THE UNIVERSITY *of* EDINBURGH

Edinburgh Research Explorer

GazeR

Citation for published version:

Geller, J, Winn, MB, Mahr, T & Mirman, D 2020, 'GazeR: A package for processing gaze position and pupil size data', *Behavior Research Methods*, vol. 52, pp. 2232-2255. <https://doi.org/10.3758/s13428-020-01374-8>

Digital Object Identifier (DOI):

[10.3758/s13428-020-01374-8](https://doi.org/10.3758/s13428-020-01374-8)

Link:

[Link to publication record in Edinburgh Research Explorer](#)

Document Version:

Peer reviewed version

Published In:

Behavior Research Methods

Publisher Rights Statement:

This is a post-peer-review, pre-copyedit version of an article published in Behavior Research Methods. The final authenticated version is available online at: <https://doi.org/10.3758/s13428-020-01374-8>

General rights

Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy

The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.

GazeR: A Package for Processing Gaze Position and Pupil Size Data

Jason Geller¹, Matthew B. Winn², Tristan Mahr³, & Daniel Mirman⁴

¹ University of Iowa

² University of Minnesota

³ University of Wisconsin-Madison

⁴ University of Edinburgh

Accepted for publication in Behavior Research Methods

Author note

1) Department of Psychological & Brain Sciences
The University of Iowa
Iowa City, IA 52242

2) Department of Speech-Language-Hearing Sciences
nn164 Pillsbury Dr. SE
Minneapolis, MN 555455

3) Waisman Center
1500 Highland Ave
Madison, WI 53705

4) Department of Psychology
7 George Sqaire
Edinburgh, UK, EH8 JZ

Correspondence concerning this article should be addressed to Jason Geller, Department of Psychological & Brain Sciences, University of Iowa. E-mail: jason-geller@uiowa.edu

Abstract

Eye-tracking is widely used throughout the scientific community, from vision science and psycholinguistics, to marketing and human-computer interaction. Surprisingly, there is little consistency and transparency in preprocessing steps, making replicability and reproducibility difficult. To increase replicability, reproducibility, and transparency, a package in R (a free and widely used statistical programming environment) called gazeR was created to read in and preprocess two types of data: gaze position and pupil size. For gaze position data, gazeR has functions for: reading in raw eye-tracking data, formatting it for analysis, converting from gaze coordinates to areas of interest, and binning and aggregating data. For data from pupillometry studies, the gazeR package has functions for: reading in and merging multiple raw pupil data files, removing observations with too much missing data, eliminating artifacts, blink identification and interpolation, subtractive baseline correction, and binning and aggregating data. The package is open-source and freely available for download and installation: <https://github.com/dmirman/gazer>. We provide step-by-step analyses of data from two tasks exemplifying the package's capabilities.

Keywords: eye-tracking, open science, pupillometry, visual world paradigm, R

Word count: 10,941

GazeR: A package for processing gaze position and pupil size data

Introduction

Recent advances in eye-tracking technology make it a highly powerful and relatively inexpensive tool to gather fine-grained measures of the temporal dynamics of cognitive processing. Because of this, a growing number of fields from vision science and psycholinguistics, to marketing and human-computer interaction, have adopted this methodology. Despite its growing presence, there is a lot of variability in how eye-tracking data are processed. With increased attention on replicability, reproducibility, and transparency, there is a need for a cross-platform, fully free implementation of standard practices in eye-tracking data processing. R (R Core Team 2018) is a widely-used, free, cross-platform, and open-source statistical programming language that provides the tools needed to meet those needs. In R, there are few established pipelines for handling pupil and fixation data from the visual world paradigm and pupillometry, especially contained in one package (see Tables 1 and 2). To meet this need, we created the gazeR package. The gazeR package is meant to facilitate the end-to-end handling of eye-tracking data within a single programming environment (R) – from reading in raw data files to statistical analysis and generating figures. The gazeR package is also designed to be as familiar as possible for the regular R user, thus handling data in formats and functions that will be accessible for most users.

In this paper, we provide a step-by-step walk-through of how to use the gazeR package to analyze data from experiments in which the primary outcome measure is gaze position or pupil size. There are several conceptual or theoretical discussions on best practices when analyzing pupil and gaze data available elsewhere (see Mathôt et al., 2018; Winn, Wendt, Koelewijn, and

Kuchinsky, 2018; Salverda & Tanenhaus, 2018). The main aim of the present paper is to illustrate and explain how to analyze gaze and pupil data in a more standardized way using gazeR, such that it may be used by researchers to analyze their own data. While there exist various packages and online resources to get started with eye-tracking, such materials are typically limited to the analysis of a single participant and do not represent what researchers typically want to do with their data. A secondary aim is to facilitate reproducible and transparent preprocessing of these types of data, using conventional practices in eye-tracking data processing, and smoothing the transition from data preprocessing to data analysis and visualization. In the remainder of this report, we provide a step-by-step walk through of the installation and core functionality of the gazeR package.

83 Table 1. Comparison of gazeR to other R packages for pupil preprocessing.

Packages	gazeR	pR	pupillometry	Pupillometry R	pupilParse	PupilPre	itrackR	itrak
Documentation	Yes	No	Yes	Yes	No	Yes	Yes	Yes
Supported file format	edfs; rectangular data (like CSV); SR EyeLink reports	Not known	Rectangular data (like csv) SMI; BeGaze sample report	Rectangular data (like csv)	Not known	SR Sample Reports	edfs	SR EyeLink reports
Supported eye trackers	Tracker agnostic (column names need to be specified)	Not known	SMI; SR EyeLink	Tacker agnostic (column names need to be specified)	SR EyeLink	SR EyeLink reports	SR EyeLink	SR EyeLink
Behavioral data extraction	Yes	No	No	No	No	No	No	No
Blink detection	Velocity based	Dilation and velocity measures	No	No	No	Relies on SR algorithms	Relies on SR algorithms	No
Blink extension	Yes	No	Yes	No	Yes	Yes	No	No
Interpolation	Linear and cubic-spline interpolation	Linear interpolation	Linear and cubic-spline interpolation	Linear and cubic-spline interpolation	Linear	linear interpolation or cubic spline interpolation	Linear	No
Smoothing/Filtering	N-moving average; hanning	No	N-moving average; hanning window	Hanning window; lowpass; median; regression based smoothing	Loess; lowess; Hampel	Butterworth	Butterworth	Low pass
Baseline correction	Subtractive or divisive baseline correction based on baseline median	Divisive method based on baseline mean	Subtractive or divisive baseline correction based on median	Subtractive baseline correction based on mean or median	Subtraction, division, and normalization based on baseline mean	Mean	None	Divisive based on baseline mean
Artifact rejection	Missing data; Median Absolute Deviation (MAD)	No	Missing data	Missing data	Min and max pupil size; SD	Median absolute deviation (MAD); Mahalanobis distance (basic or robust)	Missing data	Min and max pupil size; median absolute deviation (MAD)
Binning time data	Yes	No	Yes	Yes	Yes	Yes	Yes	No

84

85

86

87

Table 2. Comparison of gazeR to other R packages for gaze position (visual world paradigm) preprocessing.

VWP Packages	gazeR	eyetrackingR	VWpre	littlelisteners
Documentation	Yes	Yes	Yes	No
Supported file formats	edfs; rectangular data (like 'csv'); SR EyeLink reports	Rectangular data (like 'csv')	SR EyeLink Reports	Tobii (.gazepoint)
Supported eye trackers	Tracker agnostic (column names need to be specified)	Tracker agnostic (column names need to be specified)	SR EyeLink	Tobii
AOI labeling	Yes	Yes	Yes	Yes
Trackloss Identification	Yes	Yes	Yes	Yes
Binning Time data	Yes	Yes	Yes	No

Package Installation and Setup

Reading in Data

GazeR is meant to work on data in a relatively raw format, where each row is a sample corresponding to the sampling rate of the eye tracker. This allows gazeR to maximize compatibility: data from any eye-tracker can be used as long as the file contains information such as x and y coordinates, pupil size, and/or relevant event messages. For the examples contained herein, we will discuss how to preprocess data collected with one of the most popular commercial eye-trackers on the market—the SR EyeLink. Keeping with the spirit of open-access and transparency, however, we will highlight how to read in raw edf files for use with gazeR, so the proprietary SR software DataViewer is not necessary¹.

¹ Although not necessary, some EyeLink users nevertheless find it convenient to use the Fixation Reports or Sample Reports generated by DataViewer. A walkthrough for Sample and Fixation reports can be found here: <https://psyarxiv.com/gvcxb/>

100 **Package Installation**

101 The gazeR package can be installed along with helper packages using the remotes

102 package:

```
103 remotes::install_github("dmirman/gazer")  
104 #installs gazeR package from github  
105 remotes::install_github("tmalsburg/saccades/saccades")  
106 #install saccades package from github Needed the master version  
107 remotes::install_github("jashubbard/edfR")  
108 #install package if using edfs from SR
```

109 Once this has been completed, gazeR can be loaded along with additional useful packages:

```
110 library(gazer)  
111 library(tidyverse)  
112 library(zoo)  
113 library(knitr)  
114 library(edfR)  
115 library(saccades) #use master version from github  
116
```

117 Once gazeR and other helper packages have been installed and loaded the user is ready to start

118 preprocessing data.

119 **Preprocessing Gaze Position Data from the Visual World Paradigm**

120 In a typical instantiation of the Visual World Paradigm (VWP), participants hear spoken
121 instructions to manipulate or select one of several images on a computer screen or objects in the
122 real world (Cooper, 1974; Tanenhaus, Spivey-Knowlton, Eberhard, & Sedivy, 1995). Decades of
123 research have shown that the time course of fixation proportions – that is, the probability of
124 fixating a particular object at a particular time – reflects the activation of that object's mental
125 representation. Figure 1 illustrates a typical VWP task. In this example (from Mirman &

Graziano, 2012), the study examined semantic competition: the display contained a critical distractor that was related to the target either thematically (associates; e.g., *dog-leash*; shown in the left panel of Figure 1) or taxonomically (e.g., *apple-pear*). On each trial, the display contained a target object image, a semantic competitor (taxonomically or thematically related), and two unrelated distractors. The outcome measure was the probability of looks (fixation proportion) to a particular object at each point in time (example data shown in the right panel of Figure 1).

