Package: webgazeR (via r-universe)

January 7, 2025

Type Package

Title Tools for Processing Webcam Eye Tracking Data

Version 0.1.0

Author Jason Geller

Maintainer Jason Geller <drjasongeller@gmail.com>

Description A companion package to gazeR. Functions for reading and pre-processing webcam eye tracking data.

License GPL-3

Encoding UTF-8

LazyData true

Imports tidyverse

RoxygenNote 7.3.2

Depends R (>= 2.10)

Config/pak/sysreqs libfontconfig1-dev libfreetype6-dev libfribidi-dev make libharfbuzz-dev libicu-dev libjpeg-dev libpng-dev libtiff-dev libxml2-dev libssl-dev libx11-dev zlib1g-dev

Repository https://jgeller112.r-universe.dev

RemoteUrl https://github.com/jgeller112/webgazeR

RemoteRef HEAD

RemoteSha bc15ab6a87efd5a3b72bbd2e571c0ccb3f3201f6

Contents

`analyze_sampling_rate` *Analyze Sampling Rates for Eye-Tracking Data*

Description

This function calculates the sampling rate for each subject and trial in an eye-tracking dataset. It provides overall statistics, including the median and standard deviation of sampling rates, and also generates a histogram of median sampling rates by subject, with a density plot overlayed, and a vertical line showing the overall median sampling rate and the standard deviation displayed.

Usage

```
analyze_sampling_rate(eye_data)
```

Arguments

`eye_data`	A dataframe containing eye-tracking data with columns 'subject', 'trial', and 'time'. The column 'time' should represent the time in milliseconds for each trial.

Value

A list containing:

overall_median_SR The overall median sampling rate (Hz).

overall_sd_SR The overall standard deviation of sampling rates.

median_SR_by_subject A dataframe with the median sampling rate by subject.

SR_by_trial A dataframe with the sampling rate by subject and trial.

Examples

```
## Not run:
  # Assuming eye_data is a dataframe with appropriate columns
  result <- analyze_sampling_rate(eye_data)
  print(result)

## End(Not run)
```

assign_aoi *Assign coordinates to areas of interest*

Description

Takes a data frame of gaze positions (or other locations), plus screen size and aoi size (or location), and computes the area of interest (AOI) for each location. Defaults assume standard four-corner design.

Usage

```
assign_aoi(
  gaze,
  screen_size = c(1024, 768),
  aoi_size = c(400, 300),
  aoi_loc = NULL,
  X = "CURRENT_FIX_X",
  Y = "CURRENT_FIX_Y"
)
```

Arguments

gaze	data frame containing positions
screen_size	size of the screen in pixels. Defaults to c(1024, 768) and assumes reversed vertical (i.e., [0,0] is top left).
aoi_size	size of AOIs in pixels. Defaults to a c(400, 300) width-height pair and assumes AOIs are in screen corners. AOIs will be coded numerically 1 to 4 in reading order (left to right, top to bottom), with 0 as center location.
aoi_loc	location of rectangular AOIs. Use as alternative to aoi_size for non-corner AOIs. Each AOI location should be a separate row in a data frame that has variables xmin, xmax, ymin, and ymax. Assumes reversed vertical (i.e., [0,0] is top left). AOIs will be coded numerically in row order.
X	name of variable containing X coordinates. Defaults to "CURRENT_FIX_X"
Y	name of variable containing Y coordinates. Defaults to "CURRENT_FIX_Y"

Value

Original gaze data frame with AOI column added. Non-AOI and off-screen gazes are marked NA.

downsample_gaze *Downsample gaze data*

Description

This function combines gaze samples into time bins and optionally aggregates the data.

Usage

```
downsample_gaze(
  dataframe,
  bin.length = 50,
  timevar = "time",
  aggvars = c("subject", "condition", "target", "trial", "object", "time_bin")
)
```

Arguments

dataframe	DataFrame containing gaze data.
bin.length	Length of time bins (in milliseconds).
timevar	Column name representing time.
aggvars	Vector of variable names to group by for aggregation. Use "none" to skip aggregation.

Value

DataFrame with time bins added and optionally aggregated data.

extract_aois *Extract AOI-related Columns from Webcam Files and Calculate Locations*

Description

This function reads in multiple Gorilla webcam files, extracts the 'loc', 'x_normalised', 'y_normalised', 'width_normalised', and 'height_normalised' columns, and calculates the bounding box coordinates for the AOIs. It also rounds all numeric columns to 3 decimal places.

