
data("Pupilex3")

blink_summary(Pupilex3, Summary = "Event")

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

butter_filter_app *Plots the effect of Butterworth filtering by event.*

Description

butter_filter_app produces a plot of Butterworth filtered pupil data over the original data to visually inspect the effect of different filter settings.

Usage

```
butter_filter_app(data)
```

Arguments

data	A data table object.

Value

Nothing.

Examples

```
if (interactive()) {

# Load example data
data("Pupilex5")

butter_filter_app(Pupilex5)
}

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Interpolation_and_Filtering", package="PupilPre")
```

check_baseline *Check baseline window for missing data*

Description

check_baseline examines the data in a specified baseline window.

Usage

```
check_baseline(data = data, BaselineWindow = NULL, ReturnData = FALSE)
```

Arguments

data	A data table object output by create_time_series.
BaselineWindow	A numeric vector of length 1 or 2 specifying the time points of the baseline window to be examined. Providing two values indicates the start time and the end time of the baseline, respectively. Providing a single value (i.e., time point) assumes that every preceding time point is part of the baseline (N.B. trials may vary in the size of the baseline window and will result in an error).
ReturnData	A logical indicating whether to return a data table containing information for each event.

Value

Summary information

Examples

```
# Load example data
data("Pupilex3")

check_baseline(Pupilex3, BaselineWindow = c(-500, 0))

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

---

clean_artifact *Automatically clean artifacts.*

Description

clean_artifact performs two stage and distributional automated clean-up of artifacts in the pupil and gaze coordinate data.

Usage

```
clean_artifact(
  data = data,
  MADWindow = 100,
  MADConstant = 2,
  MADPadding = c(200, 200),
  MahaConstant = 2,
  Method = "Robust",
  XandY = TRUE,
  Second = TRUE,
  MaxValueRun = 5,
  NAsAroundRun = c(2, 2),
  LogFile = NULL
)
```

Arguments

data	A data frame object created from ppl_select_recorded_eye.
MADWindow	A numeric value specifying the window size (in msec) to use for the MAD calculation.
MADConstant	A numeric value specifying the constant (a multiplier for the third quartile) when determining MAD outlier status.
MADPadding	A numeric vector of length two containing values (in msec) to pad the identified artifact creating a window within which to operate the cleanup.
MahaConstant	A numeric value specifying the constant (a multiplier for the third quartile) when determining Mahalanobis outlier status.
Method	A character string ("Basic" or "Robust") indicating which method to use for the distance calculation. Basic is a standard Mahalanobis distance calculation based on covariance. Robust is also a Mahalanobis distance, however, it is based on Minimum Covariance Determinant (Rousseeuw and van Driessen, 1999) with reweighted covariance (Pison et al., 2002). For more details, see covMcd.
XandY	A logical value specifying whether to also use horizontal velocity and acceleration in outlier detection.
Second	A logical value specifying whether secondary cleaning should be applied.
MaxValueRun	A numeric value specifying the maximal run of existing values flanked by NAs that could be targeted for removal.
NAsAroundRun	A numeric vector of length two containing values (in number of subsequent NA) to be used to identify straggler runs of data that could be removed.
LogFile	A character string indicating the file name (with extension) of the log file to be created/written. The file keeps track of which events have been cleaned. We suggest "ArtifactCleanupLog.rds".

Value

An object of type data table as described in tibble.

References

Rousseeuw, P. J. and van Driessen, K. (1999) A fast algorithm for the minimum covariance determinant estimator. Technometrics 41, 212–223.

Pison, G., Van Aelst, S., and Willems, G. (2002) Small Sample Corrections for LTS and MCD, Metrika 55, 111–123.

Examples

```
# Load example data
data("Pupilex4")

# Writing log file to temporary folder for the example
dat <- clean_artifact(Pupilex4, MADWindow = 100, MADConstant = 2,
                      MADPadding = c(200, 200), MahaConstant = 2,
                      Method = "Robust", XandY = TRUE, Second = TRUE,
```

```
                        MaxValueRun = 5, NAsAroundRun = c(2,2),
                        LogFile = paste0(tempdir(),"/ArtifactCleanupLog.rds"))

