

Package: gazer (via r-universe)

October 27, 2026

Type Package

Title Tools for Processing Eye Tracking Data

Version 0.2.11

Date 2026-10-26

Description Functions for reading and pre-processing eye tracking data.

URL <http://github.com/dmirman/gazer>

BugReports <http://github.com/dmirman/gazer/issues>

License GPL-3

Depends R (>= 4.1.0)

LazyData true

Imports data.table, dplyr, jsonlite, ggplot2, purrr, readr, signal, stringr, tibble, tidyr, zoo

Suggests cowplot, knitr, lme4, patchwork, rmarkdown, saccades, tidyverse

Remotes tmalsburg/saccades/saccades

VignetteBuilder knitr

Encoding UTF-8

Roxygen list(markdown = TRUE)

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libicu-dev libx11-dev

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

Date/Publication 2026-10-27 02:41:09 UTC

RemoteUrl <https://github.com/dmirman/gazer>

RemoteRef HEAD

RemoteSha 14a4de242128630289cb26661917450f7639b5ba

Contents

assign_aoi	3
baseline_correction_pupil	4
baseline_correction_pupil_msg	5
behave_pupil	6
binify_fixations	6
blink_detect	7
blink_velocity	7
calc_mad	8
code_poly	9
CohortRhyme	10
compare_groups	10
compute_monocular_mean	11
count_missing_pupil	12
Crawford_Howell	12
cursive_agg_data	13
detect_blinks_noise	13
downsample_gaze	14
downsample_pupil	15
dynamic_pupil	16
extend_blinks	16
find_messages_asc	17
find_messages_edf	18
FuncTheme	18
FuncThemePts	19
gazer	20
generate_edf2asc_bat	20
get_gaze_diagnostics	21
get_pvalues	21
get_ranef	22
get_trackloss	23
hanning_filter	23
interpolate_pupil	24
legend_positioning	24
lowpass_filter	25
make_gazer	26
merge_asc_files	27
merge_gazer_files	27
merge_pupil	28
merge_tobii_titta	29
moving_average_pupil	31
NamingRecovery	31
onset_pupil	32
parse_asc	33
parse_edf	33
parse_edf_messages	34
parse_pl	35
process_all_subjects_PL	36

<code>assign_aoi</code>	3
-------------------------	---

<code>read_asc</code>	36
<code>read_bids_eyetrack</code>	38
<code>read_fixation_report</code>	40
<code>resample_gaze</code>	40
<code>smooth_interpolate_pupil</code>	42
<code>speed_pupil</code>	43
<code>TargetFix</code>	43
<code>upsample_gaze</code>	44
<code>WordLearnEx</code>	45

Index	46
--------------	-----------

<code>assign_aoi</code>	<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 <code>"CURRENT_FIX_X"</code>
<code>Y</code>	name of variable containing Y coordinates. Defaults to <code>"CURRENT_FIX_Y"</code>

Value

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

baseline_correction_pupil

Performs linear (sub) or divisive baseline correction relative to desired stim_onset and off_set

Description

Performs linear (sub) or divisive baseline correction relative to desired stim_onset and off_set

Usage

```
baseline_correction_pupil(
  datafile,
  pupil_colname = "pupil",
  baseline_window = NA,
  baseline_method = "sub",
  baseline_stat = "median"
)
```

Arguments

datafile raw pupil data

pupil_colname name of your pupil colname you want baseline corrected

baseline_window user-specified threshold for baseline window.

baseline_method correction method. Default is sub but can also include divisive

baseline_stat statistic for the baseline value: "median" (default) or "mean".

Value

data frame containing baseline corrected data from event of interest

`baseline_correction_pupil_msg`

Performs linear (sub) or divisive baseline correction relative to desired stim_onset and off_set with messages.

Description

Performs linear (sub) or divisive baseline correction relative to desired stim_onset and off_set with messages.

Usage

```
baseline_correction_pupil_msg(  
  datafile,  
  pupil_colname = NULL,  
  baseline_dur = 500,  
  event = "event",  
  baseline_method = "sub",  
  baseline_stat = "median"  
)
```

Arguments

<code>datafile</code>	raw pupil data
<code>pupil_colname</code>	name of your pupil colname you want baseline corrected
<code>baseline_dur</code>	duration of baseline period from stim offset
<code>event</code>	message name for stim onset
<code>baseline_method</code>	correction method. Default is sub but can also include divisive
<code>baseline_stat</code>	statistic for the baseline value: "median" (default) or "mean".
<code>message</code>	name of message column

Value

data frame containing baseline corrected data from event of interest

<code>behave_pupil</code>	<i>culls important behavioral data from the sample report</i>
---------------------------	---

Description

culls important behavioral data from the sample report

Usage

```
behave_pupil(datafile, omiterrors = TRUE, behave_colnames = NULL)
```

Arguments

<code>datafile</code>	raw pupil data
<code>omiterrors</code>	if ture returns RTs without errors for RT analysis or dataset with all error data for error analysis
<code>behave_colnames</code>	specify variables of interest

Value

data frame with or without errors

<code>binify_fixations</code>	<i>Convert fixation list into time bins</i>
-------------------------------	---

Description

Takes a data frame list of fixations and returns a time series with the fixations expanded into time bins of specified duration.

Usage

```
binify_fixations(
  gaze,
  binSize = 20,
  keepCols = c("Subject", "TrialNumber", "Target", "T"),
  maxTime = NULL
)
```

Arguments

<code>gaze</code>	a data frame of fixations, as returned by <code>readFixationReport</code>
<code>binSize</code>	length of bins, in ms
<code>keepCols</code>	set of columns to keep, fixation start and end times will be dropped
<code>maxTime</code>	optional parameter to set maximum trial length

Value

data frame containing gaze data arranged by bin

<code>blink_detect</code>	<i>Take pupil file and detect blinks merge blinks with sample report Requires the saccades package (remotes::install_github("tmalsburg/saccades/saccades")).</i>
---------------------------	--

Description

Take pupil file and detect blinks merge blinks with sample report Requires the saccades package (remotes::install_github("tmalsburg/saccades/saccades")).

Usage

```
blink_detect(samp)
```

Arguments

`samp` pupil data file

Value

merged data file with blinks detected

<code>blink_velocity</code>	<i>This function takes a dataframe and applies a simple velocity algorithm to detect blinks in the data based off velocity of pupil size changes. The algorithm has four parameters that need to be adjusted based on the particulars of the data: the amount of smoothing, the (negative) onset velocity threshold, the (positive) reversal velocity threshold, and the temporal margin</i>
-----------------------------	--

Description

This function takes a dataframe and applies a simple velocity algorithm to detect blinks in the data based off velocity of pupil size changes. The algorithm has four parameters that need to be adjusted based on the particulars of the data: the amount of smoothing, the (negative) onset velocity threshold, the (positive) reversal velocity threshold, and the temporal margin

Usage

```

blink_velocity(
  df,
  neg_velocity_thresh = -5,
  pos_velocity_thresh = 5,
  fillback = 10,
  fillforward = 10,
  hz = 250
)

```

Arguments

df data frame.

neg_velocity_thresh threshold for onset of blinks neg number here we use default of -5

fillback extend blinks backward

fillforward extend blinks forward

hz recording frequency of ET

pos_velocity_thresh threshold for offset of blinks neg number here we use default of 5

Value

data frame containing pupil values where blinks have been linear interpolated.

calc_mad	<i>calculates the median absolute deviation (MAD) and sets the threshold to filter on.</i>
-----------------	--

Description

calculates the median absolute deviation (MAD) and sets the threshold to filter on.

