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

|  |
| --- |
| Jason Geller1, Matthew B. Winn2, Tristian Mahr3, & Daniel Mirman4 |
| 1 University of Iowa |
| 2 University of Minnesota |
| 3 University of Wisconsin-Madison  4 University of Alabama at Birmingham |
|  |

# Author note

1. Department of Psychological & Brain Sciences

The University of Iowa

Iowa City, IA 52242

2) Department of Speech-Language-Hearing Sciences

164 Pillsbury Dr. SE

Minneapolis, MN 555455

3) Waisman Center  
 1500 Highland Ave  
 Madison, WI 53705

4) Department of Psychology

1300 University Blvd

Birmingham, AL 35205

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 difficult. To increase replicability 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 from the SR EyeLink eye tracker: 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:* 8,938

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. While there are many open-source tools for processing eye-tracking data, written in a variety of programming languages (e.g., R, Python, or MATLAB), they implement different processing conventions, some of which could be sub-optimal. In addition, some of these tools are not accessible to all users because they require proprietary or costly software (e.g., MATLAB). In the current climate where replicability and transparency are becoming more common, there is a need for a cross-platform, fully free implementation of standard practices in eye-tracking data processing. To this end, we have created the gazeR package in R (R Core Team 2018), which is a free, open-source statistical programming language, to aid researchers in analyzing eye-tracking data that comes from visual world paradigms and pupil dilation experiments. The package is implemented in R because it is the dominant environment for statistical analysis and visualization of eye-tracking data. Therefore, the gazeR package facilitates end-to-end analysis of eye-tracking data within a single programming environment – from reading in raw data files to statistical analysis and generating figures. The initial release version of gazeR is designed for use with the SR EyeLink eye-tracker, and extensions to other eye-trackers should be fairly straightforward.

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 limited to the analysis of a single subject and do not represent what researchers actually 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.

# Package Installation and Setup

## Raw Data

At the time of this writing, the gazeR package supports processing of data collected using an SR Research EyeLink eye tracker and exported using SR Research Data Viewer software, which generates a comma-separated text file consisting of either the full set of individual samples (“Sample Report”) or parsed fixations (“Fixation Report”).

## Package Installation

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

library(remotes)  
remotes::install\_github("dmirman/gazer") #installs package from github

Once this has been completed, gazeR can be installed by typing the following into the command line:

library(gazer)

library(tidyverse)  
library(zoo)  
library(knitr)

Once the gazeR package has been installed you are now ready to start preprocessing data!

# Preprocessing Gaze Position Data from the Visual World Paradigm

In a typical instantiation of the Visual World Paradigm (VWP), participants hear spoken instructions to manipulate or select one of several images on a computer screen or objects in the real world (Cooper, 1974; Tanenhaus, Spivey-Knowlton, Eberhard, & Sedivy, 1995). Decades of research have shown that the time course of fixation proportions – that is, the probability of fixating a particular object at a particular time – reflect the activation of that object’s mental 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).

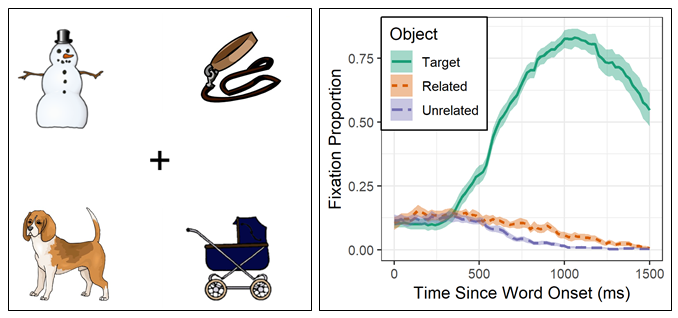


Figure 1. Left: Example display from a VWP experiment. The target is dog, the critical semantic competitor is leash (thematically related to the target), and snowman and carriage are unrelated distractors. Right: Example data showing the time course of target word recognition (soild line) and semantic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

Gaze preprocessing requires three main steps:

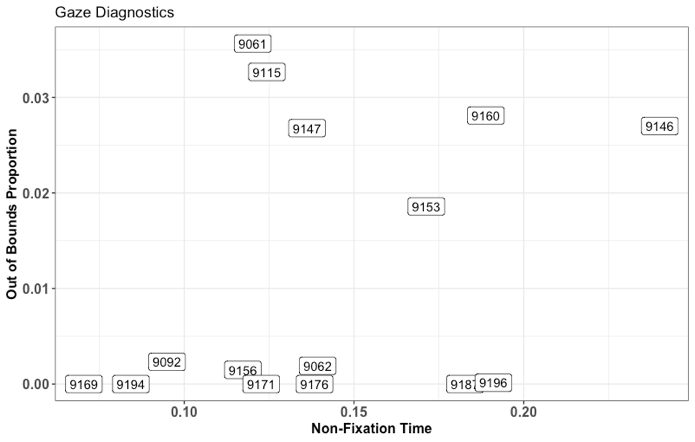
* 1. Reading in the data
  2. Assigning areas of interest
  3. Binning fixations

## Reading in Gaze Data

Gaze data need to be read from the Fixation Report file generated by the EyeLink Data Viewer application. The read\_fixation\_report() function will read in the fixation report file. By default, this function will also generate two plots: (1) a scatter plot showing participant-level proportion of time spent not in fixations and proportion of time spend with gaze outside the bounds of the screen (Figure 2, top), which can be used as calibration diagnostics; (2) scatter plots for each participant showing fixation positions and durations, along with a red rectangle that shows the screen edges (Figure 2, bottom), which can be used to check for any systematic calibration issues. A pdf file is generated for all the participants and is saved in your directory. The non-fixation and out-of-bounds proportions can also be calculated using get\_gaze\_diagnostics() function.

Example[[1]](#footnote-1)

gaze\_path <- system.file("extdata", "FixData\_v1\_N15.xls", package = "gazer")  
gaze <- read\_fixation\_report(gaze\_path, plot\_fix\_scatter = TRUE)



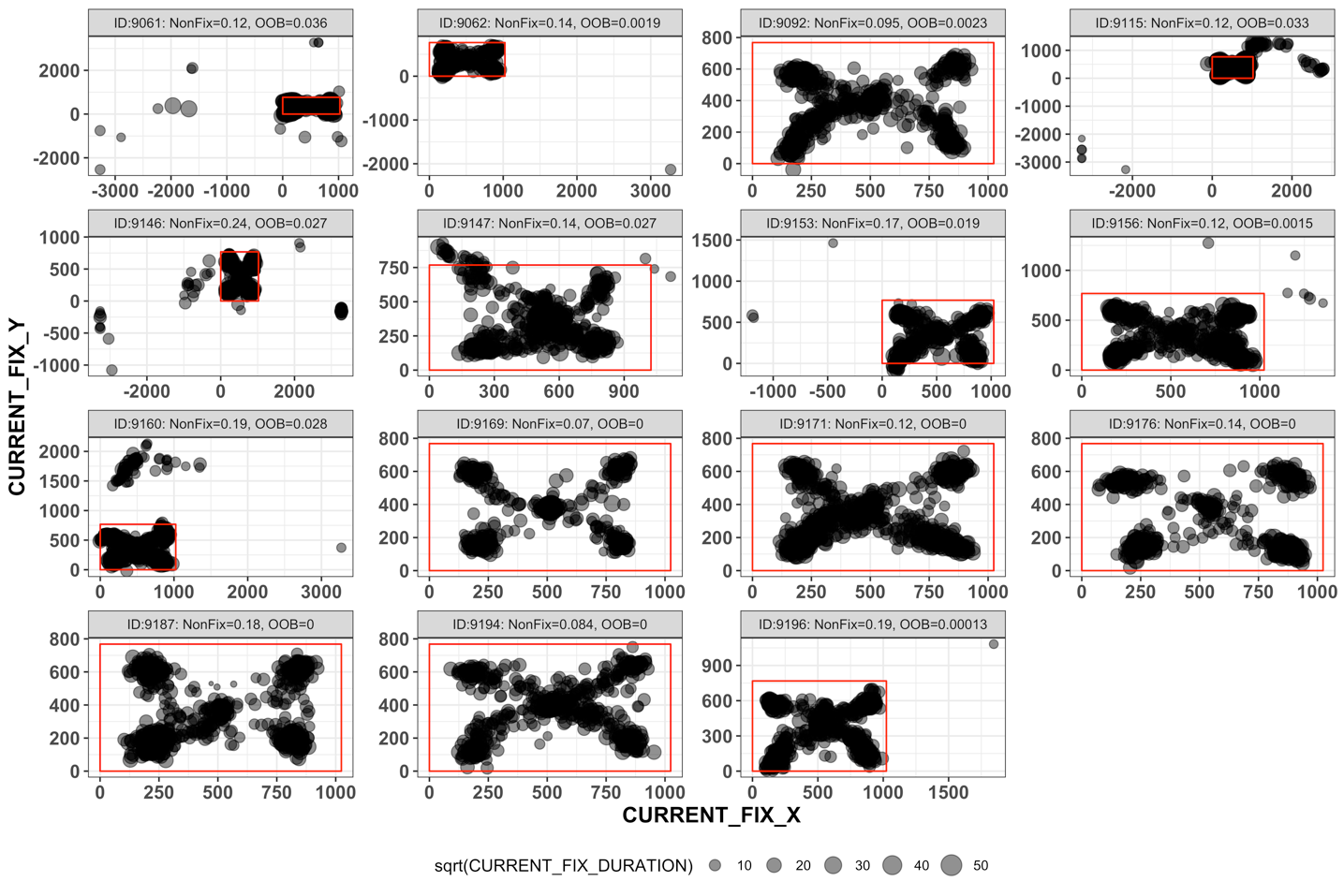


Figure 2. Plots generated when reading in fixation data. Top: gaze diagnostics. Horizontal axis is non-fixation time, vertical axis is proportion of looking time outside of screen boundaries. High values on these dimensions suggest possibly poor calibration or track quality. Bottom: scatterplots of fixation locations. Red rectangle indicates screen boundaries, circle size indicates fixation duration (square-root scaled so that perceptual effect of circle size better matches fixation duration). Most fixations should be in the corners (where the objects are) and the center cross. Systematic deviations or looks outside the suggest poor calibration.

