

Package: webgazeR (via r-universe)

September 27, 2026

Type Package

Title Tools for Processing Webcam Eye Tracking Data

Version 0.10.6

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 dplyr, ggplot2, tidyr, purrr (>= 1.0.0), readr, stringr, tibble, rlang, janitor, readxl, ggokabeito, zoo, jsonlite

RoxygenNote 7.3.2

Depends R (>= 4.0)

URL <https://jgeller112.github.io/webgazeR/>,
<https://github.com/jgeller112/webgazeR>

Suggests altdoc, knitr, tinytable

Config/pak/sysreqs libicu-dev libx11-dev

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

Date/Publication 2026-08-28 15:25:57 UTC

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

RemoteRef HEAD

RemoteSha 1146dce0bb67d1cdb9a7be1bb44a0706a6ebbb1d

Contents

analyze_sampling_rate	2
assign_aoi	3
behav_data	4

calculate_isc	4
downsample_gaze	5
extract_aois	6
eyedata	6
filter_sampling_rate	7
find_location	8
gaze_dispersion	8
gaze_oob	9
interpolate_gaze	10
jpspsych_eyedata	10
jpspsych_eyedata2	11
labvanced_eyedata	12
labvanced_eyedata_sentence	12
make_webgazer	13
merge_webcam_files	13
plot_IA_proportions	17
psychopy_eyedata	18
smooth_gaze	19
time_window_isc	19
upsample_gaze	20
vwp_counts	21
Index	23

analyze_sampling_rate	Analyze Sampling Rates (Trial and Subject Levels, with Histogram)
-----------------------	---

Description

Computes sampling rate for each trial (subject × trial) and summarizes at the subject level. Uses distinct timepoints to avoid duplicate time values. Plots histogram of subject-level sampling rates.

Usage

```
analyze_sampling_rate(eye_data, summary_stat = "Median")
```

Arguments

- eye_data A dataframe with subject, trial, time columns.
- summary_stat Either "median" (default) or "mean".

Value

A tibble with subject, trial, SR_trial, SR_subject

assign_aoi	<i>Assign coordinates to areas of interest</i>
------------	--

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

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

behav_data

Example behavioural data from a Gorilla experiment

Description

Trial-level behavioural output from the same Gorilla experiment as [eyedata](#). Used in the vignette to illustrate merging behavioural responses with gaze samples.

Usage

```
behav_data
```

Format

A data frame with 66,350 rows and 78 columns. Notable columns include participant and session identifiers (e.g., `Participant.Public.ID`, `Participant.Private.ID`), trial and spreadsheet metadata (`Trial.Number`, `Spreadsheet.Row`, `Zone.Name`), response data (`Reaction.Time`, `Response`, `Correct`), trial design columns (`ANSWER`, `targetword`, `soundfile`, `trialtype`, the `tl*/tr*/bl*/br*` picture and code columns), and a subject identifier matched to `eyedata`.

Source

Gorilla Experiment Builder (<https://gorilla.sc/>).

calculate_isc

Compute Intersubject Correlation (ISC)

Description

Computes ISC using either the **pairwise correlation** method or the **leave-one-out** method. The pairwise method computes a full correlation matrix and averages Fisher z-transformed values. The leave-one-out method computes the correlation of each participant's time series with the mean of all others.

Usage

```
calculate_isc(
  data_matrix,
  method = c("pairwise", "leave-one-out"),
  cor_method = c("pearson", "spearman"),
  return_matrix = FALSE
)
```

Arguments

<code>data_matrix</code>	A numeric matrix or data.frame where rows represent time points and columns represent participants.
<code>method</code>	A string specifying the ISC computation method. Options are "pairwise" (default) or "leave-one-out".
<code>cor_method</code>	Correlation type passed to [stats::cor()]. One of "pearson" (default) or "spearman".
<code>return_matrix</code>	Logical. If 'TRUE' and 'method = "pairwise"', return the full participant-by-participant correlation matrix (self-correlations set to NA) instead of a summary vector. Ignored for 'method = "leave-one-out"'.