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Cleanup", package="PupilPre")
```

`clean_blink` *Automatically clean Eyelink marked blinks.*

Description

`clean_blink` performs two stage automated clean-up of blinks in the pupil and gaze coordinate data.

Usage

```
clean_blink(
  data = data,
  BlinkPadding = c(100, 100),
  Delta = NA,
  MaxValueRun = 5,
  NAsAroundRun = c(2, 2),
  LogFile = NULL
)
```

Arguments

`data`	A data frame object created from `ppl_select_recorded_eye`.
`BlinkPadding`	A numeric vector of length two containing values (in msec) to pad the marked blink creating a window within which to operate the cleanup.
`Delta`	A numeric value specifying the maximal difference between subsequent pupil values in order to mark greater differences for removal. If NA, the delta will be estimated from the data using the 95th percentile value.
`MaxValueRun`	A numeric value specifying the maximal run of existing values flanked by NAs that could be targeted for removal.
`NAsAroundRun`	A numeric vector of length two containing values (in number of subsequent NA) to be used to identify straggler runs of data that could be removed.
`LogFile`	A character string indicating the file name (with extension) of the log file to be created/written. The file keeps track of which events have been cleaned. We suggest "BlinkCleanupLog.rds".

Value

An object of type data table as described in tibble.

Examples

```
# Load example data
data("Pupilex3")

# Writing log file to temporary folder for the example
dat <- clean_blink(Pupilex3, BlinkPadding = c(100, 100), Delta = 5,
                   MaxValueRun = 5, NAsAroundRun = c(2,2),
                   LogFile = paste0(tempdir(),"/BlinkCleanupLog.rds"))

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Cleanup", package="PupilPre")
```

`compare_summary` *A utility function to compare pupil size data before and after applying the cleanup*

Description

`compare_summary` is a utility function to compare pupil size data before and after applying the cleanup and summarizes a comparison between Pupil and Pupil_Previous by Event

Usage

```
compare_summary(data, ReturnData = FALSE)
```

Arguments

`data`	A data table object output by `create_time_series`.
`ReturnData`	A logical indicating whether to return a data table containing the summary information.

Value

Summary information.

Examples

```
# Load example data
data("Pupilex4")

compare_summary(Pupilex4)

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

downsample *Downsample the data*

Description

downsample reduces the sampling rate using median values for the Pupil and gaze coordinates.

Usage

```
downsample(data, SamplingRate = NULL, NewRate = NULL)
```

Arguments

data	A data frame object created from blink_cleanup.
SamplingRate	A positive integer specifying the current sampling rate.
NewRate	A positive integer specifying the desired downsampled rate.

Value

An object of type data table as described in tibble.

Examples

```
# Load example data
data("Pupilex4")

dat <- downsample(Pupilex4, SamplingRate = 250, NewRate = 25)

# Please see the vignettes for detailed example usage.
vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

interpolate_NAs *Interpolation for missing data.*

Description

interpolate_NAs performs interpolation of missing data for the pupil and gaze coordinates (if desired).

Usage

```
interpolate_NAs(data = data, Method = "linear", XandY = TRUE, MinData = 2)
```

Arguments

data	A data frame object created from auto_cleanup.
Method	A character string indicating type of interpolation ("linear" or "spline") as implemented in na.approx.
XandY	A logical specifying if interpolation should also be done on gaze coordinates
MinData	A number indicating the minimum number of data points required in order for interpolation to be applied to the event.

Value

An object of type data table as described in tibble.

Examples

```
# Load example data
data("Pupilex4")

dat <- interpolate_NAs(Pupilex4, Method = "linear",
                       XandY = TRUE, MinData = 2)

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Interpolation_and_Filtering", package="PupilPre")
```

NA_summary *Check missing data*

Description

NA_summary summarizes missing data by Event, Subject, or Item.

Usage

```
NA_summary(data, Summary = "Event", PupilColumn = NULL, ReturnData = FALSE)
```

Arguments

data	A data table object output by create_time_series.
Summary	A character string indicating the type of summary.
PupilColumn	A character string indicating which column to use for the summary.
ReturnData	A logical indicating whether to return a data table containing the summary information.

Value

Summary information.

Examples

```
# Load example data
data("Pupilex3")

NA_summary(Pupilex3, Summary = "Event", PupilColumn = "Pupil")

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

---

plot_compare_app *Plots comparison of Pupil and Pupil_Previous by event.*

Description

plot_compare_app produces a comparison plot of Pupil and Pupil_Previous by event to visual changes.