For this example data set, the fixation report contains eye-tracking variables that are created by EyeLink (fixation duration, fixation position, pupil size, etc.) and experiment-specific values (positions of different objects, trial condition, participant accuracy and response time) that are provided by the experiment software (in this case, E-Prime).

Table 1. Visual World Data Description and Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Class | Contents | Source |
| Subject | integer | Label of the data file | SR Eyelink |
| CURRENT\_FIX\_PUPIL | double | Pupil size of the current fixation | SR Eyelink |
| CURRENT\_FIX\_DURATION | integer | Duration of the current fixation | SR Eyelink |
| CURRENT\_FIX\_END | integer | Trial time when the current fixation ends | SR Eyelink |
| CURRENT\_FIX\_START | integer | Trial time when the current fixation starts | SR Eyelink |
| CURRENT\_FIX\_X | double | X coordinate of the current fixation | SR Eyelink |
| CURRENT\_FIX\_Y | double | Y coordinate of the current fixation | SR Eyelink |
| CompPort | integer | Screen location of Competitor image | E-Prime |
| Condition | integer | Trial condition (practice, associate, filler, taxonomic) | E-Prime |
| TargetLoc | integer | Screen location of Target image | E-Prime |

summary(gaze)

## Subject CURRENT\_FIX\_PUPIL CURRENT\_FIX\_DURATION CURRENT\_FIX\_END   
## 9160 :1109 Min. : 36.0 Min. : 2.0 Min. : 22.0   
## 9196 : 897 1st Qu.: 122.0 1st Qu.: 140.0 1st Qu.: 919.5   
## 9115 : 882 Median : 165.0 Median : 210.0 Median : 1886.0   
## 9187 : 839 Mean : 176.3 Mean : 279.6 Mean : 1958.0   
## 9061 : 787 3rd Qu.: 201.0 3rd Qu.: 328.0 3rd Qu.: 2614.5   
## 9171 : 786 Max. :9144.0 Max. :2660.0 Max. :26184.0   
## (Other):5616   
## CURRENT\_FIX\_START CURRENT\_FIX\_X CURRENT\_FIX\_Y CompPort   
## Min. : 4 Min. :-3270.0 Min. :-3270.0 image1:2794   
## 1st Qu.: 650 1st Qu.: 234.5 1st Qu.: 173.5 image2:2762   
## Median : 1562 Median : 510.9 Median : 362.7 image3:2716   
## Mean : 1680 Mean : 510.5 Mean : 354.0 image4:2644   
## 3rd Qu.: 2334 3rd Qu.: 799.7 3rd Qu.: 522.7   
## Max. :25848 Max. : 3270.0 Max. : 3270.0   
##   
## Condition TargetLoc ACC RT   
## associate:3059 image1:2769 Min. :0.0000 Min. : 2236   
## filler :3010 image2:2891 1st Qu.:1.0000 1st Qu.: 2957   
## practice :1702 image3:2611 Median :1.0000 Median : 3237   
## taxonomic:3145 image4:2645 Mean :0.9898 Mean : 3631   
## 3rd Qu.:1.0000 3rd Qu.: 3687   
## Max. :1.0000 Max. :26105   
##   
## Target TargetLocation  
## barn : 213 1:2769   
## walker : 194 2:2891   
## acorn : 184 3:2611   
## bandaid: 184 4:2645   
## pillow : 181   
## falcon : 180   
## (Other):9780

## Parsing areas of interest

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

First, extract the numbered location of the target and competitor in order to match the output of the assign\_aoi function, which will assign a numbered area of interest for each fixation that falls within a defined area of interest (by default, 400x300 rectangles in the corners of the screen). This sub-step is somewhat specific to how image locations were labeled in this particular experiment, where the image location is the 6th character in the location string (e.g., image2), so that is the value that needs to be extracted:

gaze$TargetLocation <- as.numeric(substr(gaze$TargetLoc, 6, 6))  
gaze$CompLocation <- as.numeric(substr(gaze$CompPort, 6, 6))

Then match fixation locations to areas of interest (AOI) based on screen coordinates:

gaze\_aoi <- assign\_aoi(gaze)  
summary(gaze\_aoi)

## Subject CURRENT\_FIX\_PUPIL CURRENT\_FIX\_DURATION CURRENT\_FIX\_END   
## 9160 :1109 Min. : 36.0 Min. : 2.0 Min. : 22.0   
## 9196 : 897 1st Qu.: 122.0 1st Qu.: 140.0 1st Qu.: 919.5   
## 9115 : 882 Median : 165.0 Median : 210.0 Median : 1886.0   
## 9187 : 839 Mean : 176.3 Mean : 279.6 Mean : 1958.0   
## 9061 : 787 3rd Qu.: 201.0 3rd Qu.: 328.0 3rd Qu.: 2614.5   
## 9171 : 786 Max. :9144.0 Max. :2660.0 Max. :26184.0   
## (Other):5616   
## CURRENT\_FIX\_START CURRENT\_FIX\_X CURRENT\_FIX\_Y CompPort   
## Min. : 4 Min. :-3270.0 Min. :-3270.0 image1:2794   
## 1st Qu.: 650 1st Qu.: 234.5 1st Qu.: 173.5 image2:2762   
## Median : 1562 Median : 510.9 Median : 362.7 image3:2716   
## Mean : 1680 Mean : 510.5 Mean : 354.0 image4:2644   
## 3rd Qu.: 2334 3rd Qu.: 799.7 3rd Qu.: 522.7   
## Max. :25848 Max. : 3270.0 Max. : 3270.0   
##   
## Condition TargetLoc ACC RT   
## associate:3059 image1:2769 Min. :0.0000 Min. : 2236   
## filler :3010 image2:2891 1st Qu.:1.0000 1st Qu.: 2957   
## practice :1702 image3:2611 Median :1.0000 Median : 3237   
## taxonomic:3145 image4:2645 Mean :0.9898 Mean : 3631   
## 3rd Qu.:1.0000 3rd Qu.: 3687   
## Max. :1.0000 Max. :26105   
##   
## Target TargetLocation CompLocation AOI   
## barn : 213 Min. :1.00 Min. :1.000 Min. :0.000   
## walker : 194 1st Qu.:1.00 1st Qu.:1.000 1st Qu.:0.000   
## acorn : 184 Median :2.00 Median :2.000 Median :2.000   
## bandaid: 184 Mean :2.47 Mean :2.477 Mean :1.721   
## pillow : 181 3rd Qu.:3.00 3rd Qu.:3.000 3rd Qu.:3.000   
## falcon : 180 Max. :4.00 Max. :4.000 Max. :4.000   
## (Other):9780 NA's :1040

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

gaze\_aoi$Targ <- gaze\_aoi$AOI == gaze\_aoi$TargetLocation  
gaze\_aoi$Comp <- gaze\_aoi$AOI == gaze\_aoi$CompLocation  
gaze\_aoi$Unrelated <-   
 ((gaze\_aoi$AOI != as.numeric(gaze\_aoi$TargetLocation)) &  
 (gaze\_aoi$AOI != as.numeric(gaze\_aoi$CompLocation)) &  
 (gaze\_aoi$AOI != 0) & !is.na(gaze\_aoi$AOI))

## Fixations to bins

Fixations can start and end at any time point, but most analysis strategies require aligned, equally-spaced time bins. The binify\_fixations function will unpack the set of fixations into a fixation time series consisting of standardized time bins with a size specified by the user (default is 20ms). In addition, it will drop columns that are no longer necessary – the fixation start and end time and duration will no longer be needed, nor will the gaze position coordinates, since gaze position has now been recoded from coordinates to objects. The user needs to specify a list columns that should be kept after the binning is done. Converting fixations to bins can be somewhat slow.