Value

A named numeric vector of ISC values (one per participant), or a correlation matrix if 'return_matrix = TRUE'.

<code>downsample_gaze</code>	<i>Downsample gaze data</i>
------------------------------	-----------------------------

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

<code>dataframe</code>	DataFrame containing gaze data.
<code>bin.length</code>	Length of time bins (in milliseconds).
<code>timevar</code>	Column name representing time.
<code>aggvars</code>	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	<i>Extract AOI-related Columns from Webcam Files and Calculate Locations</i>
--------------	--

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.

eyedata	<i>Example webcam eye-tracking data</i>
---------	---

Description

A sample of webcam-based eye-tracking gaze samples exported from a Gorilla visual-world-paradigm experiment. Used to demonstrate the preprocessing pipeline in the package vignette.

Usage

```
eyedata
```

Format

A data frame with 104,823 rows and 24 columns. Key columns include:

subject Participant identifier.
trial Trial number within a participant.
time Sample time within a trial (ms).
time_stamp Raw sample timestamp.
type Sample type (e.g., prediction, calibration).
screen_index Screen index within the trial.

x_pred, y_pred Predicted gaze coordinates in screen pixels.

x_pred_normalised, y_pred_normalised Predicted gaze coordinates normalised to [0, 1].

convergence WebGazer convergence value for the prediction.

face_conf Face-detection confidence for the sample.

zone_name Name of the interest area (zone) the sample belongs to.

zone_x, zone_y, zone_width, zone_height Zone position and size in pixels.

zone_x_normalised, zone_y_normalised, zone_width_normalised, zone_height_normalised Normalised zone position and size.

filename Source data file the sample came from.

Source

Gorilla Experiment Builder (<https://gorilla.sc/>).

filter_sampling_rate	<i>Filter or Label Data Based on Sampling Rate Threshold</i>
----------------------	--

Description

This function applies a sampling rate threshold and either removes or labels "bad" subjects/trials based on their sampling rates.

Usage

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

Arguments

data	A dataframe with columns: subject, trial, SR_subject, SR_trial.
threshold	Numeric. Sampling rate threshold to apply.
action	"remove" (default) to delete bad data or "label" to flag bad data.
by	"subject", "trial", or "both" to specify where to apply the threshold.

Value

A dataframe with either rows removed or bad subjects/trials labeled.

find_location	<i>Find Image Location in a Given Set of Locations</i>
---------------	--

Description

Find Image Location in a Given Set of Locations

Usage

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

Arguments

locations	A character vector of image names at each location.
image	A character string: the image to locate.

Value

A character string representing the location name or NA if not found.

gaze_dispersions	<i>Compute Gaze Dispersion</i>
------------------	--------------------------------

Description

Calculates centroid and dispersion from gaze data.

Usage

```
gaze_dispersions(data, x, y, grouping_vars)
```

Arguments

data	A data frame or tibble.
x	A string: name of the column with X gaze coordinates.
y	A string: name of the column with Y gaze coordinates.
grouping_vars	A character vector of column names to group by (e.g., subject, condition, trial).

Value

A tibble with centroid_x, centroid_y, dispersion, and log_dispersion for each group.

gaze_oob

*Calculate Out-of-Bounds Proportion by Subject and Trial***Description**

This function calculates the number and percentage of gaze points that fall outside the screen dimensions, and optionally removes only the out-of-bounds gaze points.

Usage

```
gaze_oob(
  data,
  subject_col = "subject",
  trial_col = "trial",
  x_col = "x",
  y_col = "y",
  screen_size = c(1, 1),
  remove = FALSE
)
```

Arguments

<code>data</code>	A data frame containing gaze data.
<code>subject_col</code>	A string specifying the name of the column that contains the subject identifier. Default is "subject".
<code>trial_col</code>	A string specifying the name of the column that contains the trial identifier. Default is "trial".
<code>x_col</code>	A string specifying the name of the column that contains the X coordinate. Default is "x".
<code>y_col</code>	A string specifying the name of the column that contains the Y coordinate. Default is "y".
<code>screen_size</code>	A numeric vector of length 2 specifying the screen width and height. Default is <code>c(1, 1)</code> assuming normalized coordinates.
<code>remove</code>	Logical; if TRUE, removes points outside of screen dimensions. Default is FALSE.