Usage

```
plot_compare_app(data)
```

Arguments

data	A data table object.

Value

Nothing.

Examples

```
if (interactive()) {

# Load example data
data("Pupilex4")

plot_compare_app(Pupilex4)
}

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

plot_events *Plot each event within a group to a directory*

Description

plot_events plots each event in a group as a multi-panel plot and saves it into specified directory.

Usage

```
plot_events(
  data = data,
  Column = NULL,
  Grouping = "Subject",
  Nrow = 1,
  Ncol = 1,
  Device = "pdf",
  ...
)
```

Arguments

data	A data table object output by ppl_select_recorded_eye.
Column	A character string indicating the column to plot.
Grouping	A character string indicating the column to serve as the grouping. For example, "Subject" will use the subject identifier, producing one image per subject containing all the events for that subject.
Nrow	= A number specifying how many rows per page.
Ncol	= A number specifying how many columns per page.
Device	A character string indicating device type passed to ggsave. By default, this is set to "pdf".
...	Arguments to be passed to ggsave.

Value

Files containing plots.

Examples

```
# Load example data
data("Pupilex3")

# Writing files to temporary folder for the example
plot_events(Pupilex3, Column = "Pupil", Device = "pdf",
            Grouping = "Subject", path = paste0(tempdir(),"/Figs"),
            Nrow = 1, Ncol = 1, width = 11, height = 8.5)
```

```
# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Plotting", package="PupilPre")
```

`plot_summary_app` *Plots summary of subject or item.*

Description

`plot_summary_app` plots summary of a given subject or item.

Usage

```
plot_summary_app(data)
```

Arguments

`data`	A data table object.

Value

Nothing.

Examples

```
if (interactive()) {

# Load example data
data("Pupilex4")

plot_summary_app(Pupilex4)
}

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

`ppl_check_eye_recording`
Check which eyes were recorded during the experiment

Description

`ppl_check_eye_recording` quickly checks which eyes contain gaze data either using the EYE_TRACKED column (if available) or the Right and Left interest area columns. It prints a summary and suggests which setting to use for the `Recording` parameter in the function `ppl_select_recorded_eye`.

Usage

```
ppl_check_eye_recording(data)
```

Arguments

data	A data table object output by create_time_series.

Value

Text feedback and instruction.

Examples

```
# Load example data
data("Pupilex2")

ppl_check_eye_recording(data = Pupilex2)

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

---

ppl_plot_avg *Plots average Pupil.*

Description

ppl_plot_avg calculates the grand or conditional averages with standard error. It then plots the results. N.B.: This function will work for data with a maximum of 2 conditions.

Usage

```
ppl_plot_avg(
  data,
  xlim = NA,
  Column = NULL,
  Averaging = "Event",
  Condition1 = NULL,
  Condition2 = NULL,
  Cond1Labels = NA,
  Cond2Labels = NA,
  ErrorBar = TRUE,
  PupilPreTheme = TRUE,
  ConfLev = 95,
  CItype = "simultaneous",
  ErrorBand = FALSE,
  ErrorType = "SE"
)
```

Arguments

data	A data table object output after having executed create_time_series.
xlim	A vector of two integers specifying the limits of the x-axis.
Column	A character string specifying the desired column.
Averaging	A character string indicating how the averaging should be done. "Event" (default) will produce the overall mean in the data, while "Subject" or "Item" (or, in principle, any other column name) will calculate the grand mean by that factor.
Condition1	A string containing the column name corresponding to the first condition, if available.
Condition2	A string containing the column name corresponding to the second condition, if available.
Cond1Labels	A named character vector specifying the desired custom labels of the levels of the first condition.
Cond2Labels	A named character vector specifying the desired custom labels of the levels of the second condition.
ErrorBar	A logical indicating whether error bars should be included in the plot.
PupilPreTheme	A logical indicating whether the theme included with the function should be applied, or ggplot2's base theme (to which any other custom theme could be added).
ConfLev	A number indicating the confidence level of the CI.
CItype	A string indicating "simultaneous" or "pointwise". Simultaneous performs a Bonferroni correction for the interval.
ErrorBand	A logical indicating whether error bands should be included in the plot.
ErrorType	A string indicating "SE" (Standard Error) or "CI" (Confidence Interval).

Value

Nothing.

Examples