## gaze\_bins <- binify\_fixations( gaze = gaze\_aoi, keepCols = c("Subject", "Target", "Condition", "ACC", "RT", "Targ", "Comp", "Unrelated"))

## Aggregate Data

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

gaze\_obj <- gather(gaze\_bins,   
 key = "Object", value = "Fix",   
 Targ, Comp, Unrelated, factor\_key = TRUE)  
# recode NA as not-fixating  
gaze\_obj$Fix <- replace(gaze\_obj$Fix, is.na(gaze\_obj$Fix), FALSE)   
summary(gaze\_obj)

## FixationID timeBin Subject Target   
## Min. : 1 Min. : 1.00 9115 : 43680 barn : 9552   
## 1st Qu.: 2732 1st Qu.: 45.00 9160 : 38553 walker : 8283   
## Median : 5295 Median : 88.00 9061 : 36645 bandaid : 8256   
## Mean : 5458 Mean : 95.09 9156 : 35202 acorn : 8019   
## 3rd Qu.: 8293 3rd Qu.: 130.00 9171 : 32793 soda : 7926   
## Max. :10916 Max. :1310.00 9092 : 32289 paintbrush: 7839   
## (Other):265791 (Other) :435078   
## Condition ACC RT Time   
## associate:135507 Min. :0.0000 Min. : 2236 Min. : 20   
## filler :135375 1st Qu.:1.0000 1st Qu.: 2947 1st Qu.: 900   
## practice : 75618 Median :1.0000 Median : 3229 Median : 1760   
## taxonomic:138453 Mean :0.9895 Mean : 3641 Mean : 1902   
## 3rd Qu.:1.0000 3rd Qu.: 3673 3rd Qu.: 2600   
## Max. :1.0000 Max. :26105 Max. :26200   
##   
## Object Fix   
## Targ :161651 Mode :logical   
## Comp :161651 FALSE:379285   
## Unrelated:161651 TRUE :105668

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 bin to calculate the number of object fixations and mean fixation proportion in each time bin; that is, the time course of fixation. These are the subject-by-condition time courses that would go into an analysis.

gaze\_subj <- gaze\_obj %>%

# keep only correct-response trials, exclude practice condition, and analyze time points only up to 3500ms after trial onset  
 filter(ACC == 1, Condition != "practice", Time < 3500) %>%   
 # calculate number of valid trials for each subject-condition  
 group\_by(Subject, Condition, Object) %>% # for every unique combination of Subject, Condition, and Object…  
 mutate(nTrials = length(unique(Target))) %>% # count the number of trials

ungroup() %>%  
 # calculate number of fixations counts and proportions  
 group\_by(Subject, Condition, Object, Time) %>% # for every unique combination of Subject, Condition, and Object in each time bin  
 summarize(sumFix = sum(Fix), # number of fixations

nTrials = unique(nTrials), # number of trials  
 meanFix = sum(Fix)/unique(nTrials)) # fixation proportion  
# there were two unrelated objects, so divide those proportions by 2  
gaze\_subj$meanFix[gaze\_subj$Object == "Unrelated"] <-   
 gaze\_subj$meanFix[gaze\_subj$Object == "Unrelated"] / 2  
summary(gaze\_subj)

## Subject Condition Object Time   
## 9061 : 1566 associate:7800 Targ :7790 Min. : 20   
## 9062 : 1566 filler :7758 Comp :7790 1st Qu.: 880   
## 9092 : 1566 practice : 0 Unrelated:7790 Median :1740   
## 9115 : 1566 taxonomic:7812 Mean :1742   
## 9146 : 1566 3rd Qu.:2600   
## 9153 : 1566 Max. :3480   
## (Other):13974   
## sumFix nTrials meanFix   
## Min. : 0.000 Min. :19.00 Min. :0.00000   
## 1st Qu.: 0.000 1st Qu.:20.00 1st Qu.:0.00000   
## Median : 2.000 Median :20.00 Median :0.07895   
## Mean : 3.495 Mean :19.87 Mean :0.15186   
## 3rd Qu.: 5.000 3rd Qu.:20.00 3rd Qu.:0.20000   
## Max. :20.000 Max. :20.00 Max. :1.00000   
##

## Plot fixation time course

After the fixations have been assigned to the object type and converted to time bins, they are ready for visualization and statistical analysis. Below is a plot of the time course of fixation proportions for each target type.

ggplot(gaze\_subj, aes(Time, meanFix, color = Object)) +   
 facet\_wrap(~ Condition) +  
 stat\_summary(fun.y = mean, geom = "line") +  
 geom\_vline(xintercept = 1300) +  
 annotate("text", x=1300, y=0.9, label="Word onset", hjust=0)

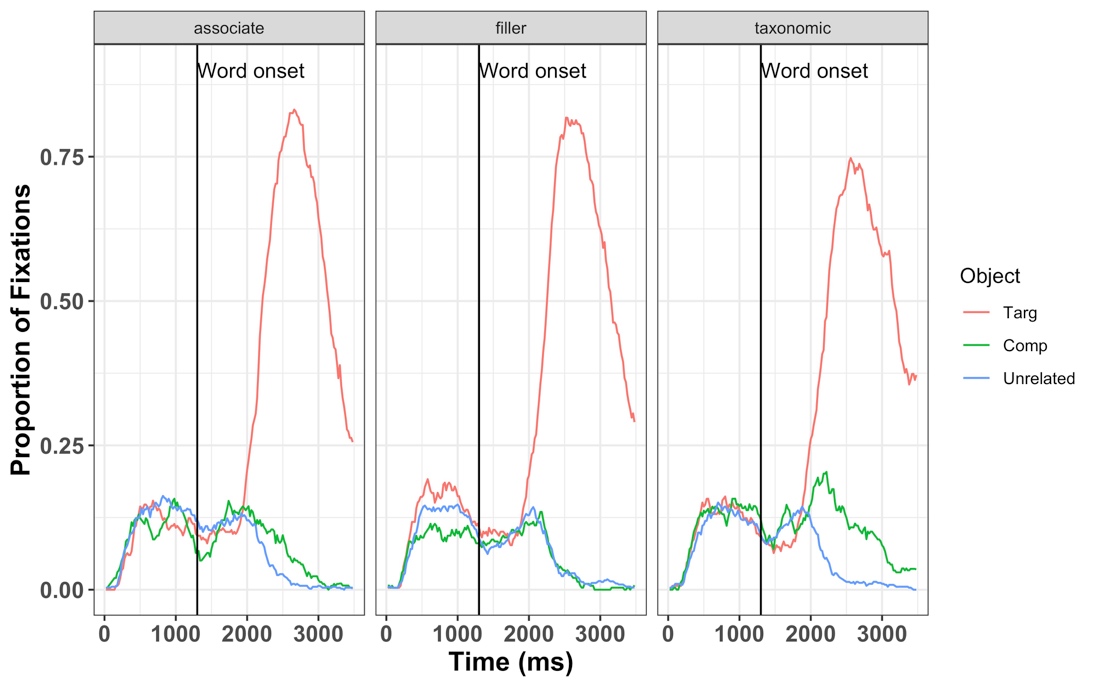


Figure 3. Time course of fixation proportions by condition. These data have been pre-processed and are ready for statistical analysis.

## Preprocessing Pupil Data from a Lexical Decison 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, the number of studies employing pupillometry has grown exponentially since the first modern boom more than a half a century ago (Kret & Sjak-Shie, 2018). The reason for this is quite simple: pupil size has been shown to be a reliable and valid index of mental effort or arousal across many domains, including word recognition (Geller, Still, & Morris, 2016), normal and impaired auditory perception (Zekveld et al., 2018), attention allocation (Karatekin, Couperus, & Marcus, 2004), working memory load (Granholm, Asarnow, Sarkin, & Dykes, 1996; Van Gerven, Paas, Van Merriënboer, & Schmidt, 2004), face perception (Goldinger, He, and Papesh, 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 walkthroughs 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.

To demonstrate analysis of pupil data, we will be using an example data set containing data from a lexical decision task. In this task, participants (*N*=41) judged the lexicality 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 (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
  + Interpolation
  1. Smoothing
  2. Baseline correction
  3. Re-scaling

1. Artifact Rejection
   * Missing data
   * Unlikely pupil values
   * Median absolute deviation (MAD)
2. Trial Clipping
3. Decimating/Downsampling
4. Aggregation

## Reading in Pupil Data

In order for the pupil functions to work properly, the Sample Report must be generated with the columns below. The functions will not work if these columns are not present in the Sample Report. Other columns should be included if needed.

Table 1. Variables Needed to Process Pupil Data

|  |
| --- |
| Names |
| RECORDING\_SESSION\_LABEL |
| TRIAL\_INDEX |
| AVERAGE\_IN\_BLINK, RIGHT\_IN\_BLKINK, or LEFT\_IN\_BLINK |
| TIMESTAMP |
| AVERAGE\_PUPIL\_SIZE, RIGHT\_PUPIL\_SIZE, or LEFT\_PUPIL SIZE |
| IP\_START\_TIME |
| SAMPLE\_MESSAGE |

If you generated separate sample reports for each participant, the function merge\_pupil will take all your pupil files from a folder path and merge them together. It will also rename variables, make all variable names lowercase, and add a new column, time, which places time in ms instead of tracker time. You must first specify a list of pupil data files, then you can call the merge\_pupil function to aggregate your data. Depending on the number of subjects and the sampling rate at experiment runtime, this could take a few minutes. There are two arguments, blink\_colname and pupil\_colname. It is important you specify what these variables are called in your data set so the pipeline runs smoothly. In our example dataset, we used the AVERAGE\_IN\_BLINK and AVERAGE\_PUPIL\_SIZE columns.