Usage

```
extract_aois(file_paths, zone_names = NULL)
```

Arguments

file_paths	A list of file paths to webcam files (in .xlsx format).

Value

A dataframe containing distinct rows with AOI-related columns and calculated coordinates.

Examples

```
# Example usage:
# file_paths <- c("file1.xlsx", "file2.xlsx")
# aoi_data <- extract_aois(file_paths)
```

`filter_sampling_rate` *Filter or Label Data Based on Sampling Rate Threshold*

Description

This function allows users to set a sampling rate threshold and choose to either remove the data that falls below the threshold or label it as "bad." Users can apply this threshold either at the subject level, the trial level, or both.

Usage

```
filter_sampling_rate(
  data,
  threshold = NA,
  action = c("remove", "label"),
  by = c("subject", "trial", "both")
)
```

Arguments

`data`	A dataframe that contains the data to be processed. The dataframe should include the columns: **'subject'** Unique identifier for each participant in the dataset. **'med_SR'** Subject-level median sampling rate (Hz). This represents the median sampling rate for a subject across trials. **'SR'** Trial-level sampling rate (Hz). This represents the sampling rate for each specific trial.
`threshold`	Numeric value specifying the sampling rate threshold. Data falling below this threshold will either be removed or labeled as "bad".
`action`	Character string specifying whether to "remove" data that falls below the threshold or "label" it as bad. Acceptable values are '"remove"' or '"label"'. **'"remove"'** Removes rows from the dataset where the sampling rate falls below the threshold. **'"label"'** Adds new columns 'is_bad_subject' and/or 'is_bad_trial' that flag rows where the sampling rate falls below the threshold.

by — Character string specifying whether the threshold should be applied at the "subject" level, the "trial" level, or "both". Acceptable values are ‘"subject"‘, ‘"trial"‘, or ‘"both"‘.

‘"subject"‘ Applies the threshold to the subject-level median sampling rate (‘med_SR‘).

‘"trial"‘ Applies the threshold to the trial-level sampling rate (‘SR‘).

‘"both"‘ Applies the threshold to both the subject-level (‘med_SR‘) and trial-level (‘SR‘) rates. Data is removed/labeled if either rate falls below the threshold.

Value

A dataframe with either rows removed or new columns (‘is_bad_subject‘, ‘is_bad_trial‘) added to indicate whether the data is below the threshold. Additionally, messages will inform the user how many subjects and trials were removed or labeled as "bad."

Output

The function will either return a dataset with rows removed based on the sampling rate threshold or add new columns, ‘is_bad_subject‘ and/or ‘is_bad_trial‘ to the dataset, which indicates whether the data is considered "bad" (i.e., below the sampling rate threshold).

Examples

```
## Not run:
# Example usage of the filter_sampling_rate function
result <- filter_sampling_rate(data = data_with_sr, threshold = 500, action = "remove", by = "both")
# Example usage to label data as "bad"
result <- filter_sampling_rate(data = data_with_sr, threshold = 500, action = "label", by = "trial")

## End(Not run)
```

---

find_location *Find Image Location in a Given Set of Locations*

Description

This function determines the location of an image within a set of locations. The function accepts a vector of locations (such as "TL", "TR", "BL", "BR") and an image identifier. It returns the corresponding location name if the image is found, or ‘NA‘ if the image is not present or is ‘NA‘.

Usage

```
find_location(locations, image)
```

Arguments

locations	A character vector representing the possible locations (e.g., ‘c("TL", "TR", "BL", "BR")‘).
image	A character value representing the image to find in the locations.

Value

A character string representing the location of the image, or ‘NA‘ if the image is not found or is missing.

Examples

```
# Example usage of the find_location function
locations <- c("apple", "banana", "cherry", "date")
find_location(locations, "banana")  # Returns "TR" if locations follow c("TL", "TR", "BL", "BR")
find_location(locations, "orange")  # Returns NA
```

---

gaze_oob *Calculate Out-of-Bounds Proportion by Subject*

Description

This function calculates the number and percentage of points that fall outside a specified range (0, 1) for both X and Y coordinates, grouped by subject.