Figure 1. Left: Example display from a VWP experiment. The target is dog, the critical semantic competitor is leash (thematically related to the target), and snowman and carriage are unrelated distractors. Right: Example data showing the time course of target word recognition (solid line) and semantic competition: the semantically related competitor (dotted line) was fixated more than the unrelated distractors (dashed line).

Gaze preprocessing requires three main steps:

- (1) Reading in the data
- (2) Eliminating trackloss (out-of-bounds) data
- (3) Assigning areas of interest
- (4) Samples to bins (optional)

Reading in Gaze Data

In order to process the edf files generated by the EyeLink system you will need to first install the edf API provided by SR-Research, which is free of charge² and required for the ‘edfR’ package (Hubbard, 2014), which gazeR uses to read in edf files. In order to read in the edf files, for both pupil and visual world data, two folder paths must be specified: one path where the edf files are located and one where the raw .csv files should be saved.

```
directory_edf = ""  
# path to edfs  
directory_csv_from_edf_conversion = ""  
# path where csv files should be stored  
file_list_edf <- list.files(path=directory_edf, pattern=".edf")  
# extract the edfs from the directory_edf path
```

Once folder paths are specified, you can call the `parse_edf` function. This function imports the sample data from the edf files. The `type` argument must be specified as either “pupil” or “vwp”, depending on experimental design used. The `parse_edf` function merges the sample and message data from your raw edf files and wrangles them into a format suitable for preprocessing with gazeR. Specifically, the function places time in milliseconds, adds participant

² <https://www.sr-support.com/forumdisplay.php?17-EyeLink-Display-Software>

ids, trials, and sample messages, and computes the mean x and y gaze coordinates and diameter values for a monocular eye variable (the left, right, or mean of both eyes).

The `parse_edf` function generates a csv file from each edf file in a directory specified by the user. The `merge_gazer_files` function can then aggregate those new csv data files stored in the `directory_csv_from_edf_conversion` path specified above. For files that were processed with `parse_edf`, you need to set the `filetype` argument to “edf.”

```

169 parse_edf(file_list=file_list_edf,
170 directory_csv_from_edf_conversion,
171 type="vwp"))
172 # parses each edf file. Path to pupil edf files and where you want csv files
173 stored needed
174 file_list_csv <- list.files(path=directory_csv_from_edf_conversion, pattern="
175 .csv")
176 # Save csv files from specified directory
177 sampled_gaze_data <- merge_gazer_files(file_list_csv, filetype = "edf")
178 # merges all the sample files from the csv folder specified.
179

```

180 Behavioral Data

When using raw edf files, relevant behavioral message variables (e.g., conditional variables, RTs, and accuracy) are usually sent outside the sampling frequency of the eye tracker. To get the relevant trial variables in a nice table, you can run the `find_messages_edf` function, which will produce a csv file from each edf file. In order for the function to work properly the user must specify specific variable names (`varnames`) and the patterns that need to be replaced (`patterns`). SR prepends a “TRIAL VAR” marker to behavioral variables. After running this function, you can merge each participant’s behavioral report with the `merge_gazer_files` function. The behavioral report can then be joined to the gaze sample reports.

```

189 find_messages_edf(file_list= file_list_edf, varnames= c("TRIALID", "TRIAL_VAR
190 Condition", "TRIAL_VAR StimSlide.ACC", "TRIAL_VAR StimSlide.RT", "TRIAL_VAR C
191 orrectPort", "TRIAL_VAR CompPort"), patterns=c("TRIALID", "TRIAL_VAR Condition

```

```

192 ", "TRIAL_VAR StimSlide.ACC", "TRIAL_VAR StimSlide.RT", "TRIAL_VAR CorrectPor
193 t", "TRIAL_VAR CompPort"),output_dir)
194 #Need to know what your variable names are called these will be specific to
195 the experiment.
196 #use the edf path and csv path specified above
197 file_list_messages <- list.files(path = directory_csv_from_edf_conversion,
198                                full.names = TRUE, pattern = '.csv')
199 messages <- merge_gazer_files(file_list_messages filetype = "edf")
200

```

201 A pre-read version of this data set is included in gazeR to demonstrate what the sample
 202 data should look like after merging the message information:

```

203 gaze_path <- system.file("extdata", "vwp_data_raw_edf.xls", package = "gazer"
204 )
205 gaze_raw <- data.table::fread(gaze_path) # reads in large datasets quickly
206
207 gaze_data <- as_tibble(gaze_raw) # save as tibble
208
209 head(gaze_data)
210 ## # A tibble: 6 x 14
211 ##   subject trial  time pupil      x      y target  acc comport  rt
212 ##   <int> <int> <int> <int> <dbl> <dbl> <chr>  <int> <chr>  <int>
213 ## 1  9061     1     0  199  507.  358. pillow    1 image1  4000
214 ## 2  9061     1     2  199  506.  358. pillow    1 image1  4000
215 ## 3  9061     1     4  199  506.  359. pillow    1 image1  4000
216 ## 4  9061     1     6  199  506.  359. pillow    1 image1  4000
217 ## 5  9061     1     8  199  506.  359. pillow    1 image1  4000
218 ## 6  9061     1    10  199  506.  359. pillow    1 image1  4000
219 ## # ... with 4 more variables: correctport <chr>, condition <chr>,
220 ## #   TargetLocation <int>, CompLocation <int>
221

```

222 For this example data set the sample gaze data contains eye-tracking variables and experiment-
 223 specific values (positions of different objects, trial condition, participant accuracy and response
 224 time) that were extracted from the raw edf files.

```

225 summary(gaze_data)
226 ##      subject      trial      time      pupil
227 ##  Min.   :9061   Min.   : 1.00   Min.   :    0   Min.   :    0
228 ## 1st Qu.:9092   1st Qu.:17.00   1st Qu.:  876   1st Qu.: 140
229 ## Median :9146   Median :34.00   Median : 1752   Median : 175
230 ## Mean   :9118   Mean   :34.26   Mean   : 2005   Mean   : 184
231 ## 3rd Qu.:9153   3rd Qu.:52.00   3rd Qu.: 2634   3rd Qu.: 206
232 ## Max.   :9160   Max.   :70.00   Max.   :26186   Max.   :9398

```

```

233 ##           x                y                target
234 ## Min.      :   -3270    Min.      :   -3270    Length:1048575
235 ## 1st Qu.:    233    1st Qu.:    167    Class :character
236 ## Median :    512    Median :    364    Mode  :character
237 ## Mean     : 3064741    Mean     : 3064588
238 ## 3rd Qu.:    812    3rd Qu.:    530
239 ## Max.     :100000000    Max.     :100000000
240 ## accuracy      comport      reaction time      correctport
241 ## Min.      :0.0000    Length:1048575    Min.      : 2236    Length:1048575
242 ## 1st Qu.:1.0000    Class :character    1st Qu.: 3018    Class :character
243 ## Median :1.0000    Mode  :character    Median : 3330    Mode  :character
244 ## Mean     :0.9899                      Mean     : 3935
245 ## 3rd Qu.:1.0000                      3rd Qu.: 3779
246 ## Max.     :1.0000                      Max.     :26105
247 ## condition      TargetLocation    CompLocation
248 ## Length:1048575    Min.      :1.000    Min.      :1.000
249 ## Class :character    1st Qu.:1.000    1st Qu.:1.000
250 ## Mode  :character    Median :2.000    Median :2.000
251 ##                      Mean      :2.456    Mean      :2.465
252 ##                      3rd Qu.:3.000    3rd Qu.:3.000
253 ##                      Max.      :4.000    Max.      :4.000
254

```

255 Trackloss

256 Once the data are loaded in, some researchers might prefer to remove trials with excessive

257 trackloss (instances where the eyes travel outside of the viewing screen). This can be determined

258 by the X and Y coordinates at each sample relative to the size (resolution) of the screen.

259 Trackloss from the EyeLink systems use 1e+08. The `get_trackloss` function determines the on/off

260 screen status of each sample, computes the proportion of trackloss by trial and participant, and

261 filters out trials and subjects that pass a user-defined threshold (this filtering can be omitted by

262 setting the threshold to 1.0). The `screen_size` argument must be supplied as a numeric vector of

263 the X and Y dimensions of the computer screen used during the experiment. In this example, we

264 will not be throwing out data due to trackloss.

```

265
266 gaze_track <- get_trackloss(gaze, screen_size=c(1024, 768), missingthresh=.2)

```

267 Parsing areas of interest

268 The following preprocessing assumes that the interest areas (locations of objects) were
 269 static and that the fixation report includes columns indicating the location of each object for each
 270 trial. For this example, the objects were always presented in the four corners of the screen, though
 271 which object was in which corner was randomized. The four possible image locations are labeled
 272 as image1, image2, image3, and image4. The TargetLoc variable identifies which of those
 273 locations was the target object and the CompPort variable identifies which of those locations was
 274 the critical semantically related competitor. The gaze position was recorded in terms of (X,Y)
 275 coordinates. In order to determine which (if any) of the objects were being fixated, first identify
 276 the locations of the target and competitor images, then use gaze coordinates to determine which
 277 image location (if any) was being fixated, then compare gaze location to target and competitor
 278 locations. If gaze location has already been coded in terms of interest areas (many experiment
 279 programs do this dynamically, as the data are being collected), then this step can be skipped.