# where to find all your pupil files  
file\_list <- list.files(path = '', pattern = ".xls")

pupil\_files <- merge\_pupil(  
 file\_list,   
 blink\_colname = “AVERAGE\_IN\_BLINK”,   
 pupil\_colname = “AVERAGE\_PUPIL\_SIZE”  
)

Due to processing constraints, we are using a Sample Report that includes data from a few participants. If you would like to try out the merge\_pupil function you can download all the participant files on Open Science Framework (OSF) here: <https://osf.io/fzu38/>. While reading in the data is pretty fast (even with many participants), some of the functions performed on the data can be computationally intensive.

#download Sample Report from Github

pupil\_path <- system.file("extdata", "Pupil\_file1.xls", package = "gazer")

#read in data  
pupil\_files <- read.table(pupil\_path)

Table 3. Pupil Data Description and Structure

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Class | Contents | Source |
| subject | integer | Label of the data file | SR Eyelink |
| trial | integer | Trial number | SR Eyelink |
| blink | integer | Whether eye was in blink | SR Eyelink |
| pupil | integer | pupil size on the current sample | SR Eyelink |
| accuracy | integer | 0=incorrect; 1=correct | SR Eyelink |
| cb | integer | counterbalance list | SR Eyelink |
| key\_pressed | integer | response made | SR Eyelink |
| rt | integer | condition (word, nonword transposed letter,  2L substition nonword) | SR Eyelink |
| alteration | integer | Trial condition (practice, associate, filler, taxonomic) | SR Eyelink |
| block | integer | Block number | SR Eyelink |
| item | character | item presented | SR Eyelink |
| response | integer | button pressed | SR Eyelink |
| script | integer | condition (cursive, type-print) | SR Eyelink |
| target | character | eye in saccade | SR Eyelink |
| average\_in\_saccade | integer | Start time of the interest period | SR Eyelink |
| ip\_start\_time | integer | Start time (in milliseconds since EyeLink tracker was Eyelink | SR Eyelink |
| sample\_message | character | Message text printed out during current sample | SR Eyelink |
| timestamp | integer | Time lapsed (in milliseconds) since eye-tracker started | SR Eyelink |
| time | integer | ip\_start\_time - timestamp | SR Eyelink |

# Behavioral Data (Optional)

If you are also interested in analyzing behavioral data (RTs and accuracy), the behave\_data function will cull the important behavioral data from the Sample Report. The function will return a data frame without errors when omiterrors=TRUE or a data frame with errors for accuracy/error analysis when omiterrors=FALSE. The columns relevant for your experiment need to be specified within the behave\_col names argument. This function does not eliminate outliers; you must use your preferred method. Grange’s (2015) trimr package implements multiple standard methods of outlier exclusion (<https://github.com/JimGrange/trimr>).

## subject script alteration trial target accuracy rt block cb  
## 1 10b print word 1 sprigp.png 1 2539 0 2  
## 960 10b cursive nwtl 2 nypmh.png 1 3254 0 2  
## 2117 10b Cursive nwtl 3 seivep.png 0 1755 0 2  
## 2882 10b cursive word 4 mourn.png 1 2435 0 2  
## 3821 10b Cursive word 5 noisy.png 1 2200 1 2  
## 5197 10b Cursive word 6 ridge.png 1 1952 1 2

For this example, we will exclude participants with overall accuracy lower than 75% and items with accuracy below 60%. Using the file generated above with omiterrors=FALSE, we can calculate subject and item accuracy, merge those values into the main data set, and use them as exclusion criteria.

Itemacc <- behave\_data %>%   
 group\_by(target) %>%   
 summarise(  
 # overall item accuracy and word condition only  
 meanitemacc = mean(accuracy[block>0 & alteration=="word"])  
 )   
   
subacc <- behave\_data %>%   
 group\_by(subject) %>%   
 summarise(  
 #subject accuracy and word condition only  
 meansubacc = mean(accuracy[block > 0 & alteration == "word"])  
 )   
   
dataraw1 <- merge(pupil\_files, itemacc) # merge into main ds  
dataraw2 <- merge(dataraw1, subacc) # merge into main ds

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

pupil\_files1 <- dataraw2 %>%

*# filter out practice blocks, incorrect responses, nonword trials, low item and subj acc*  
 filter(  
 block > 0, accuracy == 1, alteration == "word",   
 meanitemacc >= .60, meansubacc >= .75  
 ) %>%   
 arrange(subject, target, trial, time)

Pupil Preprocessing is now ready to begin!

## De-blinking

An imporatant first step in preprocessing pupil data is de-blinking. A major artifact in pupil data comes from blinking. When the eye blinks, the pupil momentarily becomes smaller as it is occluded more and more by the eyelids, making computing the center of the pupil difficult. Eye-trackers interpret this as a fast shift in pupil position and will classify it as a saccade. Additionally, the estimate of pupil size will rapidly decrease as the pupil occupies less of the camera image. This process happens in reverse (albeit a bit more slowly) as the eye is opening, so blinks are always flanked by a saccade artifact. 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 them, and then interpolating data during the blink period and even across a longer segment that extends before and after the blink. Identifying blinks is rather trivial as the EyeLink records contain a blink column with 0s or 1s denoting absence or presence of a blink. Less trivial is deciding how many data points you remove before and after the blink. It has generally been recommended that data 100 ms before and after the blink should be eliminated. The gazeR package contains several functions for dealing with blinks. If you are exporting files from SR, there is an option to extend blinks within Data Viewer. 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 after the blink (Nyström, Hooge, & Andersson, 2016; Satterthwaite et al., 2007). Extending the time window around the blinks eliminates spurious samples caused by the closing and opening of the eyelids. If you have not done this before exporting into R, you can use the extend\_blinks function. The fillback argument extends blinks back in time and the fillforward argument extends blinks forward in time. This function is robust to different sampling rates — make sure you specify the tracker sampling rate in the hz argument. For this experiment, the tracker sampled at 250Hz (once every 4 ms) and blinks were extended 100 ms forward and backward in time.

pup\_extend<- pup\_files1 %>%   
 group\_by(subject, trial) %>%   
 mutate(extendpupil=extend\_blinks(pupil, fillback=100, fillforward=100, hz=250))

## Interpolation

## Missing data stemming from blinks or failure of the eye tracker need to be interpolated. The interpolate\_pupil function searches the data and reconstructs the 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 “NA”s and are then interpolated linearly or by cubic interpolation. This function returns a tibble with a column called interp which contains interpolated values from the pupil column in your data (e.g., average, left, or right pupil size). As an important note, if the Data Viewer was used to extend blinks, the extendblinks argument should be set to FALSE. If gazer::extend\_blinks 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.”

pup\_interp <- interpolate\_pupil(  
 pup\_extend,   
 extendblinks = TRUE,   
 type = "linear")

## Performing linear interpolation

It is a good idea to check that the interpolation did what it was supposed to do. The plot below shows data from one trial with artifacts removed, the observed data are shown in black and the interpolated data are shown in green. Looks good!