```
# Load example data
data("Pupilex7")

ppl_plot_avg(data = Pupilex7, xlim = c(0, 1900), Column = "Pupil",
             Condition1 = NULL, Condition2 = NULL, Cond1Labels = NA,
             Cond2Labels = NA, ErrorBar = TRUE, PupilPreTheme = TRUE)

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Plotting", package="PupilPre")
```

ppl_plot_avg_cdiff *Plots average difference between two conditions.*

Description

ppl_plot_avg_cdiff calculates the average of differences between two specified conditions along with standard error and then plots the results.

Usage

```
ppl_plot_avg_cdiff(
  data,
  Column = NULL,
  xlim = NA,
  Averaging = "Subject",
  Condition = NULL,
  CondLabels = NA,
  ErrorBar = TRUE,
  PupilPreTheme = TRUE,
  ConfLev = 95,
  CItype = "simultaneous",
  ErrorBand = FALSE,
  ErrorType = "SE"
)
```

Arguments

data	A data table object.
Column	A character vector specifying the desired column.
xlim	A vector of two integers specifying the limits of the x-axis.
Averaging	A character string indicating how the averaging should be done. "Subject" (default) will produce the grand mean in the data, while "Item" (or, in principle, any other column name) will calculate the grand mean by that factor.
Condition	A list containing the column name corresponding to the condition and factor levels to be used for calculating the difference.
CondLabels	A named character vector specifying the desired labels of the levels of the condition.
ErrorBar	A logical indicating whether error bars should be included in the plot.
PupilPreTheme	A logical indicating whether the theme included with the function should be applied, or ggplot2's base theme (to which any other custom theme could be added).
ConfLev	A number indicating the confidence level of the CI.
CItype	A string indicating "simultaneous" or "pointwise". Simultaneous performs a Bonferroni correction for the interval.
ErrorBand	A logical indicating whether error bands should be included in the plot.
ErrorType	A string indicating "SE" or "CI".

Value

Nothing.

Examples

```
# Load example data
data("Pupilex7")

ppl_plot_avg_cdiff(data = Pupilex7, Column = "Pupil",
                   Condition = list(talker = c("EN3", "CH1")),
                   ErrorBar = TRUE, PupilPreTheme = TRUE)

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Plotting", package="PupilPre")
```

---

ppl_plot_avg_contour *Plots average contour surface of pupil data.*

Description

ppl_plot_avg_contour calculates the conditional average of pupil size by Time and a specified continuous variable. It then applies a 3D smooth (derived using gam) over the surface and plots the results as a contour plot.

Usage

```
ppl_plot_avg_contour(
  data,
  Column = NULL,
  Var = NULL,
  Averaging = "Event",
  VarLabel = NULL,
  xlim = NA,
  PupilPreTheme = TRUE,
  Colors = c("gray20", "gray90")
)
```

Arguments

data	A data table object output by either create_time_series.
Column	A string specifying the column to use.
Var	A string containing the column name corresponding to the continuous variable.
Averaging	A character string indicating how the averaging should be done. "Event" (default) will produce the overall mean in the data, while "Subject" or "Item" (or, in principle, any other column name) will calculate the grand mean by that factor.

VarLabel	A string specifying the axis label to use for Var.
xlim	A vector of two integers specifying the limits of the x-axis.
PupilPreTheme	A logical indicating whether the theme included with the function, or ggplot2's base theme (which any other custom theme could be added).
Colors	A vector of two strings specifying the colors of the contour shading - The default values represent grayscale.

Value

Nothing.

Examples

```
# Load example data
data("Pupilex7")

ppl_plot_avg_contour(data = Pupilex7, Column = "Pupil", Var = "TRIAL_INDEX",
                     VarLabel = "Trial", xlim = c(0,2000),
                     PupilPreTheme = TRUE, Colors = c("gray20", "gray90"))

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Plotting", package="PupilPre")
```

---

ppl_prep_data *Check the classes of specific columns and re-assigns as necessary.*

Description

ppl_prep_data checks for necessary columns and converts the class if needed.

Usage