280 The `assign_aoi` function will match gaze positions to numbered areas of interest (AOI)
 281 based on screen coordinates (by default, 400x300 rectangles in the corners of the 1024x768
 282 screen), which will need to be matched to image location labels:

```
283 gaze_aoi <- assign_aoi(gaze, screen_size=c(1024, 768), aoi_size=c(400, 300),
284 aoi_loc=NULL, X="x", Y="y")
285
286 summary(gaze_aoi)
```

##	subject	trial	time	pupil
289 ##	Min. :9061	Min. : 1.00	Min. : 0	Min. : 0
290 ##	1st Qu.:9092	1st Qu.:17.00	1st Qu.: 876	1st Qu.: 140
291 ##	Median :9146	Median :34.00	Median : 1752	Median : 175
292 ##	Mean :9118	Mean :34.26	Mean : 2005	Mean : 184
293 ##	3rd Qu.:9153	3rd Qu.:52.00	3rd Qu.: 2634	3rd Qu.: 206
294 ##	Max. :9160	Max. :70.00	Max. :26186	Max. :9398
295 ##				

```

296 ##           x                y                target
297 ## Min.      :   -3270    Min.      :   -3270    Length:1048575
298 ## 1st Qu.:    233    1st Qu.:    167    Class :character
299 ## Median :    512    Median :    364    Mode  :character
300 ## Mean      : 3064741    Mean      : 3064588
301 ## 3rd Qu.:    812    3rd Qu.:    530
302 ## Max.      :100000000    Max.      :100000000
303 ##
304 ##           accuracy                comport                reaction_time                correctport
305 ## Min.      :0.0000    Length:1048575    Min.      : 2236    Length:1048575
306 ## 1st Qu.:1.0000    Class :character    1st Qu.: 3018    Class :character
307 ## Median :1.0000    Mode  :character    Median : 3330    Mode  :character
308 ## Mean      :0.9899                                Mean      : 3935
309 ## 3rd Qu.:1.0000                                3rd Qu.: 3779
310 ## Max.      :1.0000                                Max.      :26105
311 ##
312 ##           condition    TargetLocation    CompLocation    AOI
313 ## Length:1048575    Min.      :1.000    Min.      :1.000    Min.      :0.00
314 ## Class :character    1st Qu.:1.000    1st Qu.:1.000    1st Qu.:0.00
315 ## Mode  :character    Median :2.000    Median :2.000    Median :2.00
316 ##                               Mean      :2.456    Mean      :2.465    Mean      :1.71
317 ##                               3rd Qu.:3.000    3rd Qu.:3.000    3rd Qu.:3.00
318 ##                               Max.      :4.000    Max.      :4.000    Max.      :4.00
319 ##                               NA's      :93045

```

320 Now determine which object was being fixated by matching AOI codes with target and
 321 competitor locations:

```

322 gaze_aoi$Targ <- gaze_aoi$AOI == gaze_aoi$TargetLocation
323 gaze_aoi$Comp <- gaze_aoi$AOI == gaze_aoi$CompLocation
324 gaze_aoi$Unrelated <-
325   ((gaze_aoi$AOI != as.numeric(gaze_aoi$TargetLocation)) &
326    (gaze_aoi$AOI != as.numeric(gaze_aoi$CompLocation)) &
327    (gaze_aoi$AOI != 0) & !is.na(gaze_aoi$AOI))

```

328

329 Gathering Data

330 The specifics of data organization and aggregation will depend on the design and
 331 hypotheses of the specific study. For this example, the fixation locations need to be “gathered”
 332 from separate columns into a single column (see Supplemental Figure for a demonstration of this)
 333 and “NA” values need to be re-coded as not-fixations:

```

334 gaze_obj <- gaze_aoi %>%
335   dplyr::gather(key = "object", value = "fix",
336     Targ, Comp, Unrelated, factor_key = TRUE) %>%
337   dplyr::mutate(Fix = replace_na(fix, FALSE)) # recode NA as not-fixating
338
339 ## gather: reorganized (Targ, Comp, Unrelated) into (object, fix) [was 104857
340 5x18, now 3145725x17]
341 ## mutate: new variable 'Fix' with 2 unique values and 0% NA
342
343 summary(gaze_obj)
344 ##      subject      trial      time      pupil
345 ## Min.   :9061   Min.   : 1.00   Min.   :    0   Min.   :    0
346 ## 1st Qu.:9092   1st Qu.:17.00   1st Qu.:  876   1st Qu.: 140
347 ## Median :9146   Median :34.00   Median : 1752   Median : 175
348 ## Mean   :9118   Mean   :34.26   Mean   : 2005   Mean   : 184
349 ## 3rd Qu.:9153   3rd Qu.:52.00   3rd Qu.: 2634   3rd Qu.: 206
350 ## Max.   :9160   Max.   :70.00   Max.   :26186   Max.   :9398
351 ##
352 ##      x      y      target
353 ## Min.   : -3270   Min.   : -3270   Length:3145725
354 ## 1st Qu.:   233   1st Qu.:   167   Class :character
355 ## Median :   512   Median :   364   Mode  :character
356 ## Mean   : 3064741   Mean   : 3064588
357 ## 3rd Qu.:   812   3rd Qu.:   530
358 ## Max.   :100000000   Max.   :100000000
359 ##
360 ##      accuracy      comport      reaction time      correctport
361 ## Min.   :0.0000   Length:3145725   Min.   : 2236   Length:3145725
362 ## 1st Qu.:1.0000   Class :character   1st Qu.: 3018   Class :character
363 ## Median :1.0000   Mode  :character   Median : 3330   Mode  :character
364 ## Mean   :0.9899               Mean   : 3935
365 ## 3rd Qu.:1.0000               3rd Qu.: 3779
366 ## Max.   :1.0000               Max.   :26105
367 ##
368 ##      condition      TargetLocation      CompLocation      AOI
369 ## Length:3145725   Min.   :1.000   Min.   :1.000   Min.   :0.00
370 ## Class :character   1st Qu.:1.000   1st Qu.:1.000   1st Qu.:0.00
371 ## Mode  :character   Median :2.000   Median :2.000   Median :2.00
372 ##                   Mean   :2.456   Mean   :2.465   Mean   :1.71
373 ##                   3rd Qu.:3.000   3rd Qu.:3.000   3rd Qu.:3.00
374 ##                   Max.   :4.000   Max.   :4.000   Max.   :4.00
375 ##                   NA's   :279135
376 ##      object      fix      Fix
377 ## Targ      :1048575   Mode :logical   Mode :logical
378 ## Comp      :1048575   FALSE:2262238   FALSE:2448328
379 ## Unrelated:1048575   TRUE :697397    TRUE :697397
380 ##                   NA's :186090
381 ##

```

382 **Samples to Bins (Optional)**

You can downsample your data, if you choose, into larger time bins using the `downsample_gaze` function. This function will aggregate the set of samples into a time series consisting of standardized time bins with a size specified by the user (default is 50ms). In addition, it will drop columns that are no longer necessary. The user needs to specify a list of columns (aggvars) that define the aggregation level (e.g., individual trials) and should be kept after the binning is done. If you would like to keep the raw data unbinned, you can skip this part.

```
bin_gaze <- downsample_gaze(gaze_obj, bin.length = 50, timevar = "time", aggvars = c("subject", "condition", "target", "trial", "object", "timebins"))
## mutate: new variable 'timebins' with 525 unique values and 0% NA
head(bin_gaze)
```

	subject	condition	target	trial	object	timebin	acc	rt	Fix
	<int>	<chr>	<chr>	<int>	<fct>	<dbl>	<int>	<int>	<lgl>
## 1	9061	associate	anchor	18	Targ	0	1	3493	FALSE
## 2	9061	associate	anchor	18	Targ	50	1	3493	FALSE
## 3	9061	associate	anchor	18	Targ	100	1	3493	FALSE
## 4	9061	associate	anchor	18	Targ	150	1	3493	FALSE
## 5	9061	associate	anchor	18	Targ	200	1	3493	FALSE
## 6	9061	associate	anchor	18	Targ	250	1	3493	FALSE

Aggregating Data

In the final stage of preprocessing, the error and practice trials can be removed and the time window can be restricted, to make the data ready for aggregation. For this example, we group the trials by Subject, Condition, and Object type to calculate number of valid trials in each cell. Then also group by time point to calculate the number of object fixations and mean fixation proportion at each time point; that is, the time course of fixation. These are the subject-by-condition time courses that would go into an analysis.

```
gaze_subj <- bin_gaze %>%
  filter(acc == 1, condition != "practice", timebins < 3500) %>%
  # calculate number of valid trials for each subject-condition
```

```

414   group_by(subject, condition, object, timebins) %>%
415   summarize(meanfix = mean(Fix, na.rm=TRUE)) # fixation proportion
416   # there were two unrelated objects, so divide those proportions by 2
417   gaze_subj$meanfix[gaze_subj$object == "Unrelated"] <-
418   gaze_subj$meanfix[gaze_subj$object == "Unrelated"] / 2
419
420   ## filter: removed 28,158 rows (22%), 99,387 rows remaining
421   ## group_by: 3 grouping variables (subject, condition, object)
422   ## mutate (grouped): new variable 'nTrials' with 5 unique values and 0% NA
423   ## group_by: 4 grouping variables (subject, condition, object, timebins)
424   ## summarize: now 5,634 rows and 7 columns, 3 group variables remaining (subject, condition, object)
425
426
427   summary(gaze_subj)
428   ##      subject      condition      object      time
429   ## Min.      :9061   Length:140703   Targ      :46901   Min.      : 0
430   ## 1st Qu.:9092   Class :character   Comp      :46901   1st Qu.: 868
431   ## Median :9146   Mode  :character   Unrelated:46901   Median :1736
432   ## Mean    :9121                                     Mean    :1737
433   ## 3rd Qu.:9153                                     3rd Qu.:2604
434   ## Max.    :9160                                     Max.    :3498
435   ##      sumfix      ntrials      meanfix
436   ## Min.      : 0.000   Min.      : 8.00   Min.      :0.0000
437   ## 1st Qu.: 1.000   1st Qu.:20.00   1st Qu.:0.0250
438   ## Median : 2.000   Median :20.00   Median :0.1000
439   ## Mean    : 3.755   Mean    :18.88   Mean    :0.1709
440   ## 3rd Qu.: 5.000   3rd Qu.:20.00   3rd Qu.:0.2000
441   ## Max.    :20.000   Max.     :20.00   Max.     :1.0000

```

442 Plot fixation time course

443 After the fixations have been assigned to the object type and converted to time bins, they

444 are ready for visualization and statistical analysis. Below is a plot of the time course of fixation

445 proportions for each target type.

```

446   ggplot(gaze_subj)+
447     aes(time, meanfix, color = object) +
448     facet_wrap(~ condition) +
449     theme_gray() +
450     labs(x = "Time (ms)", y = "Proportion of Fixations") +
451     stat_summary(fun.y = mean, geom = "line") +
452     geom_vline(xintercept = 1300) +
453     annotate("text", x=1300, y=0.9, label="Word onset", hjust=0)

```

Figure 2. Time course of fixation proportions by condition. These data have been pre-processed and are ready for statistical analysis.