Value

A list containing:

subject_results Summary of missingness at the subject level.

trial_results Summary of missingness at the trial level.

data_clean Dataset with optional removal of out-of-bounds points and missingness annotations.

interpolate_gaze	<i>Interpolate missing gaze data (X and Y) within trials, with optional max gap</i>
------------------	---

Description

Interpolate missing gaze data (X and Y) within trials, with optional max gap

Usage

```
interpolate_gaze(  
  x,  
  x_col = "Gaze_X",  
  y_col = "Gaze_Y",  
  trial_col = "Trial",  
  subject_col = "Subject",  
  time_col = "Time",  
  max_gap = Inf  
)
```

Arguments

x	A data frame containing gaze data.
x_col	The name of the X gaze column (as string).
y_col	The name of the Y gaze column (as string).
trial_col	The name of the trial column (default = "Trial").
subject_col	The name of the subject column (default = "Subject").
time_col	The name of the time column used for sorting (default = "Time").
max_gap	Maximum number of consecutive missing samples to interpolate. Gaps larger than this remain NA (default = Inf).

Value

A tibble with interpolated gaze X and Y columns (replacing originals).

jspsych_eyedata	<i>Example jsPsych webcam eye-tracking data (free-sort experiment)</i>
-----------------	--

Description

The output of `merge_webcam_files(kind = "jspsych")` run on a jsPsych + webgazer extension export from an eye-tracking free-sort experiment in Serrano-Carot, Angele, Xu, & Vasilev (2025). Used in the jsPsych vignette to demonstrate the merging pipeline, including flattening a free-sort-plugin-style webgazer_targets field, without requiring the raw JSON export (which isn't available in the browser-based interactive code chunks the package website runs).

Usage

```
jspsych_eyedata
```

Format

A data frame with 9,279 rows and 47 columns, in the standardized long format `merge_webcam_files()` returns (subject/trial/time/x/y), plus trial-level fields (e.g. `trial_type`, `rt`) and the flattened `target_0`-`target_3` boundary columns.

Source

James, Ryskin, Hartshorne, et al. (2025). *What Paradigms Can Webcam Eye-Tracking Be Used For? Attempted Replications of Five Cognitive Science Experiments*. Collabra: Psychology, 11(1), 140755. <https://doi.org/10.1525/collabra.140755>

jspsych_eyedata2	<i>Example jsPsych webcam eye-tracking data (webgazer extension targets)</i>
------------------	--

Description

The output of `merge_webcam_files(kind = "jspsych")` run on a second, separate jsPsych + webgazer extension export from Serrano-Carot, Angele, Xu, & Vasilev (2025), which records webgazer_targets in the extension's own keyed-by-selector shape ("`#scenes`") rather than the free-sort plugin shape in [jspsych_eyedata](#). Kept as a separate dataset – not merged with `jspsych_eyedata` – since it's a distinct experiment with its own subject and design.

Usage

```
jspsych_eyedata2
```

Format

A data frame with 3,686 rows and 36 columns, in the standardized long format `merge_webcam_files()` returns (subject/trial/time/x/y), plus trial-level fields and the flattened `scene_roi_*` boundary columns.

Source

James, Ryskin, Hartshorne, et al. (2025). *What Paradigms Can Webcam Eye-Tracking Be Used For? Attempted Replications of Five Cognitive Science Experiments*. Collabra: Psychology, 11(1), 140755. <https://doi.org/10.1525/collabra.140755>

labvanced_eyedata	<i>Example Labvanced webcam eye-tracking data</i>
-------------------	---

Description

The output of `merge_webcam_files(kind = "labvanced")` run on two participants' Labvanced timeseries, trials, and sessions exports. Used in the Labvanced vignette to demonstrate the merging pipeline without requiring the raw export files (which aren't available in the browser-based interactive code chunks the package website runs).