```
ppl_prep_data(
  data,
  Subject = NULL,
  Item = NA,
  EventColumns = c("Subject", "TRIAL_INDEX")
)
```

Arguments

data	A data frame object created from an Eyelink Sample Report.
Subject	An obligatory string containing the column name corresponding to the subject identifier.
Item	An optional string containing the column name corresponding to the item identifier; by default, NA.
EventColumns	A vector specifying the columns which will be used for creating the Event variable; by default, Subject and TRIAL_INDEX.

Value

An object of type data table as described in tibble.

Examples

```
# Load example data
data("Pupilex1")

dat <- ppl_prep_data(Pupilex1, Subject = "RECORDING_SESSION_LABEL",
                     Item = "item",
                     EventColumns = c("Subject","TRIAL_INDEX"))

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

---

ppl_rm_extra_DVcols *Checks for and removes unnecessary DV output columns.*

Description

ppl_rm_extra_DVcols checks for unnecessary DataViewer output columns and removes them, unless specified.

Usage

```
ppl_rm_extra_DVcols(data, Keep = NULL)
```

Arguments

data	A data frame object created from an Eyelink Sample Report.
Keep	An optional string or character vector containing the column names of SR Research sample report columns the user would like to keep in the data set.

Value

An object of type data table as described in tibble.

Examples

```
# Load example data
data("Pupilex1")

dat <- ppl_rm_extra_DVcols(Pupilex1, Keep = NULL)

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

ppl_select_recorded_eye

Select the eye used during recording

Description

ppl_select_recorded_eye examines each event and determines which eye contains interest area information, based on the Recording parameter (which can be determined using ppl_check_eye_recording). This function then selects the data from the recorded eye and copies it to new columns (Pupil, Gaze_X, Gaze_Y, Velocity_X, Velocity_Y, Acceleration_X, Acceleration_Y, In_Blink, In_Saccade). The function prints a summary of the output.

Usage

```
ppl_select_recorded_eye(data, Recording = NULL, WhenLandR = NA)
```

Arguments

data	A data table object output by create_time_series.
Recording	A string indicating which eyes were used for recording gaze data ("R" when only right eye recording is present, "L" when only left eye recording is present, "LorR" when either the left or the right eye was recorded, "LandR" when both the left and the right eyes were recorded).
WhenLandR	A string indicating which eye ("Right" or "Left) to use if gaze data is available for both eyes (i.e., Recording = "LandR").

Value

A data table with 11 additional columns added to data.

Examples

```
# Load example data
data("Pupilex2")

dat <- ppl_select_recorded_eye(data = Pupilex2, Recording = "R",
                               WhenLandR = "Right")

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

Pupildat — *This is a sample pupil size dataset included in the package*

Description

This is a sample pupil size dataset included in the package

Author(s)

Aki Kyröläinen

Pupilex1 — *This is an example dataset to illustrate certain functionality*

Description

This is an example dataset to illustrate certain functionality

Author(s)

Aki Kyröläinen

Pupilex2 — *This is an example dataset to illustrate certain functionality*

Description

This is an example dataset to illustrate certain functionality

Author(s)

Aki Kyröläinen

Pupilex3 — *This is an example dataset to illustrate certain functionality*

Description

This is an example dataset to illustrate certain functionality

Author(s)

Aki Kyröläinen

Pupilex4 *This is an example dataset to illustrate certain functionality*

Description

This is an example dataset to illustrate certain functionality

Author(s)

Aki Kyröläinen

Pupilex5 *This is an example dataset to illustrate certain functionality*

Description

This is an example dataset to illustrate certain functionality

Author(s)

Aki Kyröläinen

Pupilex6 *This is an example dataset to illustrate certain functionality*

Description

This is an example dataset to illustrate certain functionality

Author(s)

Aki Kyröläinen

Pupilex7 *This is an example dataset to illustrate certain functionality*

Description

This is an example dataset to illustrate certain functionality

Author(s)

Aki Kyröläinen

PupilPre *PupilPre: Preprocessing Pupil Size Data.*

Description

The PupilPre package provides a set of functions for preparing pupil size data collected with SR Research Eyelink eye trackers.