Usage

```
gaze_oob(
  data,
  subject_col = "subject",
  x_col = "x_pred_normalised",
  y_col = "y_pred_normalised"
)
```

Arguments

data	A data frame containing gaze data.
subject_col	A string specifying the name of the column that contains the subject identifier. Default is "subject".
x_col	A string specifying the name of the column that contains the X coordinate. Default is "x_pred_normalised".
y_col	A string specifying the name of the column that contains the Y coordinate. Default is "y_pred_normalised".

Value

A data frame with the following columns:

subject The subject identifier.

total_points The total number of points for the subject.

outside_count The number of points outside the range for both X and Y coordinates.

x_outside_count The number of points outside the range for the X coordinate.

y_outside_count The number of points outside the range for the Y coordinate.

x_outside_percentage The percentage of points outside the range for the X coordinate.

y_outside_percentage The percentage of points outside the range for the Y coordinate.

Examples

```
## Not run:
  # Example data
  data <- data.frame(
    subject = rep(1:2, each = 100),
    x_pred_normalised = runif(200, -0.5, 1.5),
    y_pred_normalised = runif(200, -0.5, 1.5)
  )

  # Calculate out-of-bounds proportion by subject
  results <- calculate_out_of_bounds_by_subject(data)
  print(results)

## End(Not run)
```

---

merge_webcam_files *Extract and Merge Gorilla Webcam Files with Optional AOI Extraction*

Description

This function reads in multiple Gorilla webcam files, merges them, and optionally extracts area of interest (AOI) data. It cleans up column names, and allows filtering by screen index. If ‘extract_aois‘ is TRUE, it extracts specific AOI-related columns (‘zone_name‘, ‘zone_x_normalized‘, ‘zone_y_normalized‘, ‘zone_width_normalized‘, and ‘zone_height_normalized‘) and returns distinct rows for these columns.

Usage

```
merge_webcam_files(file_paths, screen_index = NULL)
```

Arguments

file_paths	A list of file paths to webcam files (in .xlsx format).
screen_index	An optional screen index to filter the data by. If NULL, the filter will be ignored.
extract_aois	Logical. If TRUE, extracts AOI-related columns and returns distinct rows for them.

Value

A dataframe containing the merged and processed data from the webcam files. If 'extract_aois' is TRUE, it returns a dataframe with distinct AOI-related columns.

Examples

```
# Example usage:
# file_paths <- c("file1.xlsx", "file2.xlsx")
# merged_data <- extract_gorilla_aois(file_paths, screen_index = 4, extract_aois = FALSE)
# aoi_data <- extract_gorilla_aois(file_paths, screen_index = 4, extract_aois = TRUE)
```

---

plot_IA_proportions *Plot Proportion of Looks Over Time for Interest Areas (IAs)*

Description

This function creates a time-course plot of the proportion of looks to specified Interest Areas (IAs). Optionally, it can facet the plot by an experimental condition. Custom labels for each IA can be specified through the 'ia_mapping' argument to define the display order.

Usage

```
plot_IA_proportions(
  data,
  ia_column,
  time_column,
  proportion_column,
  condition_column = NULL,
  ia_mapping
)
```

Arguments

data	A data frame containing the data to plot.
ia_column	The name of the column containing Interest Area (IA) identifiers.
time_column	The name of the column representing time (e.g., milliseconds).
proportion_column	The name of the column with the proportion of looks for each IA.

condition_column
: Optional. The name of the column representing experimental conditions. If not provided, the plot will not be faceted by condition.

ia_mapping
: A named list specifying custom labels for each IA in the desired display order (e.g., 'list(IA1 = "Target", IA2 = "Cohort", IA3 = "Rhyme", IA4 = "Unrelated")').

Value

A ggplot2 plot of the proportion of looks over time for each IA, optionally faceted by condition.

Examples

```
## Not run:
# Example with a condition column
plot_IA_proportions(gaze_data, ia_column = "condition", time_column = "time_ms",
                    proportion_column = "proportion_looks", condition_column = "condition",
                ia_mapping = list(IA1 = "Target", IA2 = "Cohort", IA3 = "Rhyme", IA4 = "Unrelated"))

# Example without a condition column
plot_IA_proportions(gaze_data, ia_column = "condition", time_column = "time_ms",
                      proportion_column = "proportion_looks",
                ia_mapping = list(IA1 = "Target", IA2 = "Cohort", IA3 = "Rhyme", IA4 = "Unrelated"))


## End(Not run)
```

Index