Usage

```
labvanced_eyedata
```

Format

A data frame with 75,958 rows and 27 columns, in the standardized long format `merge_webcam_files()` returns (subject/trial/time/x/y), plus confidence, trial-level metadata from the trials export (e.g. `task_name`, `condition_id`), and screen/window size from the sessions export.

Source

Labvanced (<https://www.labvanced.com/>).

labvanced_eyedata_sentence	<i>Example Labvanced webcam eye-tracking data, scoped to one task</i>
----------------------------	---

Description

The same data as `labvanced_eyedata`, but merged with `merge_webcam_files(kind = "labvanced", task = "sentence")` instead of the default call – so `trial` is a plain numeral (e.g. "1", "2", ...) rather than the safe-but-verbose default composite id. Used in the Labvanced vignette wherever a numeral trial id is convenient (e.g. filtering/sorting, or the gaze-trace plot).

Usage

```
labvanced_eyedata_sentence
```

Format

A data frame with 55,114 rows and 27 columns, in the same standardized long format as `labvanced_eyedata`.

Source

Labvanced (<https://www.labvanced.com/>).

make_webgazer

*Standardize a Dataframe to WebGazer Conventions***Description**

This function takes a dataframe and renames columns to match WebGazer conventions: subject, trial, time, x, y. All other columns are preserved.

Usage

```
make_webgazer(
  data,
  col_map = list(subject = "subject", trial = "trial", time = "time", x = "x", y = "y")
)
```

Arguments

data A dataframe containing gaze data.

col_map A named list mapping your current columns to WebGazer names: 'subject', 'trial', 'time', 'x', 'y'.

Value

A dataframe with renamed columns but preserves all other original columns.

merge_webcam_files

*Merge and Process Webcam Eye-Tracking Files***Description**

Read, merge, and standardize webcam eye-tracking data from multiple platforms into a single **long-format** data frame with columns: 'subject', 'trial', 'time', 'x', 'y' (+ any retained trial-level metadata). Prints a progress bar (via 'purrr') while files or participant folders are being read, so it's clear the function is still working on larger merges.

Usage

```
merge_webcam_files(
  file_paths,
  screen_index = NULL,
  kind = c("gorilla", "jspsych", "psychopy", "labvanced"),
  col_map = list(subject = "participant_id", trial = "spreadsheet_row", time =
    "time_elapsed", x = "x", y = "y"),
  array_col = NULL,
  array_key = TRUE,
```

```

    trial_filter = NULL,
    out_dir = NULL,
    overwrite = FALSE,
    file_prefix = "eye_long",
    webgazer_target_names = NULL,
    task = NULL
)