Processing functions

- The function `ppl_prep_data` returns a data table with correctly assigned classes for important columns.
- The function `ppl_select_recorded_eye` returns a data table with data from the the recorded eye in new columns (IA_ID and IA_LABEL).
- The function `recode_off_screen` recodes samples with NA if gaze coordinates indicate that the sample was taken while the pupil was off-screen.
- The function `ppl_rm_extra_DVcols` removes DataViewer columns that are not necessary for preprocessing with this package.
- The function `clean_blink` returns a data table for which blinks have been removed based on the parameters provided.
- The function `clean_artifact` returns a data table for which artifacts have been removed based on the parameters provided.
- The function `apply_cleanup_change` applies user-selected changes to the automatic cleanup.
- The function `apply_user_cleanup` returns a data table for which the manually selected artifacts have been removed.
- The function `rm_sparse_events` removes events that do not contain enough data in the baseline and/or critical windows, as specified by the user.
- The function `interpolate_NAs` returns a data table in which NAs have been replaced using linear interpolation.
- The function `apply_butter` applies a Butterworth filter to the pupil size data.
- The function `trim_filtered` removes a specified number of milliseconds from the beginning and the end of each filtered event, as to remove artifacts created by the filter.
- The function `downsample` returns a data table for which the data have been downsampled using the specified sampling rate.
- The function `baseline` returns a data table for which the pupil data have been baselined using the specified method.

Utility functions

- The function `ppl_check_eye_recording` returns a summary of whether or not the dataset contains gaze data in both the Right and Left interest area columns.
- The function `blink_summary` returns a summary of Eyelink marked blinks by Event, Subject, or Item.

- The function NA_summary returns a summary of missing data by Event, Subject, or Item.
- The function check_baseline examines the data within a specified baseline window.
- The function compare_summary returns a summary of comparison between the columns Pupil and Pupil_Previous.

Plotting functions

- The function ppl_plot_avg returns a plot of the grand or conditional averages of pupil dilation along with error bars.
- The function ppl_plot_avg_contour returns a contour plot of the conditional average of pupil dilation over Time and a specified continuous variable.
- The function ppl_plot_avg_cdiff returns a plot of the average difference between two conditions for pupil dilation with error bars.
- The function plot_events returns image files containing plots for each event by a group and saves into a specified directory.

Interactive functions

- The function user_cleanup_app opens a Shiny app for identifying and marking datapoints associated with artifacts.
- The function plot_summary_app opens a Shiny app for inspecting by-subject or by-item averages within a specified time window.
- The function plot_compare_app opens a Shiny app for inspecting events for comparing the current pupil column to the previous pupil column.
- The function verify_cleanup_app plots the data points changed during the previously completed auto cleanup, allowing the user to reject the cleanup for specific events.
- The function butter_filter_app produces a plot of filtered pupil data over the original data to visually inspect the effect of different filter settings.

Notes

- The vignettes are available via `browseVignettes(package = "PupilPre")`.
- A list of all available functions is provided in `help(package = "PupilPre")`.
- This package can be cited using the information obtained from `citation("PupilPre")` or `print(citation("PupilPre"), bibtex = TRUE)`

Author(s)

Aki-Juhani Kyröläinen, Vincent Porretta, Jacolien van Rij, Juhani Järvikivi

Maintainer: Aki-Juhani Kyröläinen, (<akkyro@gmail.com>)

recode_off_screen *Check for samples off-screen and marks as NA.*

Description

recode_off_screen checks samples falling outside the bounds of the screen and marks them with NA

Usage

```
recode_off_screen(data = data, ScreenSize = NULL, PlotData = FALSE)
```

Arguments

data	A data frame object created from an Eyelink Sample Report.
ScreenSize	A numeric vector specifying (in pixels) the dimensions of the x and y of the screen used during the experiment.
PlotData	A logical indicating whether or not to output a visualization of the result.

Value

An object of type data table as described in tibble.