Usage

```
calc_mad(max_dilation, n = 16)
```

Arguments

max_dilation pupil file returned from speed_pupil

n constant used

Value

threshold value to filter on

code_poly	<i>Build polynomial predictor variables</i>
-----------	---

Description

Takes a data frame, name of predictor variable, and polynomial order. Creates polynomial-transformed predictor variables, adds them to the data frame and returns the result. Original data frame is unchanged, remember to assign the result.

Usage

```
code_poly(  
  df = NULL,  
  predictor = NULL,  
  poly.order = NULL,  
  orthogonal = TRUE,  
  draw.poly = TRUE  
)
```

Arguments

df	data frame, should not contain any variables called "predictor"
predictor	string name of predictor variable
poly.order	integer order of polynomial to be created
orthogonal	logical value indicating whether polynomial should be orthogonal (default) or natural (aka raw)
draw.poly	logical value indicating whether to create a graph showing transformed polynomial predictor values, defaults to TRUE

Value

Returns a data frame containing the original data and at least two new columns: "predictor".Index, and a column for each order of the polynomial-transformed predictor

Examples

```
WordLearnEx.gca <- code_poly(df=WordLearnEx, predictor="Block", poly.order=2)  
TargetFix.gca <- code_poly(df=TargetFix, predictor="Time", poly.order=2, orthogonal=FALSE)
```

CohortRhyme

Cohort and rhyme competition during spoken word recognition

Description

Fixation time course data from 8 participants with aphasia and 12 age-matched control participants.

Usage

CohortRhyme

Format

A data frame with 1600 rows and 7 variables:

subjID subject ID, factor

Group subject group: Control, Broca's aphasia, Wernicke's aphasia

Time time (in ms) since word onset in 100ms bins

timeBin time bin number (1-20)

Object object type: phonologically related Competitor or Unrelated object. Target object data are omitted

Type type of phonological competitor: Cohort (e.g., penny - pencil) or Rhyme (e.g., carrot - parrot)

FixProp fixation proportion: proportion of correct-response trials on which the object was fixated

Source

Mirman et al. (2011, Brain & Lang., 117:2, 53-68) http://www.danmirman.org/pdfs/Mirman_etal_2011.pdf

compare_groups

Compare groups on a set of variables.

Description

For each numeric variable in a data frame, computes number of observations, mean, and standard deviation, for each group, and t- and p-values for the t-test comparing the groups.

Usage

compare_groups(dat, Group)

Arguments

dat	data frame
Group	string name of grouping variable

Value

Returns a data frame with rows for each numeric variable in the input data frame (plus N, for number of observations) and columns for mean, SD, t-value, and p-value.

Examples

```
## Not run:
compare_groups(SND, "SemNear_Cond")

## End(Not run)
```

`compute_monocular_mean`*Compute a Monocular Mean Pupil Value*

Description

A "monocular mean" averages both eyes together. If data is available in just one eye, use the available value as the mean. If both eyes are missing, the result is **NA**.

Usage

```
compute_monocular_mean(x1, x2)
```

Arguments

x1	Numeric vector for the left pupil.
x2	Numeric vector for the right pupil.

Value

A numeric vector of monocular mean pupil values.

<code>count_missing_pupil</code>	<i>Calculate percent missing data by subject and by trial</i>
----------------------------------	---

Description

Calculate percent missing data by subject and by trial

Usage

```
count_missing_pupil(datafile, pupil = "pupil", missingthresh = 0.2)
```

Arguments

<code>pupil</code>	name of pupil column
<code>missingthresh</code>	user-specified threshold for missing data. set to .2

Value

data frame containing missing data information

<code>Crawford_Howell</code>	<i>Crawford-Howell (1998) t-test for case-control comparisons.</i>
------------------------------	--

Description

Computes Crawford-Howell t-test comparing a single case to a set of controls.

Usage

```
Crawford_Howell(case, control)
```

Arguments

<code>case</code>	single case score
<code>control</code>	vector of control scores

Value

Returns a data frame with three values: t, df, and p.

Examples

```
Crawford_Howell(649, c(243, 175, 231, 164, 236, 131, 302, 303, 363, 287, 205, 297, 144, 789, 210, 245))
```

cursive_agg_data	<i>Judgments to cursive vs. type-print words in LDT</i>
------------------	---

Description

Aggregate baseline corrected pupillary data for a LDT task that required participants to judge the lexical status of cursive and type-print stimuli.

Usage

```
cursive_agg_data
```

Format

A data frame with 52 rows and 3 variables

script Cursive or Print

timebins trial onset

aggbaseline baseline corrected pupil size

detect_blinks_noise	<i>Noise-based blink detection (Hershman, Henik & Cohen, 2018)</i>
---------------------	--

Description

Detects blinks in a per-sample pupil time series. Starts from the tracker's missing-data sentinel and refines each missing-sample run's onset and offset by walking outward on the smoothed first-difference signal – picking up the eyelid descent before the missing samples and the recovery rise after. Adjacent blinks separated by less than **gap_ms** are merged.

Usage

```
detect_blinks_noise(
  pupil,
  hz,
  smooth_ms = 10,
  gap_ms = 100,
  missing = c(0, NA)
)
```

Arguments

pupil	Numeric vector of pupil samples (one trial, one eye). Must be in time order at the rate given by hz .
hz	Sampling rate in Hz.
smooth_ms	Centered moving-average window applied before onset / offset refinement. Default 10 ms.
gap_ms	Adjacent blinks closer than this (in ms) are merged. Default 100 ms.
missing	Values in pupil that should be treated as missing samples. Default <code>c(0, NA)</code> – works for Tobii (0) and EyeLink-style (NA) exports without configuration.

Details

Tracker-agnostic missing handling: by default treats both NA and 0 as missing (Tobii uses 0; SR Research / EyeLink exports often use NA). Override via **missing**.

Composes with dplyr exactly like `gazeR::extend_blinks()`:

```
df |>
  group_by(subject, trial) |>
  mutate(blink = detect_blinks_noise(pupil, hz = 250))
```

If you want pupil set to NA inside blinks (gazeR-style), chain one more line:

```
... |>
  mutate(pupil = ifelse(blink > 0, NA_real_, pupil))
```

Adapted to R with Dr William Paul Boyce (paul.boyce@ntu.edu.sg); indexing bug in the original port – matching on index instead of time – was diagnosed by Dr Jason Geller.

Value

An integer vector the same length as **pupil**: 0 outside detected blinks, 1, 2, ... inside each blink. When used inside `dplyr::mutate()` under a `group_by()`, blink ids restart at 1 in each group.

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

Description

This function will combine gaze samples into time bins.