Preprocessing Pupil Data from a Lexical Decision Task

Recent advances in eye-tracking technology have lead to a burgeoning interest in cognitive pupillometry (i.e., measurement of changes in pupil size as it relates to higher-level processing). According to a recent PubMed search (see Figure 3), the number of studies employing pupillometry has grown exponentially since the first modern boom more than a half a century ago. The reason for this is quite simple: pupil size has been shown to be a reliable and valid index of cognitive effort or arousal across many domains, including word recognition (Geller, Still, & Morris, 2016), normal and impaired auditory perception (Zekveld et al., 2018), semantic cognition (Geller, Landrigan, & Mirman, 2019), attention allocation (Endogenous attention: Mathôt, Van derLinden, Grainger, Viti, 2013), working memory load (Granholm, Asarnow, Sarkin, & Dykes, 1996; Van Gerven, Paas, Van Merriënboer, & Schmidt, 2004), face perception (Goldinger, He, and Papesch, 2009), and general cognitive processing (Murphy et al.,

2014). While there are a number of good open-source programs available in R to analyze pupil data (see Forbes, 2019; Tsukahara, 2018), there are not many walk-throughs demonstrating how to go from raw data to fully pre-processed data. A recent methods review by Winn et al. (2018) describes and illustrates general principles like blink detection, interpolation, and filtering. The gazeR package includes functions for implementing these steps and here we demonstrate their use.

Figure 3. PubMed search for the keyword [pupillometry] from 1940-2019.

To demonstrate analysis of pupil data, we will be using an example data set containing data from a lexical decision task. For this walkthrough, participants ($N=3$) judged the lexicality (i.e. “was this a word or not a word?”) of printed and cursive stimuli while pupil diameter was

recorded. Because cursive stimuli are non-segmented and could be ambiguous, it was predicted that recognizing cursive stimuli would require more effort than printed words would (cf., Barnhart & Goldinger, 2010; Geller, Still, Dark, & Carpenter, 2018), resulting in larger pupil dilation.

Preprocessing pupil data requires the following steps:

- (1) Read in data
- (2) De-blinking
 - Extending blinks
 - Smoothing and interpolation
- (4) Baseline correction
- (5) Re-scaling
- (7) Artifact Rejection
 - Missing data
 - Unlikely pupil values
 - Median absolute deviation (MAD)
- (8) Event time alignment
- (9) Samples to bins

Reading in Pupil Data

In this example, we consider data from several subjects – the typical experimental scenario. For your own data (that includes many individual files), the function `parse_edf` with the type argument set to “pupil” will produce the necessary columns needed for gazeR to work. For non-edf files, you can use the `make_gazer` function to make your data suitable for gazeR.

502 This will return a data frame with column names changed to: subject, trial, blink, x, y, pupil,
 503 time, and message.

504 Once each edf file is saved as a .csv with the `parse_edf` function (explained above), you can
 505 call the `merge_gazer_files` function to aggregate all your pupil sample files.

```

506
507 directory_edf = ""
508 # path to edfs
509 directory_csv_from_edf_conversion = ""
510 # path where csv files should be stored
511
512 file_list_edf <- list.files(path=directory_edf, pattern=".edf")
513 # get list of edf files
514 parse_edf(file_list=file_list_edf,
515 output_dir = directory_csv_from_edf_conversion,
516 type="pupil"))
517 # parses edfs and saves them into directory
518 # folder path to csv folders from parse_edf
519
520 file_list_pupil_samp <- list.files(path=directory_csv_from_edf_conversion,
521 pattern=".csv")
522 # extract the processed csv files from directory_csv_from_edf_conversion
523
524 pd <- merge_gazer_files(file_list_pupil_samp, type = "edf")
525 # merges all the files from file_list_pupil_samp object

```

526 Behavioral Data (Optional)

527 For those interested in analyzing behavioral data, the `find_messages_edf` function can be
 528 used to cull the important behavioral data and merge with the sample data.

```

529 find_messages_edf(file_list= file_list_samp, varnames=c("TRIALID", "script", "
530 TRIAL_VAR item", "TRIAL_VAR RT", "ACCURACY", "alteration", "block"), patterns
531 =c("TRIALID", "!V TRIAL_VAR script", "!V TRIAL_VAR item", "!V TRIAL_VAR RT", "!
532 V TRIAL_VAR ACCURACY", "!V TRIAL_VAR alteration", "!V TRIAL_VAR block"),
533 output_dir)
534 # use edf and csv paths from above
535 #find out what variable names are called these will be specific to the experi
536 ment.

```

```

537 file_list_messages <- list.files(path = directory_csv_from_edf_conversion,
538                                full.names = TRUE, pattern = '.csv')
539
540 messages <- merge_gazer_files(file_list_messages filetype = "edf")
541 # rbind all the file_list_message files
542
543 pupil_behav_merge <- full_join(pupil_files, messages, by=c("subject", "trial"
544 ))
545 # merge behave with full sample report
546     Due to processing constraints, we are using a data file that includes data from a few

```

547 participants with all behavioral data included. The parse_edf, merge_gazer_files, and
548 find_messages_edf functions can be tested using example data that is available to download
549 from the Open Science Framework (OSF): <https://osf.io/fzu38/>. While reading in the data is fast
550 (even with many participants), some of the functions performed on the data can be
551 computationally intensive.

```

552 pupil_path <- system.file("extdata", "pupil_sample_files_edf.xls" , package =
553 "gazer") # get the file from gazer
554 pupil_raw <- data.table::fread(pupil_path) # reads in large files quickly
555 pupil_files <- as_tibble(pupil_raw) # saves the .xls file as tibble
556 summary(pupil_files)
557 ##      subject          trial          time          pupil
558 ## Length:1107527   Min.    :  1.00   Min.    :    0   Min.    : 1473
559 ## Class :character 1st Qu.: 38.00   1st Qu.: 1529   1st Qu.: 3342
560 ## Mode  :character Median : 75.00   Median : 3024   Median : 3561
561 ##              Mean   : 74.53   Mean   : 3662   Mean   : 3739
562 ##              3rd Qu.:111.00   3rd Qu.: 4684   3rd Qu.: 3927
563 ##              Max.    :148.00   Max.    :25812   Max.    :14088
564 ##              NA's    :122895
565 ##           x           y          blink
566 ## Min.    :   -1780   Min.    :   -1062   Min.    :0.0000
567 ## 1st Qu.:    946   1st Qu.:    525   1st Qu.:0.0000
568 ## Median :    996   Median :    546   Median :0.0000
569 ## Mean   : 10867320   Mean   : 10866923   Mean   :0.1082
570 ## 3rd Qu.:   1054   3rd Qu.:    572   3rd Qu.:0.0000
571 ## Max.    :100000000   Max.    :100000000   Max.    :1.0000
572 ## NA's    :5271     NA's    :5271
573 ##      message          acc          block          rt
574 ## Length:1107527   Min.    :0.0000   Min.    :0.00   Min.    : 508
575 ## Class :character 1st Qu.:1.0000   1st Qu.:1.00   1st Qu.: 1245
576 ## Mode  :character Median :1.0000   Median :1.00   Median : 2435
577 ##              Mean   :0.8671   Mean   :1.47   Mean   : 3934
578 ##              3rd Qu.:1.0000   3rd Qu.:2.00   3rd Qu.: 5018
579 ##              Max.    :1.0000   Max.    :2.00   Max.    :22449

```

```

580 ##
581 ##      item          script          alt
582 ## Length:1107527    Length:1107527    Length:1107527
583 ## Class :character  Class :character  Class :character
584 ## Mode  :character  Mode  :character  Mode  :character
585 ##
586
587

```

588 Once merged the `behave_data` function will cull the important behavioral data from the

589 sample report. The function will return a data frame without errors when `omitererrors=TRUE` or a

590 data frame with errors for accuracy/error analysis when `omitererrors=FALSE`. The columns

591 relevant for your experiment need to be specified within the `behave_col` names argument. This

592 function does not eliminate outliers; you must use your preferred method. Grange's (2015) `trimr`

593 package implements multiple standard methods of outlier exclusion

594 (<https://github.com/JimGrange/trimr>). The overall item and subject accuracy can be merged into

595 the pupil sample report.

```

596 behave_data<-
597 behave_pupil(pupil_files, omitererrors = FALSE, behave_colnames = c("subject",
598 script", "alt", "trial", "item", "acc", "rt", "block"))
599 behave_data
600 ## # A tibble: 444 x 8
601 ##   subject script alt   trial item      acc    rt block
602 ##   <chr>   <chr> <chr> <int> <chr>    <int> <int> <int>
603 ## 1 11c.edf cursive word     1 mourn.png     1  3833     0
604 ## 2 11c.edf cursive nwtl     2 nypmh.png     1  6067     0
605 ## 3 11c.edf print  word     3 sprigp.png    0  3233     0
606 ## 4 11c.edf print  nwtl     4 seivep.png    0  1781     0
607 ## 5 11c.edf print  word     5 ideal.png     1  1487     1
608 ## 6 11c.edf print  word     6 midst.png     1  1024     1
609 ## 7 11c.edf print  tl       7 quart        0   998     1
610 ## 8 11c.edf print  word     8 dirty.png     1  1198     1
611 ## 9 11c.edf print  tl       9 cheap        1  1316     1
612 ## 10 11c.edf print word    10 noisy.png    1  1141     1
613 ## # ... with 434 more rows
614
615
616 acc_by_item <- behave_pupil%>%
617   group_by(item) %>%
618   summarise(meanitemacc = mean(acc[block>0 & alt=="word"])) # overall item ac