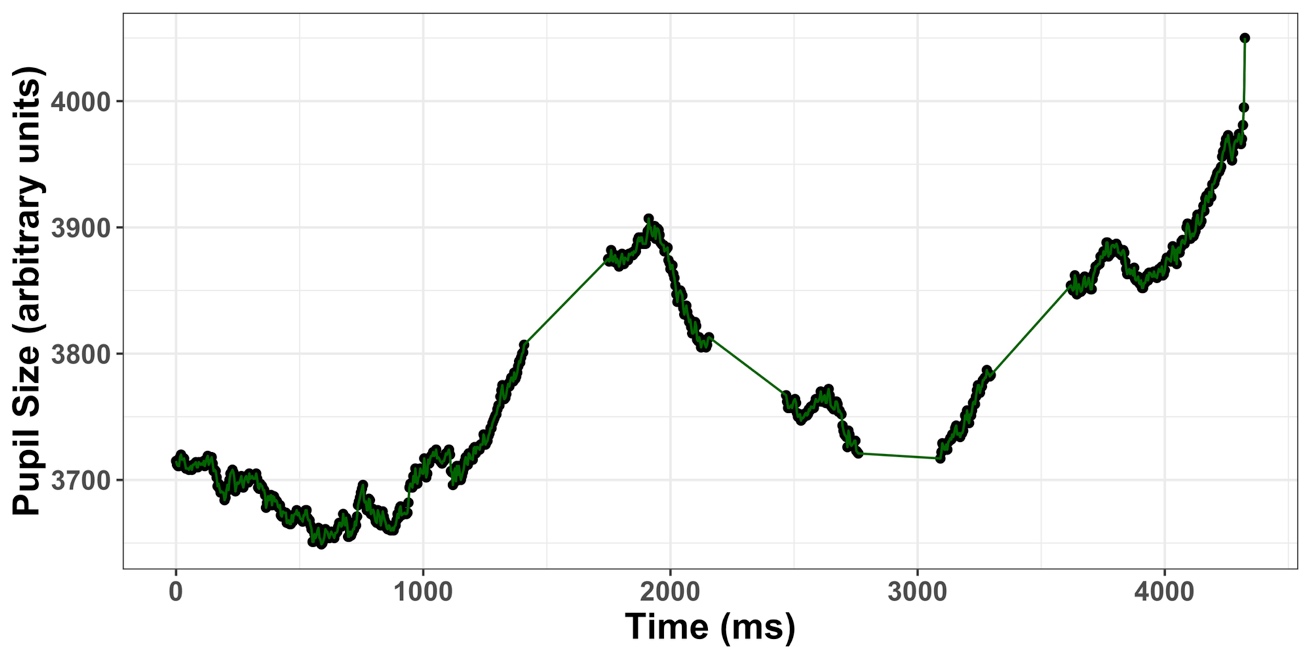


Figure 4. Linear interpolation for one trial

## Smoothing

Pupil data can be extremely noisy! There are many ways to smooth pupil data. Two common methods are implemented in gazeR: n-point moving average and a hanning filter. To smooth the data using a n-point moving average, call the moving\_average\_pupil function, and specify the column that contains the interpolated pupil values and the size (in samples) of the moving average window. In this example, we use a 5-point moving average (n=5). The variable movingavgpup is returned with the smoothed pupil data. Low-pass filtering is something that might be included in a future update to the package.

rolling\_mean\_pupil\_average <- as.data.frame(pup\_interp) %>% #must be in a data.frame  
 select(  
 subject, trial, target, pupil, script, alteration,   
 time, interp, sample\_message  
 ) %>%  
  
 mutate(movingavgpup = moving\_average\_pupil(interp, n = 5))

## 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 (pupil size / 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 supported on the basis of a study by Reilly, Kelly, Kim, Jett, and Zuckerman (2018) that argued for linearity of the pupil response, independent of baseline size[[2]](#footnote-2). 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\_colnames = "movingavgpup") though the same can be done for pupil size in mm or *z*-score. The baseline window is the 500ms immediately preceding stimulus onset, which in this study is 500-1000ms after trial onset.

baseline\_pupil <- baseline\_correction\_pupil(  
 rolling\_mean\_pupil\_average,   
 pupil\_colnames = "movingavgpup",   
 baseline\_window = c(500, 1000),

baseline\_method = ‘sub’  
)  
## Calculating baseline

## Calculating median baseline from:500-1000

## Merging baseline

## Performing subtractive baseline correction

baseline\_pupil

## # A tibble: 11,031 x 11  
## # Groups: subject, trial, time [11,031]  
## subject trial time baseline target script alteration interp  
## <fct> <int> <int> <dbl> <fct> <fct> <fct> <dbl>  
## 1 10b 5 680 4130. noisy… Cursi… word 4373  
## 2 10b 5 684 4253. noisy… Cursi… word 4375  
## 3 10b 5 688 4379. noisy… Cursi… word 4374  
## 4 10b 5 692 4382. noisy… Cursi… word 4382  
## 5 10b 5 696 4386 noisy… Cursi… word 4389  
## 6 10b 5 700 4390. noisy… Cursi… word 4392  
## 7 10b 5 704 4395 noisy… Cursi… word 4393  
## 8 10b 5 708 4399. noisy… Cursi… word 4396  
## 9 10b 5 712 4403. noisy… Cursi… word 4405  
## 10 10b 5 716 4407 noisy… Cursi… word 4408  
## # … with 11,021 more rows, and 3 more variables: sample\_message <fct>,  
## # pupil1 <dbl>, baselinecorrectedp <dbl>

## Re-Scaling

So far, the analysis steps have used arbitrary pupil units. It is advised that these be transformed into a standardized unit in order to make comparisons between individuals. Among the numerous options that have been used, there are z-scores (see Cohen, Moyal, & Henik, 2015; Einhauser, Stout, Koch, & Carter, 2008; Kang & Wheatley, 2015), absolute changes in mm (e.g., Beatty, 1982; Geller, Landrigan, & Mirman, 2019; Geller et al., 2016), proportional change relative to baseline (Winn, 2016), and absolute change relative to dynamic range of pupil reactivity elicited by the light reflex (Piquado, Isaacowitz, & Wingfield, 2010). To convert arbitrary pupil size to mm, we measured the scaling factor by running a short experiment with an artificial pupil (5 mm in size) and calculated the average pupil size in arbitrary units. At a fixed camera-to-pupil distance of 90 cm, the 5mm pupil was coded as 5570.29 arbitrary pixel units. This information was entered into the equation below to convert arbitrary units to mm. Specifically, the smoothed pupil size value is multiplied by 5/5570.29 to re-scale the values to mm.

timebinsmm <- rolling\_mean\_pupil\_average %>%   
 mutate(pupilmm = (movingavgpup \* 5)/5570.29)

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

timebinsz<- rolling\_mean\_pupil\_average %>%   
 group\_by(subject, trial) %>%  
 mutate(pupilz = scale(movingavgpup))

## Artifact Rejection

**Missingness**. The count\_missing\_pupil function will remove subjects and items that have a large amount of missing data – the threshold for “a large amount” is specified by the researcher. It has been recommended by Winn et al. (2018) that a reasonable threshold is 20%, but that the exact importance of missing data might be weighted by specific timing landmarks in the experiment trials. For this example, we have set the missingthresh argument to .2. The count\_missing\_pupil() function returns the percentage of subjects and trials that have been excluded for reporting.

pup\_missing <- count\_missing\_pupil (baseline\_pupil, missingthresh = .2)

## % trials excluded:0.011

## subjects taken out:

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

puphist <- ggplot(pup\_extend, aes(x = extendpupil)) +

geom\_histogram(aes(y = ..count..), colour = "green", binwidth = 0.5) +   
 geom\_vline(xintercept = 2500, linetype="dotted") +

geom\_vline(xintercept = 5100, linetype="dotted") +   
 xlab("Pupil Size") +   
 ylab("Count") +   
 theme\_bw()

print(puphist)

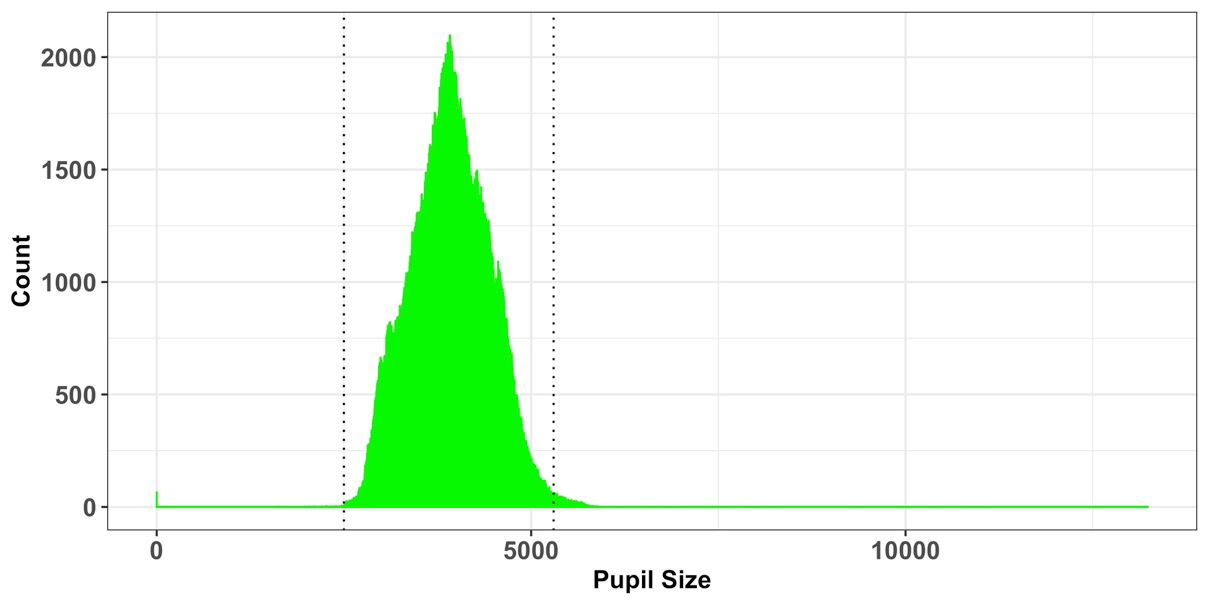


Figure 5. Histogram of recorded pupil sizes throughout experiment for all 41 participants.

pup\_outliers <- pup\_missing %>%   
 # based on visual inspection  
 dplyr::filter(interp >= 2500, interp <= 5100)

## Median absolute deviation (MAD). After interpolation, it is a good idea to perfrom a second pass on your data to make sure that the data is not contaminated by rapid pupil size disturbances. These artifacts can be detected using the median absolute deviation (Kret & Sjak-Shie, 2018). The speed\_dilation 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 median absolute deviation is calculated from the speed dilation variable, multiplied by a constant (in this case 16), and 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(interp,time)) %>%   
 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 below that our experiment now starts at zero, when the target was displayed on screen.