Usage

```
downsample_gaze(
  dataframe,
  bin.length = 50,
  timevar = "time",
  aggvars = c("subject", "condition", "target", "trial", "object", "timebins"),
  type = "gaze",
  pupil = "baselinecorrectedp"
)
```

Arguments

<code>dataframe</code>	DataFrame containing gaze data.
<code>bin.length</code>	Length of time bins.
<code>timevar</code>	Column name representing time.
<code>aggvars</code>	Vector of variable names to group by for aggregation.
<code>type</code>	Type of data, either "pupil" for pupil data or "gaze" for gaze data.
<code>pupil</code>	column name representing pupil column

Value

Downsampled DataFrame.

<code>downsample_pupil</code>	<i>Downsample pupil data</i>
-------------------------------	------------------------------

Description

This function will reduce the sampling frequency

Usage

```
downsample_pupil(
  dataframe,
  bin.length = NULL,
  timevar = "time",
  aggvars = c("subject", "script", "timebins")
)
```

Arguments

<code>dataframe</code>	dataframe
<code>bin.length</code>	Length of bins to average
<code>timevar</code>	name of time variable
<code>aggvars</code>	vector of variable names to group by for aggregation

<code>dynamic_pupil</code>	<i>calc dynamic pupil range (Ayasse et al., 2017; Piquado et al., 2010)) must have min pupil and max pupil size merged into df to work</i>
----------------------------	--

Description

calc dynamic pupil range (Ayasse et al., 2017; Piquado et al., 2010)) must have min pupil and max pupil size merged into df to work

Usage

```
dynamic_pupil(pupil_col, dmin = NA, dmax = NA)
```

Arguments

<code>pupil_col</code>	pupil col
<code>dmin</code>	mean pupil size during the ‘bright’ condition (i.e., minimum pupil size)
<code>dmax</code>	mean pupil size during the ‘dark’ condition (i.e., max pupil size)

Value

return dynamic pupil size scaled to P

<code>extend_blinks</code>	<i>Extends blinks if not already done in SR Data Viewer</i>
----------------------------	---

Description

Extends blinks if not already done in SR Data Viewer

Usage

```
extend_blinks(values, fillback = 5, fillforward = 5, hz = NA)
```

Arguments

<code>values</code>	column with pupil values
<code>fillback</code>	NA extend backward (in ms)
<code>fillforward</code>	NA extend forward (in ms)
<code>hz</code>	sampling rate of eye tracker

Value

vector containing extended blinks

<code>find_messages_asc</code>	<i>Take asc files and return important events Collect trial variables for your experiment. #' This code was written by Dr. Holger Mitterer</i>
--------------------------------	--

Description

Take asc files and return important events Collect trial variables for your experiment. #' This code was written by Dr. Holger Mitterer

Usage

```
find_messages_asc(
  dirList,
  homeDir = "./",
  time2extract = NULL,
  msg2extract = NULL,
  vars2extract = NULL,
  nOfTrials = -1,
  binocular = FALSE
)
```

Arguments

<code>dirList</code>	dir of edf files
<code>homeDir</code>	dir of files
<code>time2extract</code>	a list of strings that match message texts for which only the time should be retained.
<code>msg2extract</code>	A list of messages to extract fully based on the first word of that message output will be string variable
<code>vars2extract</code>	A list of Trial variables used by Experiment Builder that should be extracted value only, with column names set by the variable name
<code>nOfTrials</code>	number of trials to extract; -1 for all
<code>binocular</code>	set to TRUE when the recording is binocular. Message extraction is eye-independent, so this only keeps the interface symmetric with parse_asc ; the extracted output is identical either way.

<code>find_messages_edf</code>	<i>Take EDF files and return a tibble with important behavioral events</i>
--------------------------------	--

Description

Collect trial variables for your experiment.

Usage

```
find_messages_edf(file_list, varnames, patterns, output_dir)
```

Arguments

<code>file_list</code>	list of edf files
<code>varnames</code>	a vector of variable names from experiment c("TRIALID", "ACCURACY")
<code>patterns</code>	a vector of patterns to look for ("TRIALID", "!V TRIAL_VAR script")
<code>output_dir</code>	where you want files to be stored

<code>FunctTheme</code>	<i>Fixation of semantically related objects during spoken word comprehension</i>
-------------------------	--

Description

Fixation time course data from a spoken word-to-picture matching study with semantic distractors in the display.

Usage

```
FunctTheme
```

Format

A data frame with 1530 rows and 5 variables:

Subject subject ID, factor

Time time (in ms) since word onset in 50ms bins

meanFix fixation proportion (subject mean): proportion of correct-response trials on which the object was fixated

Condition semantic relatedness condition: Function (e.g., broom - sponge) or Thematic (e.g., broom - dustpan)

Object object type: Target, semantically related Competitor, or Unrelated

Source

Kalenine et al. (2012, J. Exp. Psychol. Learn. Mem. Cog., 38:5, 1274-1295) http://www.danmirman.org/pdfs/Kalenine_etal2012.pdf

FunctThemePts	<i>Fixation of semantically related objects during spoken word comprehension by participants with left hemisphere stroke</i>
----------------------	--

Description

Fixation time course data for participants with left hemisphere stroke from a spoken word-to-picture matching study with semantic distractors in the display.

Usage

FunctThemePts

Format

A data frame with 8199 rows and 8 variables:

Subject subject ID, factor

Condition semantic relatedness condition: Function (e.g., broom - sponge) or Thematic (e.g., broom - dustpan)

Object object type: Target, semantically related Competitor, or Unrelated

Time time (in ms) relative to word onset in 50ms bins; negative numbers indicate preview period (visual display before word onset)

meanFix fixation proportion (subject mean): proportion of correct-response trials on which the object was fixated

sumFix fixation sum: number of correct-response trials on which the object was fixated (i.e., numerator for meanFix)

N number of correct-response trials (i.e., denominator for meanFix)

Source

Kalenine, Mirman, & Buxbaum (2012, Front. Hum. Neurosci., 6:106, 1-12) <http://journal.frontiersin.org/Journal/10.3389/fnhum.2012.00106>

gazer

Tools for Processing Eye Tracking Data

Description

Functions for reading and pre-processing eye tracking data.

Author(s)

Maintainer: Jason Geller <drjasongeller@gmail.com>

Authors:

- Jason Geller <drjasongeller@gmail.com>
- Dan Mirman <dan@danmirman.org>
- Tristan Mahr <tristan.mahr@wisc.edu>
- Matt Winn <mwinn83@gmail.com>
- Holger Mitterer <holger.mitterer@um.edu.mt>

See Also

Useful links:

- <http://github.com/dmirman/gazer>
- Report bugs at <http://github.com/dmirman/gazer/issues>

`generate_edf2asc_bat` *Takes edf files and converts them to asc This code was written by Dr. Holger Mitterer*

Description

Takes edf files and converts them to asc This code was written by Dr. Holger Mitterer

Usage

```
generate_edf2asc_bat(homeDir, subDirs, deleteExisting = F)
```

Arguments

<code>homeDir</code>	dir of files
<code>subDirs</code>	end location of files
<code>deleteExisting</code>	overwrite file - set to false

get_gaze_diagnostics	<i>Calculate gaze track diagnostics</i>
----------------------	---

Description

Use fixation data to calculate two diagnostics of gaze track quality: non-fixation time and out-of-bounds proportion.