```



```

619 curacy
620
621 acc_by_subject <- behave_data %>%
622   group_by(subject) %>%
623   summarise (meansubacc = mean(acc[block>0 & alt=="word"]))# subject accuracy
624
625 data_to_process <- pupil_files %>%
626   full_join(acc_by_item) %>% # merge item accuracy
627   full_join(acc_by_subject) # merge subject accuracy

```

628 In our data, we want to remove subject accuracy lower than 75% and item accuracy below 60%.

629 We can use the data frame generated above to calculate this when argument omiterrors=FALSE.

630 We then merge accuracy by items and subjects into the main pupil file.

631 We can now restrict preprocessing to valid trials by removing practice blocks, trials with
 632 incorrect responses, conditions that are not words, subjects with accuracy below 75%, and items
 633 with accuracy below 60%.

```

634 pupil_files1 <-data_to_process %>%
635   filter(block>0, acc==1, alt=="word",
636     meanitemacc >.60, meansubacc>.74) %>%
637   arrange(subject,trial, time)

```

638 Pupil Preprocessing is now ready to begin!

639 De-blinking

640 An important step in preprocessing pupil data is de-blinking. A major artifact in pupil data
 641 comes from blinking. When the eye blinks, the pupil momentarily becomes smaller as it is
 642 occluded more and more by the eyelids, making it difficult to compute the center of the pupil.
 643 Eye-trackers interpret this as a fast shift in pupil position and might erroneously classify it as a
 644 saccade. Additionally, the estimate of pupil size will rapidly decrease as the pupil occupies less of
 645 the camera image. This process happens in reverse (albeit a bit more slowly) as the eye is
 646 opening, so blinks are always flanked by an artifact that includes a false saccade and/or false

change in pupil size. Occasionally there will be some additional artifacts, such as short fixations preceding or following the blink. It is thus advisable to de-blink the data, which involves identifying blinks, removing data during the blink, removing data slightly before and after the blink, and then interpolating data during the period that was removed. In gazeR, blinks are identified automatically when importing raw edfs using the saccades package (Malsburg, 2015). For data in another format, the `detect_blink` function can be used to identify blinks. Blink detection from the saccades package uses a velocity-based algorithm taken from Engbert & Kliegl (2003). Blink events are identified as anything that looks like a fixation but has much lower dispersion than the typical fixation. In the saccades package, a blink is an event with a dispersion that is smaller than the median dispersion minus four times the median absolute deviation of the dispersion and only if this is the case for horizontal and vertical dispersion.³

A less trivial matter in pre-processing pupil data is deciding how many data points to remove before and after the blink. It has generally been recommended that data 100 ms before and after the blink should be eliminated. There are several ways one can deal with blinks (see Hershman, Henik, & Cohen, 2018). One method is to eliminate all blinks from a trial. This is generally not recommended as it can eliminate too much data, resulting in a loss of power. A more acceptable approach, and the one implemented in gazeR, is to extend the time window around the blinks so the interpolation starts 100-200 ms before the blink and continues after the blink (Nyström, Hooze, & Andersson, 2016; Satterthwaite et al., 2007). Extending the time

³ A comparison of results using the blink detection algorithm in the saccades package and the SR-EyeLink algorithm resulted in negligible differences, at least on grand averaged data. However, more extensive testing should be done. .

666 window around the blinks eliminates spurious samples caused by the closing and opening of the
667 eyelids. The `extend_blinks` function implements this method, with the `fillback` argument
668 specifying how far blinks should extend back in time (before the blink) and the `fillforward`
669 argument specifying how far blinks should extend forward in time (after the blink). This
670 function is robust to different sampling rates as long as the eye-tracker sampling rate is specified
671 in the `hz` argument. For this experiment, the tracker sampled at 250Hz (once every 4 ms) and
672 blinks were extended 100 ms forward and backward in time.

```
673 pup_extend<- pupil_files1 %>%  
674   group_by(subject, trial) %>%  
675   mutate(extendpupil=extend_blinks(pupil, fillback=100, fillforward=100, hz=2  
676   50))  
677 ## group_by: 2 grouping variables (subject, trial)  
678 ## mutate (grouped): new variable 'extendpupil' with 1,617 unique values and  
679 23% NA  
680
```

681 **Smoothing/Filtering and Interpolation**

682 Pupil data can be extremely noisy. To remove some of this noise, filtering and
683 interpolation are commonly done. In gazeR this is done in one step using the
684 `smooth_inteprolate_pupil` function. With the `step.first` argument users can choose to
685 either smooth the data first with a n-moving average, and then interpolate (`step.first =`
686 `“smooth”`) or vice versa (`step.first = “interp”`). Depending on which methods are selected,
687 the order of the steps can have negligible or substantial effects (see Figures 4 and 5); if applying
688 cubic-spline interpolation, smoothing before interpolation is generally advisable. The gazeR
689 package currently implements two common ways to smooth pupil data: n-point moving average
690 and Hanning window (we plan to include more smoothing options in future updates to the
691 package). To smooth the data, you must specify the column that contains the pupil information
692 and the size (in samples) of the moving average window. In this example, we use a 5-point
693 moving average ($n=5$) which, at a sampling rate of 250 Hz, corresponds to a 1250 ms moving
694 average.

Missing data stemming from blinks or failure of the eye tracker need to be interpolated.

The `smooth_interpolate_pupil` function searches the data and reconstructs the smoothed pupil size for each trial from the relevant samples using either linear interpolation (Bradley, Miccoli, Escrig, & Lang, 2008; Cohen et al., 2015; Siegle, Steinhauer, Carter, Ramel, & Thase, 2003) or cubic-spline interpolation (Mathôt, 2018). Considering the short duration of blinks and the relatively low speed of blinks, the choice of linear versus cubic interpolation will ultimately have negligible effect. If `extendblinks= FALSE`, samples with blinks are turned into NAs and are then interpolated. This function returns a tibble object with a column called `pup_interp` which contains interpolated values from the moving averaged pupillary data. There are also options to denote the max number of gaps to interpolate over (the `maxgap` argument is set to `Inf` by default). This requires the sampling rate (`hz`) of the eye tracker be specified. As an important note, if the Data Viewer was used to extend blinks, the `extendblinks` argument should be set to `FALSE`. If the `extend_blinks` function was used, the `extendblink` argument should be set to `TRUE`. It is important to note that SR only extends the blink column and does not set pupil size estimates during blinks to “NA” in the Sample Report. For this example, we will set `extendblinks` to `TRUE` and use linear interpolation. You can use cubic interpolation by changing type to “cubic.”

```
smooth_interp <- smooth_interpolate_pupil(pup_extend, pupil="pupil", extendpupil="extendpupil", extendblinks=TRUE, step.first="interp", filter="moving", maxgap=Inf, type="linear", hz=250, n=5)

## Performing linear interpolation
## Smoothing the pupil trace with moving average
```

Figure 4. Linear interpolation for one trial. The blink extension was successful: the isolated points (blue line) have been removed. In this example, for linear interpolation, it does not matter whether the interpolation was done first (green line) or the smoothing was first red line).

Figure 5. Cubic interpolation for one trial. The blink extension was successful: the isolated points (red line) have been removed. In this example, for cubic interpolation, somewhat different results are obtained if the interpolation step is first (green line) versus if the smoothing step is first (red line).

Baseline correction

To control for variability in overall pupil size arising from non-task related (tonic) state of

arousal, baseline correction is commonly used (but see Attard-Johnson, Ó Ciardha, & Bindemann, 2019). The two most popular types of baseline correction to identify task-evoked *dilation* are subtractive (pupil size - baseline) and divisive (change in pupil size - baseline / baseline). Subtractive baseline correction is more common in the literature (cf., Beatty, 1982; Laeng et al., 2012; Zekveld, Koelewijn, & Kramer, 2018), and this practice has been argued by Reilly, Kelly, Kim, Jett, and Zuckerman (2018) to be reflective of the linearity of the pupil response independent of baseline size, although the basis of that argument has been questioned⁴. The `baseline_correction_pupil` function finds the median pupil size during a specified baseline period for each trial and performs a subtraction baseline correction by default (see Mathôt et al., 2018, for argument that baseline correction should be done using the median, and not the mean, baseline value). By changing the `baseline_method` argument to “div”, you will get proportion change from baseline. In this example, subtractive baseline correction is applied to pupil size in arbitrary units (`pupil_colname = "pup_interp"`) though the same can be done for pupil size in mm or z-score. The baseline time window can be set with numerical values using the `baseline_window` argument from the `baseline_correction_pupil` function if events are fixed or static.

⁴ Reilly et al. varied luminance in order to elicit different baseline sizes, but that is not the typical source of baseline pupil size differences. Tonic baseline pupil size differences due to arousal, age, or other variables may affect the range of dilation reactivity in ways that differ from changes that are elicited by changes in luminance. Additionally, Wang et al. (2018) suggested that brighter lighting condition elicit *larger* dilations, on account of suppression of the parasympathetic suppressive influence on dilations. These factors can be used to motivate divisive baseline correction.

```

747 baseline_pupil <- baseline_correction_pupil(smooth_interp, pupil_colname='pup
748 _interp', baseline_window=c(500,1000))
749 ## Calculating median baseline from:500-1000
750 ## Merging baseline
751 ## Performing subtractive baseline correction
752

```

753 If events in your experiment occur dynamically, the `baseline_correction_pupil_msg`

754 function can be used. This function takes a user-specified baseline period immediately preceding

755 a stimulus onset message. In the below example, we set the `baseline_dur` argument to 100 ms

756 and the `event` argument to “target.” This will calculate the median baseline value for 100 ms

757 preceding target onset. The choice of baseline duration appears to largely inconsequential (Winn