baseline\_pupil\_onset <- baseline\_pupil %>%   
 group\_by(subject, trial) %>%   
 mutate(  
 time\_zero = onset\_pupil (time, sample\_message, event = c("target"))  
 ) %>%  
 ungroup() %>%   
 filter(time\_zero >= 0, time\_zero <= 3000) %>%  
 select(  
 subject, trial, time, script, time\_zero,   
 sample\_message, baselinecorrectedp  
 )

baseline\_pupil\_onset

## # A tibble: 66,126 x 7  
## subject trial time script time\_zero sample\_message baselinecorrectedp  
## <fct> <int> <int> <fct> <int> <fct> <dbl>  
## 1 10b 11 348 Cursive 0 target -11.9  
## 2 10b 11 352 Cursive 4 <NA> -15.5  
## 3 10b 11 356 Cursive 8 <NA> -19.1  
## 4 10b 11 360 Cursive 12 <NA> -24.1  
## 5 10b 11 364 Cursive 16 <NA> -28.5  
## 6 10b 11 368 Cursive 20 <NA> -32.1  
## 7 10b 11 372 Cursive 24 <NA> -34.5  
## 8 10b 11 376 Cursive 28 <NA> -35.7  
## 9 10b 11 380 Cursive 32 <NA> -35.9  
## 10 10b 11 384 Cursive 36 <NA> -37.5

## Downsampling/Decimation

If the data are recorded at a relatively high sampling frequency (e.g., 250Hz in this example), it may be useful to aggregate the the data into time bins that are somewhat larger than the sample rate (users can specify a time bin size to use). The downsample\_pupil function takes your data and a specified bin length (in ms) as arguments and returns a tibble with a column called timebins.

timebins1 <- downsample\_pupil(baseline\_pupil\_onset, bin.length=200)  
  
timebins1

## # A tibble: 66,126 x 8  
## subject trial time script time\_zero sample\_message baselinecorrect…  
## <fct> <int> <int> <fct> <int> <fct> <dbl>  
## 1 10b 11 348 Cursi… 0 target -11.9  
## 2 10b 11 352 Cursi… 4 <NA> -15.5  
## 3 10b 11 356 Cursi… 8 <NA> -19.1  
## 4 10b 11 360 Cursi… 12 <NA> -24.1  
## 5 10b 11 364 Cursi… 16 <NA> -28.5  
## 6 10b 11 368 Cursi… 20 <NA> -32.1  
## 7 10b 11 372 Cursi… 24 <NA> -34.5  
## 8 10b 11 376 Cursi… 28 <NA> -35.7  
## 9 10b 11 380 Cursi… 32 <NA> -35.9  
## 10 10b 11 384 Cursi… 36 <NA> -37.5  
## # … with 66,116 more rows, and 1 more variable: timebins <dbl>

## Aggregating Data

To further simplify the data, they can be aggregated to produce an average pupil diameter for each subject in each condition at each time bin.

agg\_subject<- timebins1 %>%   
 dplyr::group\_by(subject, script,timebins) %>%

dplyr::summarise(aggbaseline=mean(baselinecorrectedp)) %>%   
 ungroup()  
  
## # A tibble: 80 x 4  
## subject script timebins aggbaseline  
## <fct> <fct> <dbl> <dbl>  
## 1 10b Cursive 0 16.0   
## 2 10b Cursive 200 3.03  
## 3 10b Cursive 400 -3.92  
## 4 10b Cursive 600 10.8   
## 5 10b Cursive 800 38.8   
## 6 10b Cursive 1000 74.8   
## 7 10b Cursive 1200 102.   
## 8 10b Cursive 1400 113.   
## 9 10b Cursive 1600 114.

## Pupillary Data Visualization

After baseline-correction and aggregation, 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. Below is a plot of the time course for the baseline-corrected pupillary response between cursive and type-print stimuli. A cursory look suggests that that recognizing cursive words resulted in a larger pupillary response at around 1600-2500ms.

data(cursive\_new)

## # A tibble: 6 x 4  
## subject script timebins aggbaseline  
## <chr> <chr> <dbl> <dbl>  
## 1 10b cursive 0 15.7   
## 2 10b cursive 200 3.14  
## 3 10b cursive 400 -4.53  
## 4 10b cursive 600 6.63  
## 5 10b cursive 800 34.6   
## 6 10b cursive 1000 73.8

runningSE <- cursive\_new %>%  
 filter(timebins <= 3500) %>%   
 split(.$timebins) %>%   
 map(~Rmisc::summarySEwithin(data = ., measurevar = "aggbaseline", withinvars = "script", idvar="subject"))  
  
cur1 <- filter(cursive\_new, timebins <= 3500)  
  
WSCI <- map\_df(runningSE, extract) %>%  
 mutate(Time = rep(unique(cur1$timebins), each = 2))  
 #Note, you'll have to change 2 to match the number of conditions  
  
WSCI.plot <- ggplot(WSCI) + geom\_line(aes(Time, aggbaseline, linetype=script, color=script), size=3) +  
 theme\_bw() +  
 labs(x = "Time (ms)",y = "Baseline-corrected pupil size (a.u)") +  
 geom\_hline(yintercept = 0,linetype = "dashed") +  
 geom\_ribbon(data = WSCI, aes(x=Time, ymin = aggbaseline-ci, ymax = aggbaseline+ci, linetype=script, colour=script), alpha = 0.3) +  
 theme(axis.title.y=element\_text(size = 14, face="bold"), axis.title.x = element\_text(size=14, face="bold"), axis.text.x=element\_text(size = 12, face="bold"),axis.text.y=element\_text(size=12, face="bold"))

WSCI.plot

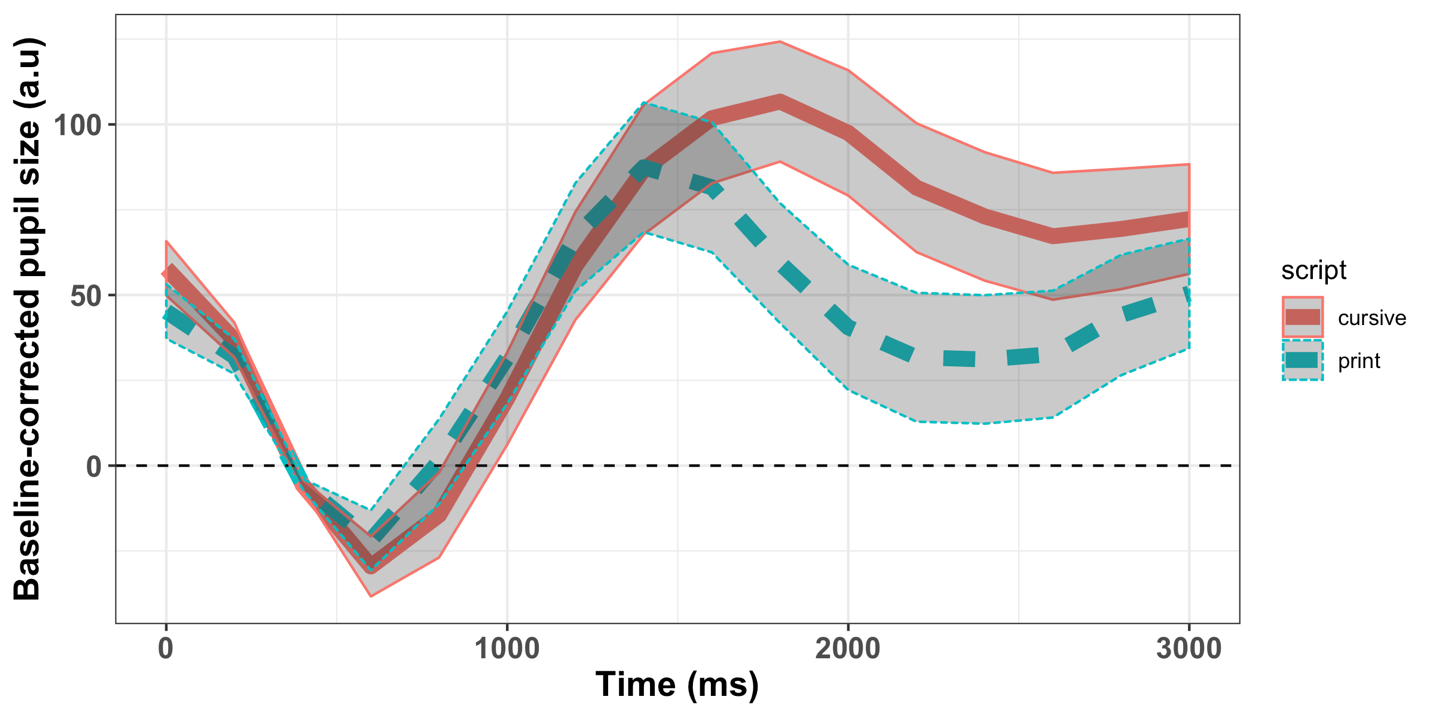


Figure 6. Pupillary time course as a function of script type. Ribbons denote 95% CIs.