Usage

```
get_gaze_diagnostics(gaze, screen_size = c(1024, 768))
```

Arguments

gaze	fixation list from fixation report
screen_size	size of the screen in pixels. Defaults to c(1024, 768)

Value

data frame containing gaze diagnostics

get_pvalues	<i>Get p-values for a multilevel regression model</i>
-------------	---

Description

Takes a model fit with lmer and (optionally) an estimation method, returns a data frame containing fixed effect coefficient estimates, SE, t-values, and p-values. Can use normal approximation, Kenward-Roger approximation for degrees of freedom, or both. If Kenward-Roger approximation is used, result also contains approximated degrees of freedom.

Usage

```
get_pvalues(model, method = "normal")
```

Arguments

model	a merMod model object fit by lmer
method	p-value estimation method, defaults to "normal", can also be "KR" or "all"

Value

data frame containing fixed effect estimates, SE, t-values, (degrees of freedom if appropriate) and p-values.

Examples

```
require(lme4)
model <- lmer(Reaction ~ Days + (Days | Subject), sleepstudy)
get_pvalues(model)
get_pvalues(model, "KR")
```

get_ranef
Get random effects for individual effect size quantification

Description

Takes a model and name of a random effect. Returns a data frame containing those random effect estimates arranged and ready for quantifying individual effect sizes.

Usage

```
get_ranef(model = NULL, ranef_name = NULL)
```

Arguments

model	(g)lmerMod object
ranef_name	string name of random effect (right of pipe in (g)lmer call)

Value

Returns a data frame containing the random effect estimates

Examples

```
## Not run:
# FunctTheme example model
FunctTheme.gca <- code_poly(FunctTheme, predictor = "Time", poly.order = 2, draw.poly = FALSE)
m.FT <- lme4::lmer(meanFix ~ (poly1 + poly2) * Object * Condition +
  (poly1 + poly2 | Subject) + (poly1 + poly2 | Subject:Object:Condition),
  data = FunctTheme.gca, REML = FALSE)
re <- get_ranef(m.FT, "Subject:Object:Condition")

## End(Not run)
```

<code>get_trackloss</code>	<i>Calculate trackloss and remove trials and subjects with excessive trackloss</i>
----------------------------	--

Description

Calculate trackloss and remove trials and subjects with excessive trackloss

Usage

```
get_trackloss(gaze, screen_size = c(1024, 768), missingthresh = 0.2)
```

Arguments

<code>gaze</code>	fixation list from fixation report
<code>screen_size</code>	size of the screen in pixels. Defaults to c(1024, 768)
<code>missingthresh</code>	threshold to throw out trials and subjs

Value

data frame containing gaze df with trackloss by subjs and trial. Trials and Subjs with excessive trackloss removed

<code>hanning_filter</code>	<i>Performs hanning smoothing on pupil data</i>
-----------------------------	---

Description

Performs hanning smoothing on pupil data

Usage

```
hanning_filter(pupil, degree = 11)
```

Arguments

<code>pupil</code>	column with pupil values
<code>degree</code>	default of 11.

Value

vector containing hanning filtered pupil data

<code>interpolate_pupil</code>	<i>Turns pupil sizes with blinks NA and linearly interpolates values</i>
--------------------------------	--

Description

Turns pupil sizes with blinks NA and linearly interpolates values

Usage

```
interpolate_pupil(
  datafile,
  pupil,
  subject,
  trial,
  extendblinks = FALSE,
  maxgap = Inf,
  type = c("linear", "cubic"),
  hz = NA
)
```

Arguments

<code>datafile</code>	data frame.
<code>extendblinks</code>	blinks already extended in data frame
<code>maxgap</code>	max number of NAs to interpolate.
<code>type</code>	type of interpolation (linear or cubic)
<code>hz</code>	recording frequency of ET

Value

data frame containing interpolated data

<code>legend_positioning</code>	<i>Set legend position</i>
---------------------------------	----------------------------

Description

A wrapper function to make setting legend position easier when using ggplot.

Usage

```
legend_positioning(pos, edge_color = "black", legend_fill = "white")
```

Arguments

<code>pos</code>	Legend position as a two-element vector
<code>edge_color</code>	Legend edge color, defaults to black
<code>legend_fill</code>	Legend fill color, defaults to white

Examples

```
library(ggplot2)
p <- ggplot(TargetFix, aes(Time, meanFix, color = Condition)) + stat_summary(fun = mean, geom = "line")
p + legend_positioning(c(0,1)) #put legend in upper left corner
p + legend_positioning(c(1,0)) #put legend in lower right corner
```

<code>lowpass_filter</code>	<i>This function performs low pass filtering</i>
-----------------------------	--

Description

This function performs low pass filtering

Usage

```
lowpass_filter(pd, order, filter_to_use, samplerate = NA)
```

Arguments

<code>pd</code>	pupil column.
<code>order</code>	order butter filter
<code>filter_to_use</code>	hz to low pass filter
<code>samplerate</code>	sampling rate of the eye tracker in Hz

Value

vector containing low pass filtered pupil data

make_gazer

Prepare data for pre-processing in Gazer

Description

This will make sure your data is in the right format for processing. This package is designed to deal with data as 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.

`x`, `y`, and `pupil` may each name a single column or several. A single column is renamed to gazeR's canonical name (`x`, `y`, or `pupil`); when you pass more than one column (e.g. left and right eye) the original column names are kept as-is. Either way the selected columns are moved to the front of the data frame and every other column is kept, in its original order, after them.

Usage

```
make_gazer(
  data,
  subject = "subject",
  trial = "trial",
  time = "time",
  x = "x",
  y = "y",
  pupil = NULL
)
```

Arguments

<code>data</code>	a raw, long form dataframe organised by subject, trial, and time.
<code>subject</code>	column name indicating subject ID
<code>trial</code>	column name indicating trial ID. This should be unique for participants
<code>time</code>	column name indicating time column (should be numeric)
<code>x</code>	gaze x column(s). One name is renamed to <code>x</code> ; several (e.g. <code>c("x_left", "x_right")</code>) keep their names. Use <code>NULL</code> if there is no gaze.
<code>y</code>	gaze y column(s); see <code>x</code> .
<code>pupil</code>	pupil column(s). One name is renamed to <code>pupil</code> ; several (e.g. <code>c("pupil_left", "pupil_right")</code>) keep their names. <code>NULL</code> (default) if none.

Value

A dataframe ready to use in `gazer`

<code>merge_asc_files</code>	<i>Take asc files and extract relevant data Merges eye data (from <code>parse_asc</code>) with message data (from <code>find_messages_asc</code>) This code was written by Dr. Holger Mitterer</i>
------------------------------	--

Description

Take asc files and extract relevant data Merges eye data (from `parse_asc`) with message data (from `find_messages_asc`) This code was written by Dr. Holger Mitterer

Usage

```
merge_asc_files(dirList, homeDir = "./", dataType = c("fixation", "sample"))
```

Arguments

<code>dirList</code>	list if asc files
<code>homeDir</code>	directory of edf files
<code>dataType</code>	type of eye data being merged. Both left-join message data onto eye data, so every eye row is kept (one row per eye row, like <code>parse_asc</code>). "fixation" (default) joins by trial only, so every message row for a trial is attached to every eye row for that trial. "sample" joins by trial and time, so message columns are filled in only where a message landed on that exact timestamp and NA elsewhere (requires <code>_messages.csv</code> to have a <code>time</code> column, i.e. <code>find_messages_asc</code> was called with <code>time2extract</code>).