758 et al., 2018).

```

759
760 baseline_pupil<-baseline_correction_pupil_msg(smooth_interp, pupil_colname='p
761 up_interp', baseline_dur=100, event="target", baseline_method = "sub")
762
763 ## Calculating median baseline from: target
764 ## Merging baseline
765 ## Performing subtractive baseline correction
766
767 head(baseline_pupil)
768
769 ##   subject trial baseline      item time pupil      x      y blink      mess
770 age
771 ## 1 11c.edf      5      3632 ideal.png      0 3648 974.3 588.4      0 MODERECORD
772 CRL
773 ## 2 11c.edf      5      3632 ideal.png      4 3641 972.9 587.3      0      <
774 NA>
775 ## 3 11c.edf      5      3632 ideal.png      8 3634 972.0 584.4      0      <
776 NA>
777 ## 4 11c.edf      5      3632 ideal.png     12 3636 971.8 584.9      0      <
778 NA>
779 ## 5 11c.edf      5      3632 ideal.png     16 3636 972.9 586.3      0      <
780 NA>
781 ## 6 11c.edf      5      3632 ideal.png     20 3631 973.0 586.1      0      <
782 NA>
783 ##   acc block   rt script  alt meanitemacc meansubacc extendpupil interp
784 ## 1   1      1 1487 print word      1 0.9722222      3648 3648
785 ## 2   1      1 1487 print word      1 0.9722222      3641 3641
786 ## 3   1      1 1487 print word      1 0.9722222      3634 3634
787 ## 4   1      1 1487 print word      1 0.9722222      3636 3636
788 ## 5   1      1 1487 print word      1 0.9722222      3636 3636
789 ## 6   1      1 1487 print word      1 0.9722222      3631 3631

```



```

790 ##   pup_interp baselinecorrectedp
791 ## 1    3641.00           9.00
792 ## 2    3639.75           7.75
793 ## 3    3639.00           7.00
794 ## 4    3635.60           3.60
795 ## 5    3634.20           2.20

```

796

797 Re-Scaling

798 So far, the analysis steps have used arbitrary pupil units. It is advised that these be
799 transformed into a standardized unit in order to make comparisons between individuals.
800 Numerous options have been used in prior studies: z-scores (see Cohen, Moyal, & Henik, 2015;
801 Einhauser, Stout, Koch, & Carter, 2008; Kang & Wheatley, 2015), absolute changes in mm (e.g.,
802 Beatty, 1982; Geller, Landrigan, & Mirman, 2019; Geller et al., 2016), proportional change
803 relative to baseline (Winn, 2016), proportional change relative to peak (Winn, 2016; Winn &
804 Moore, 2018), and absolute change relative to dynamic range of pupil reactivity elicited by the
805 light reflex (Piquado, Isaacowitz, & Wingfield, 2010). To convert arbitrary pupil size to mm, we
806 measured the scaling factor by running a short experiment with an artificial pupil (5 mm in size)
807 and calculated the average pupil size in arbitrary units. At a fixed camera-to-pupil distance of 90
808 cm, the 5mm pupil was coded as 5570 arbitrary pixel units. This information was entered into the
809 equation below to convert arbitrary units to mm. Specifically, the smoothed pupil size value is
810 multiplied by 5/5570 to re-scale the values to mm.

```

811 timebinsmm <- baseline_pupil %>%
812   mutate(pupilmm = (pup_interp * 5)/5570.29)

```

813 Alternatively, the arbitrary pupil units can be converted to a z-score using the scale
814 function.

```
815 timebinsz <- baseline_pupil %>%  
816   group_by(subject, trial) %>%  
817   mutate(pupilz = scale(pup_interp))  
818
```

819 For the rest of the walkthrough, arbitrary pupil size will be used.

820 **Artifact Rejection**

821 **Missingness.** The `count_missing_pupil` function will remove subjects and items that
822 have a large amount of missing data – the threshold for “a large amount” is specified by the
823 researcher. It has been recommended by Winn et al. (2018) that a reasonable threshold is 20%,
824 but that the exact importance of missing data might be weighted by specific timing landmarks in
825 the experiment trials. For this example, we have set the `missingthresh` argument to .2 and the
826 pupil argument to the raw, non-interpolated, pupil dilation column in our dataset. The
827 `count_missing_pupil` function returns the percentage of subjects and trials that have been
828 excluded for reporting.

```
829 pup_missing <- count_missing_pupil(baseline_pupil, pupil= “pupil”, missingthr  
830 esh = .2)  
831  
832 ## % trials excluded:0.07  
833 ## subjects taken out:
```

834 **Spurious pupil values.** Unlikely pupil values that are too small and too large should be
835 removed from the data (Mathôt et al., 2018; Winn et al., 2018). Mathôt (2018) recommended
836 against removing data based on a subject-independent fixed criterion (e.g., above or below a SD
837 cut-off or a specified lower and upper pupil boundary). This is due to the inherent heterogeneity
838 of pupil sizes across experiments. Instead, Mathôt (2018) recommend visual inspection to
839 determine unlikely pupil values. This can be done using a simple histogram to plot the
840 pupillometric data. Based on the histogram below, it seems reasonable to remove pupil sizes less
841 than 2500 and greater than 5000.

```

842 puphist <- ggplot(baseline_pupil aes(x = pup_interp)) +
843   geom_histogram(aes(y = ..count..), colour = "green", binwidth = 0.5) +
844   geom_vline(xintercept = 2500, linetype="dotted") +
845   geom_vline(xintercept = 5100, linetype="dotted") +
846   xlab("Pupil Size") +
847   ylab("Count") +
848   theme_bw()
849
850 print(puphist)

```

Figure 6. Histogram of recorded pupil sizes.

```

853 pup_outliers <- pup_missing %>%
854   # based on visual inspection
855   dplyr::filter(pup_interp >= 2500, pup_interp <= 4000)
856

```

Median absolute deviation (MAD). After interpolation, it is a good idea to perform a second pass to make sure that the data are not contaminated by rapid pupil size disturbances. These artifacts can be detected using the median absolute deviation, which is a more robust variability metric than traditional measures of variability (e.g., standard deviation; Hampel, 1974; Kret & Sjak-Shie, 2018). For each time point, the `speed_pupil` function calculates the normalized dilation speed, which is the absolute change in pupil size between samples divided by

the temporal separation between them. To detect outliers, the MAD is calculated from the dilation speed variable and multiplied by a constant (in this case 16, which is the value used in Kret & Sjak-Shie, 2018). The constant controls the sensitivity threshold: The higher the constant, the more extreme a value needs to be in order to be marked for removal. The MAD is added to the median dilation speed variable using the `calc_mad` function; values above this threshold are then removed.

```
mad_removal <- pup_outliers %>%
  group_by(subject, trial) %>%
  mutate(speed=speed_pupil(pup_interp,time_zero)) %>%
  mutate(MAD=calc_mad(speed, n = 16)) %>%
  filter(speed < MAD)
```

Event Time Alignment

In most psychological experiments, each trial includes several events. In the example experiment, each trial began with a fixation screen (small cross in the center of the screen) and the stimulus of interest appeared on screen 1s after trial onset. These events are documented in the data file: the onset of the target is denoted by the trial message “target.” We can use this information to align the data so that time=0 corresponds to stimulus onset (i.e., the analysis window of interest) rather than trial onset. The `onset_pupil` function performs this alignment using three arguments: time column, sample message column, and the event of interest (“target” in our example). In the output below, we can see that our experiment now starts at zero, when the target was displayed on screen.

```
baseline_pupil_onset <- mad_removal %>%
  dplyr::group_by(subject, trial) %>%
  dplyr::mutate(time_zero=onset_pupil(time, message, event=c("target"))) %>%
  ungroup() %>%
  dplyr::filter(time_zero >= -100 & time_zero <= 2500) %>%
  select(subject, trial, time_zero, message, baseline, baselinecorrectedp, pupil, pup_interp, script)
## select: dropped 13 variables (time, x, y, blink, acc, ...)
```

```

892
893 head(baseline_pupil_onset)
894
895 ## # A tibble: 123,322 x 9
896 ##   subject trial time_zero message baseline baselinecorrect... pupil pup_int
897 erp
898 ##   <chr>   <int>   <int> <chr>         <dbl>         <dbl> <int>   <d
899 bl>
900 ## 1 11c.edf     5         0 target      3632         -0.600  3630    36
901 31.
902 ## 2 11c.edf     5         4 <NA>        3632         -2.60   3629    36
903 29.
904 ## 3 11c.edf     5         8 <NA>        3632         -4.20   3631    36
905 28.
906 ## 4 11c.edf     5        12 <NA>        3632         -5.4    3625    36
907 27.
908 ## 5 11c.edf     5        16 <NA>        3632          -6    3624    36
909 26
910 ## 6 11c.edf     5        20 <NA>        3632          -8    3624    36
911 24
912 ## 7 11c.edf     5        24 <NA>        3632          -9    3626    36
913 23
914 ## 8 11c.edf     5        28 <NA>        3632         -9.4    3621    36
915 23.
916 ## 9 11c.edf     5        32 <NA>        3632         -9.60   3620    36
917 22.
918 ## 10 11c.edf    5        36 <NA>        3632        -10.8   3622    36
919 21.
920 ## # ... with 123,312 more rows, and 1 more variable: script <chr>

```

921 Samples to Bins

922 If the data are recorded at a relatively high sampling frequency (e.g., 250Hz in this

923 example), it may be useful to aggregate the data into time bins that are somewhat larger than the

924 sample rate (users can specify a time bin size to use). The `downsample_gaze` function will

925 aggregate the set of samples into a time series consisting of standardized time bins with a size

926 specified by the user for pupil data when the `type` argument is set to “pupil”. In addition, it will

927 drop columns that are no longer necessary. The user needs to specify a vector of column names

928 (`aggvars`) that define the aggregation level (e.g., individual trials) and should be kept after the

929 binning is done. This produces an average baseline-corrected pupil diameter for each subject,

condition, and timebin. If you would like to keep the raw data unbinned, you can skip this part.