```

Arguments

file_paths	Character vector of paths to files to read.
screen_index	Optional. If provided, filters Gorilla data by one or more screen indices (requires a 'screen_index' column in the data).
kind	Data collection platform. One of "gorilla", "jspsych", "psychopy", "labvanced".
col_map	Named list mapping your file's column names to standardized names: 'subject', 'trial', 'time', 'x', 'y'. For jsPsych/PschoPy you usually only need 'subject' and 'trial' because 'time/x/y' are read from the sample arrays.
array_col	For 'kind = "jspsych"' or "psychopy": name of the column containing per-trial eye samples (list-column or JSON string). Required for these kinds.
array_key	Logical. If 'FALSE', assume unkeyed triplets '(t,x,y)'. If 'TRUE', use keys when present.
trial_filter	Optional function 'f(df) -> df' to filter trial rows before parsing (useful for jsPsych; e.g., keep only trials that contain WebGazer samples).
out_dir	Optional. If not 'NULL', writes one CSV per participant (subject) into this folder.
overwrite	Logical. If 'TRUE', overwrite existing per-participant files in 'out_dir'.
file_prefix	Character. Prefix for per-participant output files written to 'out_dir'.
webgazer_target_names	Optional named character vector for 'kind = "jspsych"' JSON files that recorded per-trial target/AOI zones in a 'webgazer_targets' field – either the webgazer extension's own 'targets' parameter (a nested object keyed by CSS selector, e.g. "#scenes", each value itself 'x'/ 'y'/ 'width'/ 'height'/ 'top'/ 'right'/ 'bottom'/ 'left'), or a plugin that logs one row per target with an identifying column instead (e.g. 'jspsych-free-sort's draggable zones, keyed by 'selector'; 'id', 'name', 'target', 'roi', 'label', and 'aoi' are also recognized). This function always flattens 'webgazer_targets', whichever shape it's in, into flat '<selector>_<field>' columns, e.g. 'scene_roi_x', 'scene_roi_y', 'scene_roi_width', etc. – never left as a nested object, and works the same way even if different trials use different shapes, selector sets, or omit 'webgazer_targets' entirely. By default each selector's columns are named from the raw selector string via 'janitor::clean_names()' (e.g. "#scenes" becomes 'number_scenes', giving 'number_scenes_x', 'number_scenes_y', ...). Supply a vector where names are the raw selectors and values are the desired column names, e.g. 'c("#scenes" = "scene_roi")', to control that naming instead. Selectors not present in 'webgazer_target_names' keep the default cleaned name.

`task` For `'kind = "labvanced"'` only: optionally restrict to a single Labvanced `'Task_Name'` (e.g. `"sentence"`). Within one task, `'Trial_Nr'` alone is a stable, familiar numeral trial id, so supplying `'task'` makes `'trial'` just that numeral (e.g. `"1"`, `"2"`, ...) instead of the safe-but-verbose default composite id. Errors if the task spans more than one block (where `'Trial_Nr'` alone wouldn't be unique) or if `'task'` doesn't match any `'Task_Name'` in the data.

Details

The function supports:

- `**Gorilla**` (`'kind = "gorilla"'`): expects already-long data where `'type == "prediction"'`, then renames columns to the standardized schema using `'col_map'`.
- `**jsPsych**` (`'kind = "jspsych"'`): supports either (a) jsPsych `**JSON trials exports**` that contain a top-level `'data'` object (e.g., `'subject-<id>.json'`), or (b) tabular files (.csv/.tsv/.xlsx) where each row is a trial and `'array_col'` contains per-trial eye samples (as a JSON string). For JSON files, if the subject column is missing, the function will infer it from the filename pattern `'subject-<id>.json'`.
- `**PsychoPy**` (`'kind = "psychopy"'`): expects tabular trial-level data where `'array_col'` contains per-trial eye samples (list-column or JSON string). `'time'` is always converted from seconds to milliseconds, since PsychoPy's clocks report elapsed time in seconds.
- `**Labvanced**` (`'kind = "labvanced"'`): combines Labvanced's per-experiment `**timeseries export**` (one row per logged event, long-format, with columns including `'Trial_Nr'`, `'Trial_Id'`, `'Task_Name'`, `'Exp_Subject_Id'`, `'variable_name'`, `'timestamp'`, `'value'`) with its `**trials export**` (one row per trial, containing trial-level metadata) and, if present, its `**sessions export**` (one row per recording session, with screen/window size in pixels). Files are told apart automatically by their columns, so `'file_paths'` can list them in any order, the trials and sessions files are each optional (with no trials file, only the standardized `'subject'/'trial'/'time'/'x'/'y'/'confidence'` columns are returned), and any file that's none of the three is ignored with a message rather than silently merged in. You can pass every subject's files together in one call (e.g. everything under a study folder via `'list.files(..., recursive = TRUE)'`). Only `'variable_name == "gaze_data"'` rows are kept.

If a sessions export is supplied, `'Screen_Width_In_Pixels'`, `'Screen_Height_In_Pixels'`, `'Window_Width_In_Pixels'`, and `'Window_Height_In_Pixels'` (whichever are present – not the rest of the file, which carries mostly browser/timing metadata unrelated to gaze data) are joined onto every matching row by `'Rec_Session_Id'`. Since `'Rec_Session_Id'` is the join key, a sessions file doesn't need to live in the same folder as the participant it describes.