Examples

```
# Load example data
data("Pupilex3")

dat <- recode_off_screen(data = Pupilex3, ScreenSize = c(1920, 1080),
                         PlotData = FALSE)

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

rm_sparse_events *Removes events with excessive missing data*

Description

rm_sparse_events removes events with less data than the specified amount.

Usage

```
rm_sparse_events(
  data = data,
  BaselineWindow = NULL,
  CriticalWindow = NULL,
  BaselineRequired = NULL,
  CriticalRequired = NULL
)
```

Arguments

`data`	A data table object output after having run `ppl_select_recorded_eye`.
`BaselineWindow`	A numeric vector of length 1 or 2 specifying the time points of the baseline window to be examined. Providing two values indicates the start time and the end time of the baseline, respectively. Providing a single value (i.e., time point) assumes that every preceding time point is part of the baseline (N.B. trials may vary in the size of the baseline window and will result in an error).
`CriticalWindow`	A numeric vector of length 1 or 2 specifying the time points of the critical (i.e., post-stimulus) window to be examined. Providing two values indicates the start time and the end time of the window, respectively. Providing a single value (i.e., time point) assumes that every subsequent time point is part of the window.
`BaselineRequired`	A number indicating the percentage of data required in the baseline to be included (i.e., drop events with less than this amount of data).
`CriticalRequired`	A number indicating the percentage of data required in the critical window to be included (i.e., drop events with less than this amount of data).

Value

An object of type data table as described in tibble.

Examples

```
# Load example data
data("Pupilex3")

dat <- rm_sparse_events(data = Pupilex3, BaselineWindow = c(-500, 0),
                        CriticalWindow = c(200, 2000),
                        BaselineRequired = 50,
                        CriticalRequired = 50)

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Basic_Preprocessing", package="PupilPre")
```

trim_filtered *Trim the beginning and end of filtered events.*

Description

trim_filtered removes events skipped by the filter as well as a specified number of milliseconds from the beginning and the end of each filtered event (as to remove artifacts created by the filter).

Usage

```
trim_filtered(data = data, RmSkipped = NULL, RmEdges = NULL)
```

Arguments

data	A data frame object created from auto_cleanup.
RmSkipped	A logical value indicating whether or not to remove events that were skipped during the filtering process (due to NAs).
RmEdges	A numeric vector of length 2 indicating the number of milliseconds to remove from the beginning and end of each event.

Value

An object of type data table as described in tibble.

Examples

```
# Load example data
data("Pupilex6")

dat <- trim_filtered(data = Pupilex6, RmSkipped = TRUE,
                     RmEdges = c(75, 75))

# Please see the vignettes for detailed example usage.
vignette("PupilPre_Interpolation_and_Filtering", package="PupilPre")
```

user_cleanup_app *Interactive app for manually cleaning pupil data.*

Description

user_cleanup_app plots current pupil data and allows the user to select data points which should be removed (changed to NA). The app saves a record of the to-be-executed changes in an RDS file.

Usage

```
user_cleanup_app(data = data, LogFile = NULL)
```

Arguments

data	A data table object.
LogFile	A character string indicating the name (and location) of the log file to be read/written.

Value

Log file.

Examples

```
if (interactive()) {

# Load example data
data("Pupilex4")

# Writing log file to temporary folder for the example
user_cleanup_app(Pupilex4, LogFile = paste0(tempdir(),"/UserCleanupLog.rds"))
}

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Cleanup", package="PupilPre")
```

---

verify_cleanup_app *Interactive app for verifying auto cleanup.*

Description

verify_cleanup_app plots the data points changed during the previously completed auto cleanup and allows the user to verify the cleanup for specific events. The app saves the selection to the RDS file, which can then be used to apply the changes to the data set.

Usage

```
verify_cleanup_app(data = data, LogFile = NULL)
```

Arguments

data	A data table object.
LogFile	A character string indicating the name (and location) of the log file.

Value

Log file

Examples

```
if (interactive()) {

# Load example data
data("Pupilex3")

# Ensure the log file exists by running cleanup
# Writing log file to temporary folder for the example
dat <- clean_blink(Pupilex3, BlinkPadding = c(100, 100), Delta = 5,
                   MaxValueRun = 5, NAsAroundRun = c(2,2),
                   LogFile = paste0(tempdir(),"/BlinkCleanupLog.rds"))

# Read log file from temporary folder
verify_cleanup_app(dat, LogFile = paste0(tempdir(),"/BlinkCleanupLog.rds"))
}

# Please see the vignettes for detailed example usage.
# vignette("PupilPre_Cleanup", package="PupilPre")
```

Index