<code>merge_gazer_files</code>	<i>Merge eye files from a ufolder if filetype is from sr renames column variables and makes them lower case Places time in ms if edf files just reads them in</i>
--------------------------------	---

Description

Merge eye files from a ufolder if filetype is from sr renames column variables and makes them lower case Places time in ms if edf files just reads them in

Usage

```
merge_gazer_files(
  file_list,
  blink_colname = "blink",
  pupil_colname = "pupil",
  filetype = "sr"
)
```

Arguments

`file_list` path to .xls files

`blink_colname` name of your blink colname: AVERAGE_IN_BLINK, LEFT_, #RIGHT

`pupil_colname` name of your pupil colname: AVERAGE_IN_Pupil, LEFT_, #RIGHT

`filetype` if reading raw edf file use 'edf' if using sr files use type 'sr'

Value

data frame containing pupil data

<code>merge_pupil</code>	<i>Merge pupil files from a folder Renames column variables and makes them lower case Places time in ms</i>
--------------------------	---

Description

Merge pupil files from a folder Renames column variables and makes them lower case Places time in ms

Usage

```
merge_pupil(file_list, blink_colname = "", pupil_colname = "", filetype = "sr")
```

Arguments

`file_list` path to .xls files

`blink_colname` name of your blink colname: AVERAGE_IN_BLINK, LEFT_, #RIGHT

`pupil_colname` name of your pupil colname: AVERAGE_IN_Pupil, LEFT_, #RIGHT

`filetype` if reading raw edf file use 'edf' if using sr files use type 'sr'

Value

data frame containing pupil data

merge_tobii_titta	<i>Merge paired Tobii Titta CSVs into a gazeR-ready, per-trial dataframe</i>
-------------------	--

Description

Loads gaze samples and event messages from two parallel folders, attaches each message to the nearest gaze sample by `system_time_stamp` (as-of merge), assigns trial numbers from onset / offset markers, and rebases time to be trial-relative. The result is one row per gaze sample, with event-marker columns set sparsely on the samples that absorbed an event.

Usage

```
merge_tobii_titta(
  gaze,
  msg,
  onset_prefix = "onset_",
  offset_prefix = "offset_"
)
```

Arguments

<code>gaze</code>	A directory of gaze CSVs, or a character vector of gaze file paths.
<code>msg</code>	A directory of message CSVs, or a character vector of msg file paths. Paired with <code>gaze</code> by basename.
<code>onset_prefix</code>	Message prefix marking trial onsets. Required. Default "onset_".
<code>offset_prefix</code>	Message prefix marking trial offsets. Pass NULL for onset-only recordings. Default "offset_".

Details

`merge_tobii_titta()` reads the split CSVs that the current Titta build emits.

Inputs:

Files are paired by basename. A subject's gaze and msg file must share the same filename minus extension, e.g. `gaze/sub01.csv` <-> `msg/sub01.csv`. The basename (without extension) becomes `subject`.

Trial assignment:

Trials are identified by message prefixes. Three modes are supported, picked automatically from the data:

- **Bracketed** (default): `onset_<label>` starts a trial, `offset_<label>` ends it. Samples between an offset and the next onset are dropped.
- **Onset-only**: pass `offset_prefix = NULL`, or simply have no rows matching the offset prefix. Each onset runs until the next onset.

- **Custom prefixes:** override `onset_prefix` / `offset_prefix` for any other convention (`start_<n>`, `startfix_<n>`, etc.).

The text following the onset prefix is kept as `event_label` and broadcast across every sample in that trial – it can be a stim filename, a trial number, a condition tag, whatever your protocol uses.

Nearest-sample merge:

Because gaze is sampled on a fixed cadence (30 / 120 / 300 Hz on Tobii trackers) but messages fire asynchronously, a message timestamp almost never exactly equals a sample timestamp. Each message is therefore merged onto the gaze sample whose timestamp is closest, found vectorized via `findInterval()`. `match_offset_us` (`matched sample ts - msg ts`) is exposed for diagnostics; on a 30 Hz tracker this is at most ~16 ms (half the inter-sample gap), and well below 1 ms on a 300 Hz tracker.

Pupil / openness:

Raw `left_pupil_diameter`, `right_pupil_diameter`, `left_eye_openness_diameter`, and `right_eye_openness_diameter` (if present) pass through unchanged. Average them downstream if you want a single signal – `gazeR::compute_monocular_mean()` or `rowMeans(..., na.rm = TRUE)`.

Output time:

`time` is in milliseconds relative to each trial's first sample (the one the onset message merged onto), matching `gazeR`'s per-trial convention.

Value

A tibble with one row per gaze sample inside a trial window. Leading columns: `subject`, `system_time_stamp`, `time`, `msg`, `message`, `event`, `event_label`, `match_offset_us`, `trial`. The raw Tobii sample columns (gaze x/y, pupil diameter, eye openness, validity flags, etc.) follow.

Examples

```
## Not run:
out <- merge_tobii_titta(
  gaze = "data/raw/gaze",
  msg = "data/raw/msg"
)

# Onset-only recording with custom prefix
out <- merge_tobii_titta(
  gaze = "data/raw/gaze",
  msg = "data/raw/msg",
  onset_prefix = "startfix_",
  offset_prefix = NULL
)

## End(Not run)
```

`moving_average_pupil` *Moving average function*

Description

Creates a moving average using a specified averageing window if n is 5, then it uses the target value, 2 preceding and 2 following

Usage

```
moving_average_pupil(x, n = 5, centered = TRUE)
```

Arguments

x	data
n	sed for moving window
centered	whether moving average window should be centered (TRUE) or trailing (FALSE)

Value

return sum divided by count

`NamingRecovery` *Picture naming during aphasia treatment*

Description

Picture naming data from individuals with aphasia undergoing a treatment trial. Each participant was tested on a 175-item picture naming test (Philadelphia Naming Test) multiple times over the course of the treatment and responses were coded.

Usage

```
NamingRecovery
```

Format

A data frame with 115 rows and 9 variables:

SubjectID subject ID, factor

Diagnosis participant's aphasia subtype: Anomic, Conduction, or Wernicke's aphasia

TestTime test iteration, 0 is the baseline at the start of treatment

Correct proportion correct naming responses

Semantic.error proportion semantic errors (cat - "dog")

Mixed.error proportion mixed errors (cat - "rat")

Formal.error proportion formal errors (cat - "mat")

Unrelated.error proportion unrelated errors (cat - "log")

Nonword.error proportion nonword errors (cat - "lat")

Source

Moss Aphasia Psycholinguistics Database <mappd.org>

onset_pupil	<i>Zero Time Index to Event Onset</i>
-------------	---------------------------------------

Description

Centers the sample index or time stamp of your time series to a reference marker, such that 0 always coincides with the onset of a particular event.

Usage

```
onset_pupil(time, marker, event)
```

Arguments

time	Numeric vector of sample indices or time stamps corresponding to a single time series.
marker	Vector containing markers of trial events.
event	Label in marker that indicates the onset event to which the time series should be zeroed. If this label is a character string, it must be enclosed in quotation marks.