Labvanced's `'Trial_Nr'` is at least a *per-task* counter, not a session-global one: a `'sentence'` task and a later `'sentence_DC'` task each have their own `'Trial_Nr'` sequence starting back at 1, so the same `'Trial_Nr'` (and even the same `'Trial_Id'`, which turns out to be a per-task randomized item-order id rather than a stable identity) can land on unrelated trials across different tasks or different subjects' sessions – and since a task can be reused across multiple blocks, `'Trial_Nr'` isn't guaranteed not to also reset per block in some study designs. Rather than bet on exactly which level of Labvanced's session > block > task > trial hierarchy resets the counter, `'trial'` defaults to `"<Block_Name>_<Task_Name>_<Trial_Nr>"` (using whichever of those three columns are actually present), and trial metadata is joined on `'Rec_Session_Id' + 'Block_Name' + 'Task_Name' + 'Trial_Nr'` – so a trial can never be confused with an unrelated one from another block, task, session, or subject.

Labvanced's timeseries CSV only names columns through `'value'`, but each `'gaze_data'` row packs three further, unnamed values onto the end of the line. This function always splits those four trailing

values into 'x', 'y', 't' (a high-precision per-sample timestamp), and 'c' (a gaze confidence score, kept as 'confidence'). Some 'gaze_data' events (e.g. right as tracking starts up for a trial) carry no sample at all – those rows are dropped rather than kept as 'NA', so the returned data only ever contains rows with real 'x'/'y'/'time'/'confidence' values.

't' itself is an absolute Unix epoch timestamp in ms (confirmed in Labvanced's own docs: it's the raw camera-capture timestamp, not trial-relative on its own), so it's re-expressed as 'time': milliseconds elapsed from 0 at that trial's first *actual* gaze sample, matching how Gorilla/jsPsych report 'time'. This deliberately doesn't anchor to the trials export's 'trial_start' instead: 'Trial_Start' isn't an official Labvanced system variable, so a 'trial_start' column is whatever a given study's own condition table happened to define it as, not something guaranteed to exist or mean the same thing across studies. Anchoring to the first gaze sample only ever needs the timeseries export itself, so it works the same way for every Labvanced study.

Eye sample formats in 'array_col' The per-trial sample array can be any of: - **Unkeyed triplets**: '[[t, x, y], ...]' (or a 3-column matrix/data.frame) - **Keyed objects**: [{t=..., x=..., y=...}, ...] or [{time=..., x=..., y=...}, ...]

If 'array_key = FALSE', samples are treated as positional '(t, x, y)' and field names are ignored. If 'array_key = TRUE', the parser uses keys (preferring 'col_map\$time/x/y', then falling back to "time"/"t", "x", "y").

Value

A data frame containing aggregated long-format eye data across all files. 'webgazer_targets' is always flattened into flat columns (see 'webgazer_target_names'), never left nested. Any *other* remaining nested column (e.g. jsPsych fields like 'mouse_events', 'final_locations', 'raw_gaze', 'validation_points') is collapsed into a single string per row so the result is always flat and CSV-writable: a flat, unnamed vector becomes pipe-delimited text, while a data frame or named/nested list (structured, per-record data) is serialized as JSON so its field names and row grouping survive instead of being scrambled by a plain 'unlist()'.