In addition to pupillary time course, it is common to use summary measures: mean and max pupil size. Below you can see how to construct a graph based on mean and max pupil size using the *ggstatsplot* package (Patil, 2018).

data(cursive\_new)  
library(ggstatsplot)

mean\_pup<-subset(cursive\_new, timebins<=2500) %>%   
 group\_by(subject, script) %>%  
 summarise(meanpup=mean(aggbaseline), maxpup=max(aggbaseline)) %>%  
 ungroup()  
   
mean<-ggstatsplot::ggwithinstats(  
 data = mean\_pup,  
 x = script,  
 y = meanpup,  
 title = "Mean Pupil Size",  
 xlab = “Script”, # turn off the default subtitle  
 Ylab = ="Mean Change in Pupil Size (arbituary units)”,

)

plot(mean)

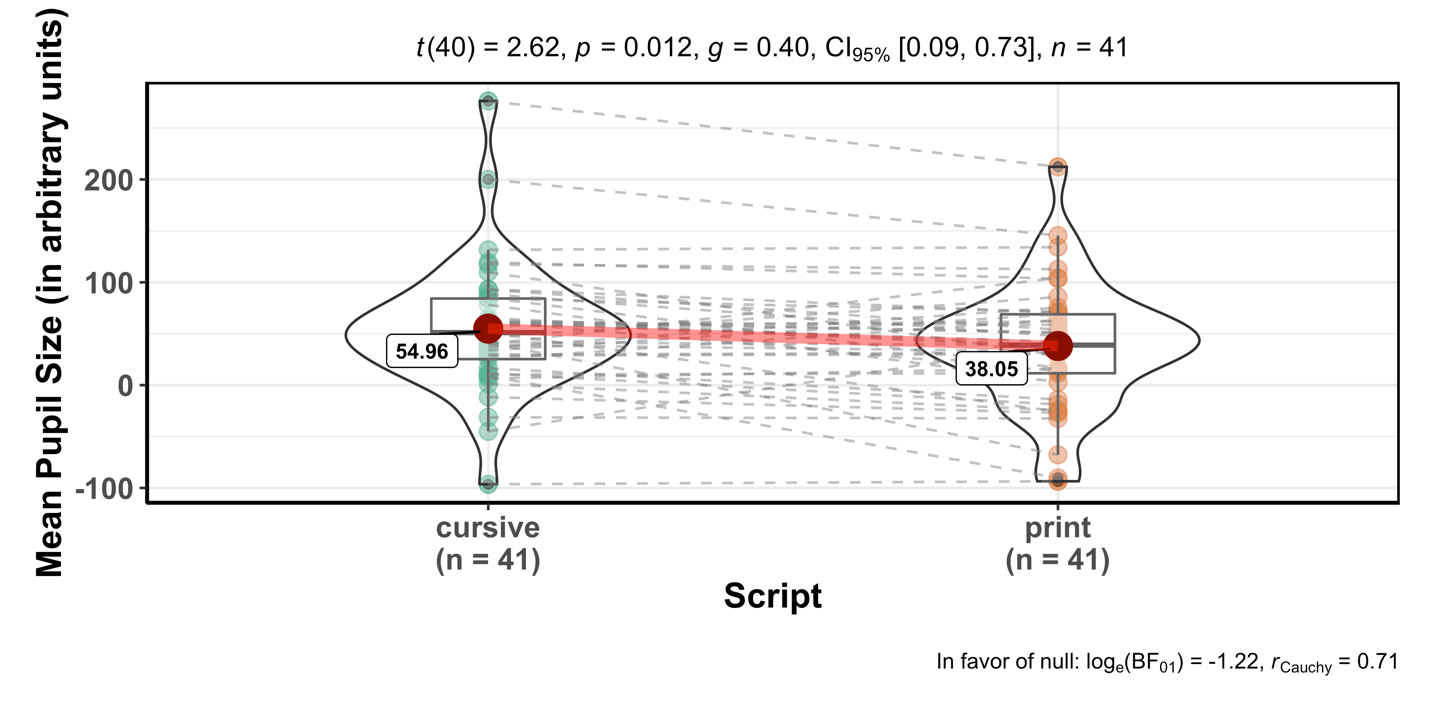


Figure 7. Mean Pupil Size.

#plot max pupil size

mean<-ggstatsplot::ggwithinstats(  
 data = mean\_pup,  
 x = script,  
 y = maxpup,  
 title = "Mean Pupil Size",  
 xlab = “Script”, # turn off the default subtitle  
 Ylab = ="Mean Change in Pupil Size (arbituary units)”,

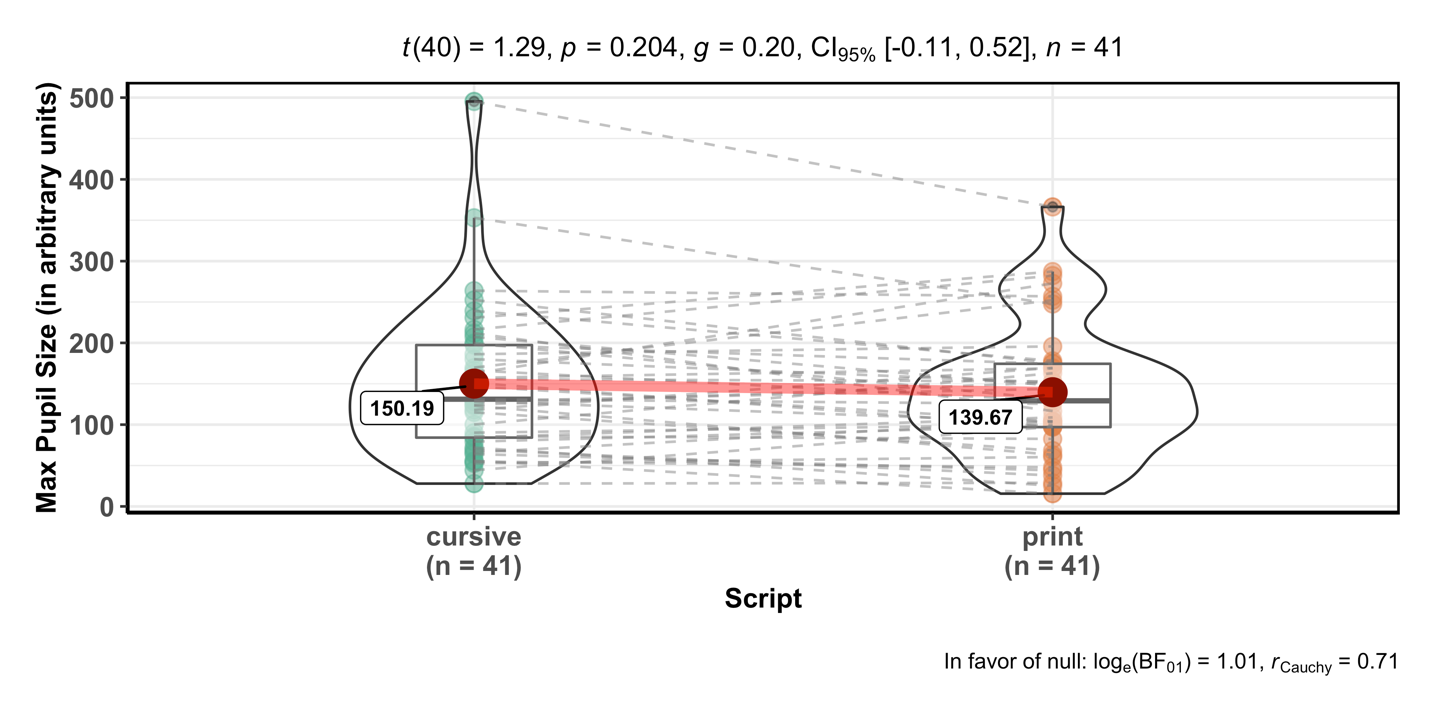
)  
****

Figure 8. Max Pupil Size

**Discussion**

While there are a number of viable solutions available to process eye-tracking data, they are typically unsuitable for research for several reasons:

* An all-graphical interface seldom provides information about the underlying data analysis
* File formats are sometimes proprietary and undocumented, lacking detailed annotation necessary for replicability
* Source code and description of the algorithms are not accessible to the user
* Some implementations are expensive or rely on expensive underlying 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 pre-process your data and make it suitable for analysis. This provides a generalized, replicable, and transparent method for preprocessing raw eye-tracking data.

## Limitations

There are several limitations of this package. The gazeR package is deliberately agnostic to type of statistical analysis. While the gazeR package does contain helper functions such as code\_poly to facilitate growth curve analysis (GCA) using orthogonal polynomials (Mirman, 2014), the pre-processed results could also be analyzed using other functional forms (e.g., reverse Gaussian and logistic; Seedorff, Oleson, and McMurray, 2018) and/or statistical techniques (e.g., general additive models and functional data analysis; Jackson & Sirois, 2009). In the absence of a field-standard statistical approach, we leave it up to the researcher to choose what statistical analysis to use.