Value

Numeric vector that zeroes **time** to the trial onset. In the event that no onset marker is found, a warning is issued and a vector of NAs is returned.

<code>parse_asc</code>	<i>Take asc files and extract relevant data Does not merge data with messages for fixation data, obtains sample report and puts time in ms This code was written by Dr. Holger Mitterer</i>
------------------------	---

Description

Take asc files and extract relevant data Does not merge data with messages for fixation data, obtains sample report and puts time in ms This code was written by Dr. Holger Mitterer

Usage

```
parse_asc(
  dirList,
  homeDir = "./",
  overwriteBlinks = FALSE,
  cutPreview = 0,
  binocular = FALSE
)
```

Arguments

<code>dirList</code>	list if asc files
<code>homeDir</code>	directory of edf files
<code>overwriteBlinks</code>	set to false
<code>cutPreview</code>	number of ms to cut from the start of each trial (default 0)
<code>binocular</code>	set to TRUE to return both left and right eye samples (x/y/pupil for each eye plus per-eye blink flags); FALSE (default) returns a single (monocular) eye.

<code>parse_edf</code>	<i>Take EDF files and return data that is in format usable for gazeR for pupil merges the sample data with the message data (merges on nearest values (millisecond diff between SR and this function) puts time in ms adds subject variable column cleans up the column names creates pupil column that is ambiguous as to whether you sampled from left eye, right eye, or both (takes the average) For fixation data, obtains sample report and puts time in ms</i>
------------------------	---

Description

Take EDF files and return data that is in format usable for gazeR for pupil merges the sample data with the message data (merges on nearest values (millisecond diff between SR and this function) puts time in ms adds subject variable column cleans up the column names creates pupil column that is ambiguous as to whether you sampled from left eye, right eye, or both (takes the average) For fixation data, obtains sample report and puts time in ms

Usage

```
parse_edf(file_list, output_dir, type = "pupil")
```

Arguments

<code>file_list</code>	directory to edf files
<code>type</code>	include whether you want to parse edf pupil data (pupil) or vwp (fixations)
<code>output.dir</code>	directory to save new cleaned files

<code>parse_edf_messages</code>	<i>Take EDF files and return message events to extract behav data creates pupil column that is ambiguous to whether you sampled from left eye, right eye, or both (takes the average)</i>
---------------------------------	---

Description

Take EDF files and return message events to extract behav data creates pupil column that is ambiguous to whether you sampled from left eye, right eye, or both (takes the average)

Usage

```
parse_edf_messages(file_list, output.dir)
```

Arguments

<code>file_list</code>	directory to edf files
<code>output.dir</code>	directory to save new cleaned files

<code>parse_pl</code>	<i>Read and align a single Pupil Labs subject, define trials, and reset time</i>
-----------------------	--

Description

Reads `gaze.csv`, `3d_eye_states.csv`, `events.csv`, and `gaze_surface.csv` from a subject folder, attaches the nearest (non-recording) event to gaze rows (within a lag), defines trial boundaries, removes blank trials, and resets `time` to 0 within each trial.

Usage

```
parse_pl(
  subject_dir,
  subject_id = basename(normalizePath(subject_dir)),
  aoi = FALSE,
  max_event_lag_ms = 100,
  start_mode = c("any", "exact"),
  start_messages = NULL
)
```

Arguments

<code>subject_dir</code>	Path to a single subject directory containing Pupil Labs CSVs.
<code>subject_id</code>	Character ID for the subject. Defaults to folder name.
<code>aoi</code>	Logical; if TRUE, use surface (AOI) stream as well.
<code>max_event_lag_ms</code>	Maximum absolute difference (ms) allowed when snapping an event to the nearest gaze row. Defaults to 100.
<code>start_mode</code>	Trial start rule: "any" = any non-empty, non-recording message; "exact" = only messages in <code>start_messages</code> (normalized) start a trial.
<code>start_messages</code>	Character vector of trial-start messages when <code>start_mode = "exact"</code> . Case/whitespace/punctuation ignored after normalization.

Value

Tibble: `subject`, `trial`, `time`, `x`, `y`, `pupil`, `blink`, `message`, `start_message_first`.

```
process_all_subjects_PL
```

Process all Pupil Labs subjects in a root directory

Description

Recursively finds immediate subfolders, runs `read_pl_subject` on each, writes per-subject CSVs and a combined CSV, and invisibly returns the combined tibble.

Usage

```
process_all_subjects_PL(
  root_dir,
  output_dir = file.path(root_dir, "processed"),
  ...
)
```

Arguments

<code>root_dir</code>	Directory containing one folder per subject.
<code>output_dir</code>	Directory for outputs. Defaults to <code>file.path(root_dir, "processed")</code> .
<code>...</code>	Extra arguments forwarded to <code>read_pl_subject</code> (e.g., <code>start_mode</code> , <code>start_messages</code>).

Value

Invisibly returns a combined tibble of all processed subjects.

Examples

```
## Not run:
process_all_subjects("/path/Timeseries",
  start_mode = "exact",
  start_messages = c("trial-started-light", "start-dark"))

## End(Not run)
```

```
read_asc
```

Read an EyeLink .asc file into gazeR format

Description

A fast, single-file reader for EyeLink `.asc` files (convert `.edf` with SR Research's `edf2asc` first). The file is parsed in a few vectorised passes, so even large recordings take seconds. Unlike `parse_asc()` and `find_messages_asc()`, it needs no folder-per-subject layout and writes nothing to disk: it returns one long data frame, one row per sample, ready for the pupil functions (`extend_blinks()`, `smooth_interpolate_pupil()`, `baseline_correction_pupil_msg()`, ...), or the shiny gazeR app (<https://github.com/jgeller112/shiny-gazeR>).

Usage

```
read_asc(
  file,
  eye = c("average", "left", "right", "both"),
  subject = NULL,
  trial_start = "TRIALID",
  trial_end = "TRIAL_RESULT",
  keep_messages = FALSE
)
```

Arguments

<code>file</code>	path to an <code>.asc</code> file (may be gzipped).
<code>eye</code>	which eye to use for binocular recordings: "average", "left", "right" or "both". Ignored for monocular recordings.
<code>subject</code>	subject label. Defaults to the file name without extension.
<code>trial_start, trial_end</code>	message text (first word) marking the start and end of a trial.
<code>keep_messages</code>	if TRUE, the full message table is attached as attribute "messages" (columns <code>trial</code> , <code>time</code> , <code>tracker_time</code> , <code>text</code>).

Details

What it does:

- **Trials** run from each `TRIALID` message to the matching `TRIAL_RESULT` message (both configurable). Samples outside a trial are dropped. `time` is in ms relative to the start of recording in that trial (the `!MODE RECORD` message, or `TRIALID` if there is none), matching `parse_asc()`. The raw tracker clock is kept in `tracker_time`.
- **Messages** are attached to the first sample at or after the message time, in a `message` column. Experiment Builder offsets (`MSG <time> <offset> <text>`) are applied (`time - offset`). Tracker housekeeping messages (`RECCFG`, `GAZE_COORDS`, `!V DRAW_LIST`, ...) are left out of the sample data but kept in the full message table.
- **Trial variables** written by Experiment Builder (`!V TRIAL_VAR name value`) become one column per variable, with types guessed by `utils::type.convert()`. These are usually your condition columns.
- **Blinks** come from the tracker's `SBLINK`/`EBLINK` events (`blink` is 1 during a blink). Pupil values of 0 and missing gaze (.) become `NA`.
- **Binocular data** are handled with `eye`: "average", "left", "right", or "both" (per-eye columns plus the average). With "average" and "both", `pupil`, `x` and `y` are the **mean of the two eyes**, computed only where both eyes are valid (otherwise `NA`), and `blink` is 1 if **either** eye is in a blink.