Examples

```
## Not run:
# Gorilla (already-long predictions)
df <- merge_webcam_files(
  file_paths = "gorilla_export.csv",
  kind = "gorilla",
  col_map = list(
    subject = "participant_id", trial = "spreadsheet_row",
    time = "time_elapsed", x = "x", y = "y"
  )
)

# jsPsych JSON trials export (subject inferred from filename if needed)
df <- merge_webcam_files(
  file_paths = "subject-6085bd39a5358.json",
  kind = "jpspsych",
  col_map = list(subject = "subject_id", trial = "trial_index"),
  array_col = "webgazer_data",
  array_key = TRUE
)
```

```

# PsychoPy CSV (requires subject_id column in the CSV); `time` is always
# converted from seconds to ms, since PsychoPy's clocks log seconds
df <- merge_webcam_files(
  file_paths = "psychopy_export.csv",
  kind = "psychopy",
  col_map = list(subject = "subject_id", trial = "trial_index"),
  array_col = "gaze_data",
  array_key = TRUE
)

# Labvanced: combine a timeseries export with a trials export
# (order doesn't matter, and the trials file is optional)
df <- merge_webcam_files(
  file_paths = c("timeseries.csv", "trials.csv"),
  kind = "labvanced"
)

# Labvanced, restricted to one task: `trial` becomes a plain numeral
# ("1", "2", ...) instead of the default "<block><task><trial_nr>" id
df <- merge_webcam_files(
  file_paths = c("timeseries.csv", "trials.csv"),
  kind = "labvanced",
  task = "sentence"
)

## End(Not run)

```

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,
  use_color = TRUE
)

```

Arguments

<code>data</code>	A data frame containing the data to plot.
<code>ia_column</code>	The name of the column containing Interest Area (IA) identifiers.
<code>time_column</code>	The name of the column representing time (e.g., milliseconds).
<code>proportion_column</code>	The name of the column with the proportion of looks for each IA.
<code>condition_column</code>	Optional. The name of the column representing experimental conditions. If not provided, the plot will not be faceted by condition.
<code>ia_mapping</code>	A named list specifying custom labels for each IA in the desired display order (e.g., <code>'list(IA1 = "Target", IA2 = "Cohort", IA3 = "Rhyme", IA4 = "Unrelated")'</code>).
<code>use_color</code>	Logical. If <code>'TRUE'</code> (default), the plot will use colors to differentiate Interest Areas. If <code>'FALSE'</code> , different line types, shapes, and line widths will be used instead.

Value

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

<code>psychopy_eyedata</code>	<i>Example PsychoPy webcam eye-tracking data</i>
-------------------------------	--

Description

The output of `merge_webcam_files(kind = "psychopy")` run on a PsychoPy export with per-trial gaze arrays. Used in the PsychoPy vignette to demonstrate the merging pipeline, including the seconds-to-milliseconds time conversion, without requiring the raw CSV export (which isn't available in the browser-based interactive code chunks the package website runs).

Usage

```
psychopy_eyedata
```

Format

A data frame with 13,006 rows and 19 columns, in the standardized long format `merge_webcam_files()` returns (subject/trial/time/x/y), plus trial-level fields (e.g. `task_correct`, `hoo_id`, `win_width`, `win_height`).

Source

PsychoPy (<https://www.psychopy.org/>).

smooth_gaze	<i>Apply a moving average smoothing function to gaze data (X and Y). This is generally recommended after up-sampling the data.</i>
-------------	--

Description

Apply a moving average smoothing function to gaze data (X and Y). This is generally recommended after up-sampling the data.

Usage

```
smooth_gaze(
  x,
  n = 5,
  x_col = "Gaze_X",
  y_col = "Gaze_Y",
  trial_col = "Trial",
  subject_col = "Subject"
)
```

Arguments

x	A data frame containing gaze data.
n	The window size (in samples) for the moving average.
x_col	The name of the X gaze column (as string).
y_col	The name of the Y gaze column (as string).
trial_col	The name of the trial column used for grouping (default = "Trial").
subject_col	The name of the subject column used for grouping (default = "Subject").

Value

A tibble with smoothed gaze X and Y columns (replacing originals).

time_window_isc	<i>Time-windowed Intersubject Correlation (ISC) per subject</i>
-----------------	---

Description

Computes ISC for each subject by correlating that subject's time series against the mean of all *other* subjects within sliding windows, then Fisher z-transforms correlations, averages them per subject, and converts back to r.

Usage