This function returns a tibble with an added column called timebins, which can be used for

aggregation (e.g., calculating the mean pupil size in each time bin).

```
timebins1<- downsample_gaze(baseline_pupil_onset, bin.length=100, timevar = "
time_zero", aggvars = c("subject", "script", "timebins"), type="pupil")
## mutate: new variable 'timebins' with 27 unique values and 0% NA
timebins1
## # A tibble: 162 x 4
##   subject script  timebins aggbaseline
##   <chr>   <chr>      <dbl>      <dbl>
## 1 11c.edf cursive    -100        2.63
## 2 11c.edf cursive      0       -4.00
## 3 11c.edf cursive     100       -6.59
## 4 11c.edf cursive     200       -6.89
## 5 11c.edf cursive     300       -2.04
## 6 11c.edf cursive     400        6.83
## 7 11c.edf cursive     500       12.3
## 8 11c.edf cursive     600        7.88
## 9 11c.edf cursive     700        4.28
## 10 11c.edf cursive     800        9.07
## # ... with 152 more rows
```

Pupillary Data Visualization

After baseline-correction and artifact rejection, the data are ready for visualization and statistical analysis. The pre-processed data produced by gazeR are highly flexible and compatible with different visualization strategies.

```
cursive_plot <- ggplot(timebins1)+
  aes(timebins, aggbaseline, linetype=script, color=script) +
  stat_summary(fun.y = "mean", geom = "line", size = 1)
  theme_bw() +
  labs(x = "Time (ms)", y = "Pupil Dilation (change from baseline (a.u.))") +
  geom_hline(yintercept=0.0)
cursive_plot
```

Figure 7. Pupillary time course as a function of script type for the 3 participants.

Data Analysis

The gazeR package is deliberately agnostic to how researchers should analyze their data, leaving the data in a format that is flexible and compatible with a variety of statistical modeling strategies. For instance, various curve-fitting strategies such as growth curve analysis, general additive models and/or functional data analysis (Jackson & Sirois, 2009; Mirman, 2014; Seedorff, Oleson, and McMurray, 2018; van Rij et al., 2019). In the absence of a field-standard statistical approach, we leave it up to the researcher to choose what statistical analysis to use.

For those interested in growth curve modeling, the gazeR package does contain a helper function (`code_poly`) to facilitate growth curve analysis (GCA) using orthogonal polynomials (Mirman, 2014). The `code_poly` function takes your time column and adds polynomial transformations of time up to a user-specified order. The polynomial transformation can be natural or orthogonal; the orthogonal version can be useful because it makes the time terms

981 uncorrelated and scales them to the same range (for a discussion of natural vs. orthogonal
 982 polynomials see Mirman, 2014).

```
983 code_poly(df = gaze_subj, predictor = "time", poly.order = 4, orthogonal = TR
984 UE, draw.poly = TRUE)
```

985
 986 Figure 8. 4th-order Orthogonal Polynomials for the VWP data.

```
987 ## # A tibble: 140,703 x 12
988 ## # Groups:   subject, condition, object [81]
989 ##   subject condition object  time sumfix ntrials meanfix time.Index  poly
990 1
991 ##     <int> <chr>      <fct> <int> <int> <int> <dbl> <dbl> <dbl>
992 >
993 ## 1    9061 associate Targ      0      0    20      0      1 -0.041
994 4
995 ## 2    9061 associate Targ      2      0    20      0      2 -0.041
996 3
997 ## 3    9061 associate Targ      4      0    20      0      3 -0.041
```



```

998 3
999 ## 4 9061 associate Targ 6 0 20 0 4 -0.041
1000 2
1001 ## 5 9061 associate Targ 8 0 20 0 5 -0.041
1002 2
1003 ## 6 9061 associate Targ 10 0 20 0 6 -0.041
1004 1
1005 ## 7 9061 associate Targ 12 0 20 0 7 -0.041
1006 1
1007 ## 8 9061 associate Targ 14 0 20 0 8 -0.041
1008 0
1009 ## 9 9061 associate Targ 16 0 20 0 9 -0.041
1010 0
1011 ## 10 9061 associate Targ 18 0 20 0 10 -0.041
1012 0
1013 ## # ... with 140,693 more rows, and 3 more variables: poly2 <dbl>,
1014 ## # poly3 <dbl>, poly4 <dbl>
1015 The gazeR package also includes a function (get_ranef) for extracting random effects

```

1016 from a growth curve model in a format that is convenient for quantifying individual differences
 1017 (see Mirman, 2014, Chapter 7) and a function for estimating p-values for linear mixed effects
 1018 models using the normal and/or Kenward-Roger approximations (`get_pvalues`).

1019 Discussion

1020 While there are a number of viable solutions available to process eye-tracking data, not all
 1021 are suitable for research purposes:

- 1022 • An all-graphical interface seldom provides information about the underlying data
 1023 analysis and can be difficult to reproduce.
- 1024 • File formats are sometimes proprietary and undocumented, lacking detailed
 1025 annotation necessary for replicability.
- 1026 • Source code and description of the algorithms are not always accessible to the
 1027 user.
- 1028 • Some implementations rely on expensive proprietary software.

The research community needs solutions that are completely open, with the possibility of directly manipulating and annotating the code, data, and parameters so that others may replicate or critique the methods. This article summarized and demonstrated the functionality of gazeR -- a free, open-source package written in R. We walked through important functions needed to preprocess both gaze position and pupil size data and make it suitable for analysis. This provides a generalized, replicable, and transparent method for preprocessing raw eye-tracking data.

How does gazeR compare to other existing solutions?

While there exist several packages in R to analyze pupillometry data or visual world paradigm data (see Table 1), gazeR has several advantages over existing solutions. First, gazeR is currently the only package that allows one to analyze pupillometric and gaze data all in one package. Second, many of the R packages rely on proprietary software and algorithms for key analysis steps (such as fixation/saccade parsing and blink detection). GazeR avoids this by allowing users to read in edf files directly and by using open-source event detection algorithms. Third, when deciding on a package to use, it is important the package be well documented and supported. Many of the packages listed in the table were developed and updated several years ago, and it is unclear if the package developers are still servicing the packages. Most crucially, the source code and descriptions for many of the packages are not well explained. Lastly, a significant problem for the standardization of analyses, especially with the implementation of a package, is that with constantly changing software environments results likely differ. For example, different R versions could change the results, as could differences between platforms (Windows, macOS, or Linux). To this end, a containerized version of gazeR is available via Binder on Github, which allows users to follow along with this walkthrough in the exact environment where it was created, thus supporting reproducibility. Our hope is this encapsulated environment will help facilitate use of our pipeline to users' own data. In the future we hope to host gazeR on Docker, similar to other

preprocessing pipelines (e.g., fmripreg; Esteban et al., 2019). Overall, we feel that this walkthrough, end-to-end implementation in the free, cross-platform environment of R, and the fact that gazeR conforms to best practices for pupillometry laid out in recent reviews (Winn et al., 2018; Mathot et al., 2018) makes gazeR a valuable tool for analyzing your pupillometry and gaze data.

Limitations

The gazeR pre-processing pipeline is not exhaustive. The implemented set of functions should suffice for researchers to pre-process their gaze and pupil data based on recent recommendations, but there are factors that are not included yet. For example, gaze position is known to influence pupil size (Brisson et al., 2013; Gagl, Hawelka, & Hutzler, 2011), called the pupil foreshortening effect. This effect occurs when rotations of the eyes change the angle at which the camera records the pupil, and therefore also the pupil's apparent size. As such, this manifestation of gaze position in pupil size should ideally be controlled or corrected for. A simple way to do this would be to include X and Y gaze coordinates into the analysis model as a co-variate. If reading in edf files, the X-Y gaze coordinates are included and can easily be included in this analysis. Additionally, various aspects of pupil dilation might be more or less important to the analysis, which might benefit from examination of additional features such as onset and offset slopes (c.f., Winn & Moore, 2018). Because the gazeR package is open-source, modifications can always be made to incorporate additional functionality. Suggestions and contributions from users are encouraged and can be submitted through the package github page: <https://github.com/dmirman/gazer>.

Finally, the current instantiation of gazeR has only been tested with data files from the SR EyeLink (i.e., raw edfs and sample reports). Much of the gazeR functionality is easily compatible

1076 with other eye-trackers with the addition of functions for reading data and renaming columns
1077 (variables) to match the gazeR conventions. Future updates to the package will include added
1078 support for Tobii and Gazepoint eye trackers.

1079 To summarize, the gazeR package provides general, open-source tools for replicable and
1080 transparent processing of gaze and pupillometry data. GazeR grew out of in-house preprocessing
1081 code in several research groups and is already being used by several additional research groups. It
1082 is our hope that more researchers will use it and will contribute to its improvement.

1083

1084

1085

1086

1087

1088

1089

1090

1091

References

Attard-Johnson, J., Ó Ciardha, C., & Bindemann, M. (2019). Comparing methods for the analysis of pupillary response. *Behavior Research Methods*, 51(1), 83–95.

<https://doi.org/10.3758/s13428-018-1108-6>

Barnhart, A. S., & Goldinger, S. D. (2010). Interpreting chicken-scratch: Lexical access for handwritten words. *Journal of Experimental Psychology: Human Perception and Performance*, 36(4), 906–923. <https://doi.org/10.1037/a0019258>

Beatty, J. (1982a). Task-evoked pupillary responses, processing load, and the structure of processing resources. *Psychological Bulletin*, 91(2), 276–292. <https://doi.org/10.1037/0033-2909.91.2.276>

Bradley, M. M., Miccoli, L., Escrig, M. A., & Lang, P. J. (2008). The pupil as a measure of emotional arousal and autonomic activation. *Psychophysiology*, 45(4), 602–607. <https://doi.org/10.1111/j.1469-8986.2008.00654.x>

Brisson, J., Mainville, M., Mailloux, D., Beaulieu, C., Serres, J., & Sirois, S. (2013). Pupil diameter measurement errors as a function of gaze direction in corneal reflection eyetrackers. *Behavior Research Methods*, 45(4), 1322–1331. <https://doi.org/10.3758/s13428-013-0327-0>

Cohen, N., Moyal, N., & Henik, A. (2015). Executive control suppresses pupillary responses to aversive stimuli. *Biological Psychology*, 112, 1–11. <https://doi.org/10.1016/j.biopsycho.2015.09.006>