Value

A tibble with one row per sample:

- **subject**: participant label (file name without extension, or **subject**).
- **trial**: trial number (one per TRIALID).
- **time**: ms from the start of recording in the trial (!MODE RECORD).
- **pupil**: pupil size in tracker units (area or diameter, see **asc_info**); NA when missing. For **eye** = "average"/"both", the mean of both eyes where both are valid.
- **x, y**: gaze position in screen pixels; NA when missing. Averaged like **pupil**.
- **blink**: 1 during a blink, 0 otherwise. For **eye** = "average"/"both", 1 if either eye is in a blink.
- **message**: event message(s) attached to this sample, otherwise NA.
- **tracker_time**: the tracker's own clock (ms).
- With **eye** = "both": **pupil_left**, **pupil_right**, **x_left**, **y_left**, **x_right**, **y_right**, **blink_left**, **blink_right**, each for one eye.
- One column per Experiment Builder trial variable.

Attribute "**asc_info**" holds the sampling rate (**hz**), recorded eyes, pupil type (area or diameter) and the eye used.

Examples

```
## Not run:
d <- read_asc("data/PN001.asc", eye = "left")
attr(d, "asc_info")$hz

## End(Not run)
```

read_bids_eyetrack	<i>Read BIDS eye-tracking data into gazeR format</i>
--------------------	--

Description

Reads eye-tracking recordings stored in the BIDS format for physiological recordings (BEP020): ***_recording-<label>_physio.tsv.gz** files (no header) with JSON sidecars describing their columns, plus optional ***_physioevents.tsv.gz** files with messages and events. Returns one long data frame, one row per sample, in the same layout as [read_asc\(\)](#).

Usage

```
read_bids_eyetrack(
  path,
  eye = c("average", "left", "right", "both"),
  trial_start = NULL
)
```

Arguments

<code>path</code>	a BIDS dataset folder (searched recursively), or a vector of <code>*_physio.tsv.gz</code> file paths.
<code>eye</code>	which eye to use for binocular data: "average", "left", "right" or "both".
<code>trial_start</code>	optional character vector of messages that start a trial (matched against the start of each message).

Details

- **Participants:** subject is built from the `sub-`, `ses-`, `task-` and `run-` entities of each file name (e.g. `sub-01_task-rest_run-1`).
- **Columns:** `timestamp`, `x_coordinate`, `y_coordinate` and `pupil_size` become `time`, `x`, `y` and `pupil`; other columns are kept. Timestamps are converted to ms from the Units of `timestamp` in the sidecar (`s`, `ms`, `us` or `ns`, so Pupil Labs and Tobii conversions work too), or, without units, from the step between samples and `SamplingFrequency`.
- **Eyes:** recordings of the left and right eye (e.g. `recording-eye1` and `recording-eye2`, identified by `RecordedEye` in the sidecar) are merged by timestamp. `eye` works as in `read_asc()`: "average" (mean of both eyes where both are valid; `blink` is 1 if either eye blinks), "left", "right", or "both" (per-eye columns plus the average).
- **Events:** rows of `*_physioevents.tsv.gz` with a `message` are attached to the first sample at or after their onset; rows with `trial_type == "blink"` (or `blink == 1`) mark `blink` for their duration.
- **Trials:** a BIDS recording is continuous, so by default each run is one trial. With `trial_start`, a new trial starts at each message matching one of these strings, and time is ms from that message; samples before the first one are dropped.

Value

A tibble with `subject`, `trial`, `time`, `pupil`, `x`, `y`, `blink`, `message`, `tracker_time`, `sample_rate`, `eyes_recorded` and any extra columns. Attribute "bids_info" has one row per subject with `sample_rate`, `eyes_recorded`, `manufacturer` and `pupil_units` from the sidecars, and `blink_events` (whether any blink events were found).

Examples

```
## Not run:
d <- read_bids_eyetrack("my_bids_dataset", eye = "left")
d <- read_bids_eyetrack("my_bids_dataset", trial_start = "TRIALID")

## End(Not run)
```

`read_fixation_report` *Read EyeDataViewer fixation report*

Description

Reads in a fixation report created by EyeLink's EyeDataViewer application

Usage

```
read_fixation_report(
  filename,
  screen_size = c(1024, 768),
  name_changes = c(Subject = "RECORDING_SESSION_LABEL", ACC = "StimSlide.ACC", RT =
    "StimSlide.RT", TargetLoc = "CorrectPort"),
  plot_fix_scatter = TRUE,
  pages = 1
)
```

Arguments

<code>filename</code>	string containing full filename of fixation report
<code>screen_size</code>	size of the screen in pixels. Used to compute track quality diagnostics. Defaults to <code>c(1024, 768)</code>
<code>name_changes</code>	set of variable name pairs for improving variable names. Should be formatted as named character vector with new names as values and old names as names; see <code>plyr::rename()</code>
<code>plot_fix_scatter</code>	logical indicating whether a scatterplot of fixations should be created
<code>page</code>	outputs a plots 4 x 3 and displays the first page. Change number to see other pages if you have more Ps

Value

gaze data frame containing fixation report

`resample_gaze` *Resample gaze and pupil data to a target sampling rate*

Description

Brings each participant's data to `target_hz`, working within each subject and trial:

Usage

```
resample_gaze(
  x,
  target_hz,
  hz = NULL,
  pupil_cols = c("pupil", "pupil_left", "pupil_right"),
  gaze_cols = c("x", "y", "x_left", "y_left", "x_right", "y_right"),
  blink_cols = c("blink", "blink_left", "blink_right"),
  event_cols = "message",
  interpolate = TRUE
)
```

Arguments

x	data frame with subject , trial and time (ms) columns.
target_hz	target sampling rate in Hz.
hz	recorded sampling rate: a single number, a named vector (names = subjects), or NULL (default) to estimate each subject's rate from the median time step.
pupil_cols, gaze_cols	pupil and gaze columns. Columns not in x are ignored.
blink_cols	blink-flag columns (1 = in a blink). Columns not in x are ignored.
event_cols	event/message columns. Columns not in x are ignored.
interpolate	when upsampling, interpolate the new samples' pupil and gaze values (default TRUE); if FALSE they are left NA.

Details

- **Downsampling** (recorded rate above **target_hz**): consecutive samples are averaged into bins of $1000 / \text{target_hz}$ ms, which avoids aliasing. Pupil and gaze values are the bin mean, and are NA if any sample in the bin is missing (pupil values ≤ 0 count as missing). Blink columns are 1 if any sample in the bin was in a blink. Event columns keep every message in the bin (joined with "; "). Other columns take the bin's first value. **time** becomes the start of the bin.
- **Upsampling** (recorded rate below **target_hz**): new samples are added with `upsample_gaze()`. With **interpolate** = TRUE, their pupil and gaze values are linearly interpolated from the recorded samples either side, and left NA if either neighbour is missing (e.g. during a blink).
- **Same rate**: each trial is put on an evenly spaced grid, inserting any samples the tracker dropped as NA (never interpolated), ready for the usual blink and interpolation steps.