```
time_window_isc(  
  data,  
  window_size = 10,  
  step = 1,  
  min_overlap = 3,  
  method = c("pearson", "spearman"),  
  return_per_window = FALSE  
)
```

Arguments

data	A numeric matrix or data.frame with rows = time points, columns = participants.
window_size	Integer, window length in time points (default 10).
step	Integer, step size between consecutive windows (default 1).
min_overlap	Integer, minimum number of non-NA paired points within a window required to compute a correlation (default 3).
method	Correlation method passed to [stats::cor()], usually "pearson" (default), or "spearman".
return_per_window	Logical; if TRUE, also return a data.frame of per-window Fisher-z correlations per subject (default FALSE).

Details

Correlations of exactly +/-1 are clamped to +/-0.9999999 before Fisher z-transformation so they are included in the average rather than silently dropped.

Value

If 'return_per_window = FALSE', a named numeric vector of ISC values (length = n subjects). If 'TRUE', a list with: - 'isc': named numeric vector of per-subject ISC (as above) - 'per_window': data.frame with columns 'window_start', 'window_end', 'subject', 'z'

upsample_gaze	<i>Up-sample gaze and pupil data</i>
---------------	--------------------------------------

Description

Increase the sampling frequency to 'target_hz' Hz by inserting additional rows. Missing values in gaze and pupil data will be preserved for later interpolation.

Usage

```
upsample_gaze(
  x,
  pupil_cols = c("Pupil_Diameter"),
  gaze_cols = c("x_pred_normalised", "y_pred_normalised"),
  target_hz = 1000,
  upsample_pupil = TRUE
)
```

Arguments

x A dataframe containing gaze and pupil data with columns: ‘subject’, ‘trial’, and ‘time’.

pupil_cols Character vector of pupil diameter column names.

gaze_cols Character vector of gaze position column names.

target_hz Target sampling frequency (default is 1000 Hz).

upsample_pupil Logical; if ‘TRUE’, pupil data will also be upsampled.

Value

A dataframe with up-sampled time points and an ‘up_sampled’ column.

vwp_counts	<i>Visual-world-paradigm fixation counts</i>
------------	--

Description

Aggregated fixation counts by participant, condition, and time bin for a visual-world-paradigm analysis. Suitable for demonstrating empirical-logit or growth-curve modelling of gaze proportions.

Usage

```
vwp_counts
```

Format

A data frame with 814 rows and 6 columns:

subject Participant identifier.

condition_num Numeric coding of condition (e.g., $-0.5 / 0.5$) suitable as a contrast.

time_bin Time bin relative to target onset (ms).

condition Condition label (e.g., "unrelated").

fix Number of samples in the bin that fell on the target interest area.

total_fix Total number of samples in the bin across all interest areas.

Source

Aggregated from the example webcam eye-tracking data shipped with this package.

Index

* datasets

- behav_data, [4](#)
- eyedata, [6](#)
- jspsych_eyedata, [10](#)
- jspsych_eyedata2, [11](#)
- labvanced_eyedata, [12](#)
- labvanced_eyedata_sentence, [12](#)
- psychopy_eyedata, [18](#)
- vwp_counts, [21](#)

[smooth_gaze, 19](#)

[time_window_is, 19](#)

[upsample_gaze, 20](#)

[vwp_counts, 21](#)

[analyze_sampling_rate, 2](#)

[assign_aoi, 3](#)

[behav_data, 4](#)

[calculate_is, 4](#)

[downsample_gaze, 5](#)

[extract_aois, 6](#)

[eyedata, 4, 6](#)

[filter_sampling_rate, 7](#)

[find_location, 8](#)

[gaze_dispersion, 8](#)

[gaze_oob, 9](#)

[interpolate_gaze, 10](#)

[jspsych_eyedata, 10, 11](#)

[jspsych_eyedata2, 11](#)

[labvanced_eyedata, 12, 12](#)

[labvanced_eyedata_sentence, 12](#)

[make_webgazer, 13](#)

[merge_webcam_files, 13](#)

[plot_IA_proportions, 17](#)

[psychopy_eyedata, 18](#)