- 1111 Cooper, R. M. (1974). The control of eye fixation by the meaning of spoken language: A
1112 new methodology for the real-time investigation of speech perception, memory, and language
1113 processing. *Cognitive Psychology*, 6(1), 84–107. [https://doi.org/10.1016/0010-0285\(74\)90005-X](https://doi.org/10.1016/0010-0285(74)90005-X)
- 1114 Einhauser, W., Stout, J., Koch, C., & Carter, O. (2008). Pupil dilation reflects perceptual
1115 selection and predicts subsequent stability in perceptual rivalry. *Proceedings of the National*
1116 *Academy of Sciences*, 105(5), 1704–1709. <https://doi.org/10.1073/pnas.0707727105>
- 1117 Engbert, R., & Kliegl, R. (2003). Microsaccades uncover the orientation of covert
1118 attention. *Vision Research*, 43(9), 1035–1045. [https://doi.org/10.1016/S0042-6989\(03\)00084-1](https://doi.org/10.1016/S0042-6989(03)00084-1)
- 1119 Esteban, O., Markiewicz, C. J., Blair, R. W., Moodie, C. A., Isik, A. I., Erramuzpe, A., ...
1120 Gorgolewski, K. J. (2019). fMRIPrep: a robust preprocessing pipeline for functional MRI. *Nature*
1121 *Methods*, 16(1), 111–116. <https://doi.org/10.1038/s41592-018-0235-4>
- 1122 Forbes, S.H. (2019). pupillometryR: An R package for preparing and analysing
1123 pupillometry data. Retrieved from <https://github.com/samhforbes/PupillometryR>
- 1124 Gagl, B., Hawelka, S., & Hutzler, F. (2011). Systematic influence of gaze position on
1125 pupil size measurement: analysis and correction. *Behavior Research Methods*, 43(4), 1171–1181.
1126 <https://doi.org/10.3758/s13428-011-0109-5>
- 1127 Geller, J., Landrigan, J.-F., & Mirman, D. (2019). A Pupillometric Examination of
1128 Cognitive Control in Taxonomic and Thematic Semantic Memory. *Journal of Cognition*, 2(1).
1129 <https://doi.org/10.5334/joc.56>

- 1130 Geller, J., Still, M. L., Dark, V. J., & Carpenter, S. K. (2018). Would disfluency by any
1131 other name still be disfluent? Examining the disfluency effect with cursive handwriting. *Memory*
1132 & *Cognition*, 46(7), 1109–1126. <https://doi.org/10.3758/s13421-018-0824-6>
- 1133 Geller, J., Still, M. L., & Morris, A. L. (2016). Eyes wide open: Pupil size as a proxy for
1134 inhibition in the masked-priming paradigm. *Memory & Cognition*, 44(4), 554–564.
1135 <https://doi.org/10.3758/s13421-015-0577-4>
- 1136 Goldinger, S. D., He, Y., & Papesch, M. H. (2009). Deficits in cross-race face learning:
1137 Insights from eye movements and pupillometry. *Journal of Experimental Psychology: Learning,*
1138 *Memory, and Cognition*, 35(5), 1105–1122. <https://doi.org/10.1037/a0016548>
- 1139 Grange, J.A. (2015). trimr: An implementation of common response time trimming
1140 methods. R package version 1.0.1. <https://cran.r-project.org/web/packages/trimr/index.html>
- 1141 Granholm, E., Asarnow, R. F., Sarkin, A. J., & Dykes, K. L. (1996). Pupillary responses
1142 index cognitive resource limitations. *Psychophysiology*, 33(4), 457–461. Retrieved from
1143 <http://www.ncbi.nlm.nih.gov/pubmed/8753946>
- 1144 Hampel, F. R. (1974). The influence curve and its role in robust estimation. *Journal of the*
1145 *American Statistical Association*, 69(346), 383–393.
1146 <https://doi.org/10.1080/01621459.1974.10482962>
- 1147 Hershman, R., Henik, A., & Cohen, N. (2018). A novel blink detection method based on
1148 pupillometry noise. *Behavior Research Methods*, 50(1), 107–114.
1149 <https://doi.org/10.3758/s13428-017-1008-1>

- 1150 Karatekin, C., Couperus, J. W., & Marcus, D. J. (2004). Attention allocation in the dual-
1151 task paradigm as measured through behavioral and psychophysiological responses.
1152 *Psychophysiology*, 41(2), 175–185. <https://doi.org/10.1111/j.1469-8986.2004.00147.x>
- 1153 Kret, M. E., & Sjak-Shie, E. E. (2018). Preprocessing pupil size data: Guidelines and
1154 code. *Behavior Research Methods*, 1–7. <https://doi.org/10.3758/s13428-018-1075-y>
- 1155 Mathôt, S. (2018). Pupillometry: Psychology, Physiology, and Function. *Journal of*
1156 *Cognition*, 1(1). <https://doi.org/10.5334/joc.18>
- 1157 Mathôt, S., Fabius, J., Van Heusden, E., & Van der Stigchel, S. (2018). Safe and sensible
1158 preprocessing and baseline correction of pupil-size data. *Behavior Research Methods*, 50(1), 94–
1159 106. <https://doi.org/10.3758/s13428-017-1007-2>
- 1160 Mathôt, S., van der Linden, L., Grainger, J., & Vitu, F. (2013). The Pupillary Light
1161 Response Reveals the Focus of Covert Visual Attention. *PLoS ONE*, 8(10), e78168.
1162 <https://doi.org/10.1371/journal.pone.0078168>
- 1163 Murphy, P. R., O'connell, R. G., O'sullivan, M., Robertson, I. H., & Balsters, J. H. (2014).
1164 Pupil diameter covaries with BOLD activity in human locus coeruleus. *Human Brain*
1165 *Mapping*, 35(8), 4140-4154.
- 1166 Nyström, M., Hooge, I., & Andersson, R. (2016). Pupil size influences the eye-tracker
1167 signal during saccades. *Vision Research*, 121, 95–103.
1168 <https://doi.org/10.1016/J.VISRES.2016.01.009>
- 1169 Patil, I. (2018). ggstatsplot:“ggplot2” Based Plots with Statistical Details. CRAN.

- 1170 Piquado, T., Isaacowitz, D., & Wingfield, A. (2010). Pupillometry as a measure of
1171 cognitive effort in younger and older adults. *Psychophysiology*, 47(3), 560–569.
1172 <https://doi.org/10.1111/j.1469-8986.2009.00947.x>
- 1173 Reilly, J., Kelly, A., Kim, S. H., Jett, S., & Zuckerman, B. (2018). The human task-evoked
1174 pupillary response function is linear: Implications for baseline response scaling in pupillometry.
1175 *Behavior Research Methods*. <https://doi.org/10.3758/s13428-018-1134-4>
- 1176 Salverda, A. P., & Tanenhaus, M. K. (2018). The visual world paradigm. In Annette M. B.
1177 de Groot and Peter Hagoort (Eds) *Research methods in psycholinguistics and the neurobiology of*
1178 *language: A practical guide*, pp. 89-110. Wiley Blackwell.
- 1179 Satterthwaite, T. D., Green, L., Myerson, J., Parker, J., Ramaratnam, M., & Buckner, R. L.
1180 (2007). Dissociable but inter-related systems of cognitive control and reward during decision
1181 making: Evidence from pupillometry and event-related fMRI. *NeuroImage*, 37(3), 1017–1031.
1182 <https://doi.org/10.1016/j.neuroimage.2007.04.066>
- 1183 Seedorff, M., Oleson, J., & McMurray, B. (2018). Detecting when timeseries differ: Using
1184 the Bootstrapped Differences of Timeseries (BDOTS) to analyze Visual World Paradigm data
1185 (and more). *Journal of Memory and Language*, 102, 55–67.
1186 <https://doi.org/10.1016/J.JML.2018.05.004>
- 1187 Siegle, G. J., Steinhauer, S. R., Carter, C. S., Ramel, W., & Thase, M. E. (2003). Do the
1188 Seconds Turn Into Hours? Relationships between Sustained Pupil Dilation in Response to
1189 Emotional Information and Self-Reported Rumination. *Cognitive Therapy and Research*, 27(3),
1190 365–382. <https://doi.org/10.1023/A:1023974602357>

- 1191 Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K. M., & Sedivy, J. C. (1995).
1192 Integration of visual and linguistic information in spoken language comprehension. *Science (New*
1193 *York, N.Y.)*, 268(5217), 1632–1634. Retrieved from
1194 <http://www.ncbi.nlm.nih.gov/pubmed/7777863>
- 1195 Tsukahara, J.S. (2018). pupillometry: An R Package to Preprocess Pupil Data. Retrieved
1196 from <https://dr-jt.github.io/pupillometry>
- 1197 Van Gerven, P. W. M., Paas, F., Van Merriënboer, J. J. G., & Schmidt, H. G. (2004).
1198 Memory load and the cognitive pupillary response in aging. *Psychophysiology*, 41(2), 167–174.
1199 <https://doi.org/10.1111/j.1469-8986.2003.00148.x>
- 1200 van Rij, J., Hendriks, P., van Rijn, H., Baayen, R. H., & Wood, S. N. (2019). Analyzing
1201 the Time Course of Pupillometric Data. *Trends in Hearing*, 23, 233121651983248.
1202 <https://doi.org/10.1177/2331216519832483>
- 1203 von der Malsburg, T. (2019). Saccades: Detection of fixations in eye-tracking data.
1204 Retrieved from <https://github.com/tmalsburg/saccades>
1205
1206
- 1207 Winn, M. B., Wendt, D., Koelewijn, T., & Kuchinsky, S. E. (2018). Best Practices and
1208 Advice for Using Pupillometry to Measure Listening Effort: An Introduction for Those Who
1209 Want to Get Started. *Trends in Hearing*, 22, 2331216518800869.
1210 <https://doi.org/10.1177/2331216518800869>
- 1211 Winn, M., Moore, A. (2018). Pupillometry reveals that context benefit in speech
1212 perception can be disrupted by later-occurring sounds, especially in listeners with cochlear
1213 implants. *Trends in Hearing*, 22:2331216518808962. <https://doi.org/10.1177/2331216518808962>

1214

1215 Supplemental Figure: A demonstration of how `tidyr::gather` converts “wide” data with three
1216 separate object columns into “long” data that contains a “key” variable (Object) and a “value”
1217 variable (Fix).

1218

1219