Inserted samples are flagged in a **resampled** column. Upsampling and the uniform grid need a whole-millisecond step (e.g. 1000, 500, 250 Hz).

Value

x at **target_hz**, sorted by subject, trial and time, with a logical **resampled** column (TRUE for samples added by upsampling or gap filling).

See Also

`upsample_gaze()`, `downsample_pupil()` and `downsample_gaze()` (which average into analysis time bins rather than resample).

Examples

```
x <- data.frame(subject = "s1", trial = 1, time = 0:9,
                pupil = c(1000:1003, NA, 1005:1009), message = NA)
x$message[3] <- "stim"
resample_gaze(x, target_hz = 500) # 1000 -> 500 Hz
```

`smooth_interpolate_pupil`

Smooths the pupil trace before linear or cubic interpolation.

Description

Smooths the pupil trace before linear or cubic interpolation.

Usage

```
smooth_interpolate_pupil(
  datafile,
  pupil = "pupil",
  extendpupil = "extendpupil",
  extendblinks = FALSE,
  step.first = "smooth",
  filter = "moving",
  maxgap = Inf,
  type = NULL,
  hz = NA,
  n = NA
)
```

Arguments

<code>datafile</code>	data frame.
<code>pupil</code>	name of pupil column
<code>extendpupil</code>	name of pupil col that was extended
<code>extendblinks</code>	blinks already extended in data frame
<code>step.first</code>	whether you want to smooth/interpolate or interpolate/smooth
<code>filter</code>	use either moving and hann filter
<code>maxgap</code>	max number of NAs to interpolate.
<code>type</code>	type of interpolation (linear or cubic)
<code>hz</code>	recording frequency of ET
<code>n</code>	moving average window

Value

df containing interpolated data

<code>speed_pupil</code>	<i>Calculates max absolute dilation speed as suggested in Kret & Sjak-Shie (in press) calculates the median absolute deviation (MAD) and sets the threshold to filter on.</i>
--------------------------	---

Description

Calculates max absolute dilation speed as suggested in Kret & Sjak-Shie (in press) calculates the median absolute deviation (MAD) and sets the threshold to filter on.

Usage

```
speed_pupil(pup, time)
```

Arguments

<code>pup</code>	column listing pupil sizes
<code>time</code>	column in data file listing time

Value

data with pupil speeds for each time point

<code>TargetFix</code>	<i>Target fixation data from a typical "visual world paradigm" experiment</i>
------------------------	---

Description

Time course of target object fixation during a spoken word-to-picture matching (i.e., "visual world paradigm").

Usage

```
TargetFix
```

Format

A data frame with 300 rows and 7 variables

Subject subject ID, factor

Time time (in ms) relative to word onset in 50ms bins, 300-1000

timeBin rescaling of Time variable for easier model-fitting, 1-15

Condition word frequency condition: High (e.g., torch, frog) or Low (e.g., class, horse)

meanFix fixation proportion (subject mean): proportion of correct-response trials on which the object was fixated

sumFix fixation sum: number of correct-response trials on which the object was fixated (i.e., numerator for meanFix)

N number of correct-response trials (i.e., denominator for meanFix)

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,
  event_cols = c("message")
)
```

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.
event_cols	Character vector of event/message columns that should stay empty on inserted rows (default "message"). Columns not present are ignored.

Details

Original rows are returned unchanged. Each inserted row takes its other columns (e.g. condition, trial variables) from the original sample before it, except `pupil_cols/gaze_cols` (left NA for interpolation) and `event_cols` (left NA, so each event message still appears once).

Value

A dataframe with up-sampled time points and an `up_sampled` column (TRUE for inserted rows, FALSE for original ones).

WordLearnEx

Effect of transitional probability on novel word learning

Description

Accuracy in 2-alternative forced-choice word learning experiment. On each trial, participants see two complex geometric shapes, hear a novel word (e.g., "pibu"), and have to guess which object corresponds to the word. They receive feedback and gradually learn which object goes with which word. There are 6 trials in each of the 10 training blocks. Transitional probability of the novel words (i.e., $p(\text{"bu"}|\text{"pi"})$) was experimentally manipulated (between-subjects) during an exposure phase immediately preceding the word learning phase.

Usage

WordLearnEx

Format

A data frame with 560 rows and 4 variables

Subject subject id, N=56

TP transitional probability of words, manipulated between-subjects

Block training block (1-10), each block consists of 6 trials

Accuracy proportion correct responses during the block, chance = 0.5

Source

Partial data from Mirman et al. (2008, Cognition, 108:1, 271-280). <http://www.danmirman.org/pdfs/MirmanMagnusonGrafEstesDixon2008.pdf>

Index

- * datasets
 - CohortRhyme, 10
 - cursive_agg_data, 13
 - FunctTheme, 18
 - FunctThemePts, 19
 - NamingRecovery, 31
 - TargetFix, 43
 - WordLearnEx, 45
- assign_aoi, 3
- baseline_correction_pupil, 4
- baseline_correction_pupil_msg, 5
- behave_pupil, 6
- binify_fixations, 6
- blink_detect, 7
- blink_velocity, 7
- calc_mad, 8
- code_poly, 9
- CohortRhyme, 10
- compare_groups, 10
- compute_monocular_mean, 11
- count_missing_pupil, 12
- Crawford_Howell, 12
- cursive_agg_data, 13
- detect_blinks_noise, 13
- downsample_gaze, 14
- downsample_gaze(), 42
- downsample_pupil, 15
- downsample_pupil(), 42
- dynamic_pupil, 16
- extend_blinks, 16
- find_messages_asc, 17
- find_messages_asc(), 36
- find_messages_edf, 18
- FunctTheme, 18
- FunctThemePts, 19
- gazer, 20
- gazer-package (*gazer*), 20
- generate_edf2asc_bat, 20
- get_gaze_diagnostics, 21
- get_pvalues, 21
- get_ranef, 22
- get_trackloss, 23
- hanning_filter, 23
- interpolate_pupil, 24
- legend_positioning, 24
- lowpass_filter, 25
- make_gazer, 26
- merge_asc_files, 27
- merge_gazer_files, 27
- merge_pupil, 28
- merge_tobii_titta, 29
- moving_average_pupil, 31
- NamingRecovery, 31
- onset_pupil, 32
- parse_asc, 17, 33
- parse_asc(), 36, 37
- parse_edf, 33
- parse_edf_messages, 34
- parse_pl, 35
- process_all_subjects_PL, 36
- read_asc, 36
- read_asc(), 38, 39
- read_bids_eyetrack, 38
- read_fixation_report, 40
- read_pl_subject, 36
- resample_gaze, 40
- smooth_interpolate_pupil, 42

`speed_pupil`, [43](#)

`TargetFix`, [43](#)

`upsample_gaze`, [44](#)

`upsample_gaze()`, [41](#), [42](#)

`utils::type.convert()`, [37](#)

`WordLearnEx`, [45](#)