Another limitation is that the gazeR pre-possessing pipeline is not exhaustive. We included a set of functions that we think will suffice for researchers to pre-process their gaze and pupil data, 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. 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 is limited to data that comes from the SR EyeLink. Much of the gazeR functionality is easily portable to data from other eye-trackers with the addition of functions for reading data and possibly renaming columns (variables) to match the EyeLink conventions.

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

# 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>

Cooper, R. M. (1974). The control of eye fixation by the meaning of spoken language: A new methodology for the real-time investigation of speech perception, memory, and language processing. *Cognitive Psychology*, *6*(1), 84–107. <https://doi.org/10.1016/0010-0285(74)90005-X>

Einhauser, W., Stout, J., Koch, C., & Carter, O. (2008). Pupil dilation reflects perceptual selection and predicts subsequent stability in perceptual rivalry. *Proceedings of the National Academy of Sciences*, *105*(5), 1704–1709. <https://doi.org/10.1073/pnas.0707727105>

Forbes, S.H. (2019). pupillometryR: An R package for preparing and analysing pupillometry data. Retrieved from https://github.com/samhforbes/PupillometryR

Gagl, B., Hawelka, S., & Hutzler, F. (2011). Systematic influence of gaze position on pupil size measurement: analysis and correction. *Behavior Research Methods*, *43*(4), 1171–1181. <https://doi.org/10.3758/s13428-011-0109-5>

Geller, J., Landrigan, J.-F., & Mirman, D. (2019). A Pupillometric Examination of Cognitive Control in Taxonomic and Thematic Semantic Memory. *Journal of Cognition*, *2*(1). <https://doi.org/10.5334/joc.56>

Geller, J., Still, M. L., Dark, V. J., & Carpenter, S. K. (2018). Would disfluency by any other name still be disfluent? Examining the disfluency effect with cursive handwriting. *Memory & Cognition*, *46*(7), 1109–1126. <https://doi.org/10.3758/s13421-018-0824-6>

Geller, J., Still, M. L., & Morris, A. L. (2016). Eyes wide open: Pupil size as a proxy for inhibition in the masked-priming paradigm. *Memory & Cognition*, *44*(4), 554–564. <https://doi.org/10.3758/s13421-015-0577-4>

Goldinger, S. D., He, Y., & Papesh, M. H. (2009). Deficits in cross-race face learning: Insights from eye movements and pupillometry. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *35*(5), 1105–1122. <https://doi.org/10.1037/a0016548>

Grange, J.A. (2015). trimr: An implementation of common response time trimming methods. R package version 1.0.1. <https://cran.r-project.org/web/packages/trimr/index.html>

Granholm, E., Asarnow, R. F., Sarkin, A. J., & Dykes, K. L. (1996). Pupillary responses index cognitive resource limitations. *Psychophysiology*, *33*(4), 457–461. Retrieved from <http://www.ncbi.nlm.nih.gov/pubmed/8753946>

Hershman, R., Henik, A., & Cohen, N. (2018). A novel blink detection method based on pupillometry noise. *Behavior Research Methods*, *50*(1), 107–114. https://doi.org/10.3758/s13428-017-1008-1

Karatekin, C., Couperus, J. W., & Marcus, D. J. (2004). Attention allocation in the dual-task paradigm as measured through behavioral and psychophysiological responses. *Psychophysiology*, *41*(2), 175–185. <https://doi.org/10.1111/j.1469-8986.2004.00147.x>

Kret, M. E., & Sjak-Shie, E. E. (2018). Preprocessing pupil size data: Guidelines and code. *Behavior Research Methods*, 1–7. <https://doi.org/10.3758/s13428-018-1075-y>

Mathôt, S. (2018). Pupillometry: Psychology, Physiology, and Function. *Journal of Cognition*, *1*(1). <https://doi.org/10.5334/joc.18>

Mathôt, S., Fabius, J., Van Heusden, E., & Van der Stigchel, S. (2018). Safe and sensible preprocessing and baseline correction of pupil-size data. *Behavior Research Methods*, *50*(1), 94–106. <https://doi.org/10.3758/s13428-017-1007-2>

Murphy, P. R., O'connell, R. G., O'sullivan, M., Robertson, I. H., & Balsters, J. H. (2014). Pupil diameter covaries with BOLD activity in human locus coeruleus. *Human Brain Mapping*, *35*(8), 4140-4154.

Nyström, M., Hooge, I., & Andersson, R. (2016). Pupil size influences the eye-tracker signal during saccades. *Vision Research*, *121*, 95–103. <https://doi.org/10.1016/J.VISRES.2016.01.009>

Patil, I. (2018). ggstatsplot:“ggplot2” Based Plots with Statistical Details. CRAN.

Piquado, T., Isaacowitz, D., & Wingfield, A. (2010). Pupillometry as a measure of cognitive effort in younger and older adults. *Psychophysiology*, *47*(3), 560–569. <https://doi.org/10.1111/j.1469-8986.2009.00947.x>

Reilly, J., Kelly, A., Kim, S. H., Jett, S., & Zuckerman, B. (2018). The human task-evoked pupillary response function is linear: Implications for baseline response scaling in pupillometry. *Behavior Research Methods*. <https://doi.org/10.3758/s13428-018-1134-4>

Salverda, A. P., & Tanenhaus, M. K. (2018). The visual world paradigm. In Annette M. B. de Groot and Peter Hagoort (Eds) *Research methods in psycholinguistics and the neurobiology of language: A practical guide*, pp. 89-110. Wiley Blackwell.

Satterthwaite, T. D., Green, L., Myerson, J., Parker, J., Ramaratnam, M., & Buckner, R. L. (2007). Dissociable but inter-related systems of cognitive control and reward during decision making: Evidence from pupillometry and event-related fMRI. *NeuroImage*, *37*(3), 1017–1031. https://doi.org/10.1016/j.neuroimage.2007.04.066

Seedorff, M., Oleson, J., & McMurray, B. (2018). Detecting when timeseries differ: Using the Bootstrapped Differences of Timeseries (BDOTS) to analyze Visual World Paradigm data (and more). *Journal of Memory and Language*, *102*, 55–67. <https://doi.org/10.1016/J.JML.2018.05.004>

Siegle, G. J., Steinhauer, S. R., Carter, C. S., Ramel, W., & Thase, M. E. (2003). Do the Seconds Turn Into Hours? Relationships between Sustained Pupil Dilation in Response to Emotional Information and Self-Reported Rumination. *Cognitive Therapy and Research*, *27*(3), 365–382. <https://doi.org/10.1023/A:1023974602357>

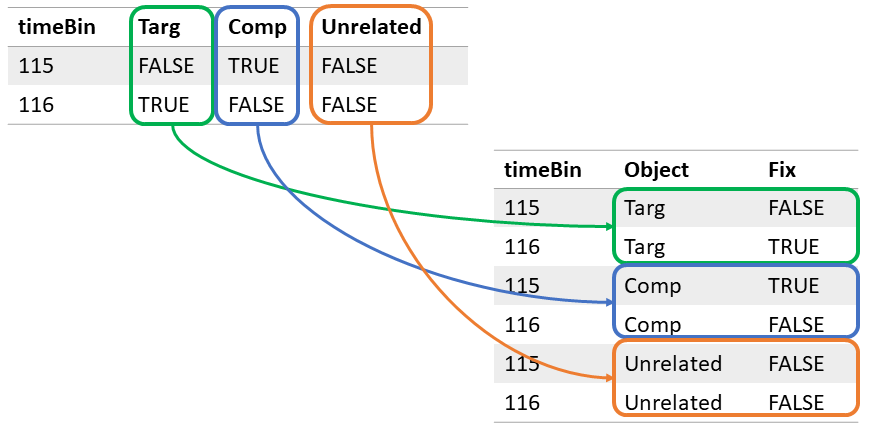
Tanenhaus, M. K., Spivey-Knowlton, M. J., Eberhard, K. M., & Sedivy, J. C. (1995). Integration of visual and linguistic information in spoken language comprehension. *Science (New York, N.Y.)*, *268*(5217), 1632–1634. Retrieved from <http://www.ncbi.nlm.nih.gov/pubmed/7777863>

Tsukahara, J.S. (2018). pupillometry: An R Package to Preprocess Pupil Data. Retrieved from <https://dr-jt.github.io/pupillometry>

Van Gerven, P. W. M., Paas, F., Van Merriënboer, J. J. G., & Schmidt, H. G. (2004). Memory load and the cognitive pupillary response in aging. *Psychophysiology*, *41*(2), 167–174. <https://doi.org/10.1111/j.1469-8986.2003.00148.x>

Winn, M. B., Wendt, D., Koelewijn, T., & Kuchinsky, S. E. (2018). Best Practices and Advice for Using Pupillometry to Measure Listening Effort: An Introduction for Those Who Want to Get Started. *Trends in Hearing*, *22*, 2331216518800869. <https://doi.org/10.1177/2331216518800869>

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



1. The first line of code defines the path to the fixation report file included with the package. Because package installations differ across platforms and users, this line is necessary to define the user-specific path to the included data file. More generally, when a user wants to analyze their own data set, the gaze\_path variable will need to be the path to that data file. [↑](#footnote-ref-1)
2. 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. Additonally, 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. [↑](#footnote-ref-2)