Oh, the Places You'll Go! A Step-by-Step Guide to Analyzing Webcam Eye-Tracking Data for L2 Research

Jason Geller[1], Yanina Prystauka[2], Sarah Colby[3], and Julia Droulin[4]

[1]Department of Psychology and Neuroscience, Boston College

[2]Department of Psychology and Neuroscience, University of Bergen

[3]Department of Psychology and Neuroscience, University of Ottowa

[4]Department of Psychology and Neuroscienc, University of North Carolina at Chapel Hill

Author Note

Jason Geller http://orcid.org/0000-0002-7459-4505

Yanina Prystauka http://orcid.org/0000-0000-0000-0002

Sarah Colby http://orcid.org/0000-0000-0000-0003

Julia Droulin http://orcid.org/0000-0000-0000-0003

This study was not preregistered. The authors have no conflicts of interest to disclose.

Author roles were classified using the Contributor Role Taxonomy (CRediT; https://credit.niso.org/) as follows: *Jason Geller***:** conceptualization, writing – original draft, data curation, writing – review & editing, software, and formal analysis. *Yanina Prystauka***:** writing – original draft, writing – review & editing, and formal analysis. *Sarah Colby***:** writing – original draft and writing – review & editing. *Julia Droulin***:** conceptualziation, writing – original draft, writing – review & editing, and funding acquisition

Correspondence concerning this article should be addressed to Jason Geller, Department of Psychology and Neuroscience, Boston College, Mcguinn Hall 405, Chestnut Hill, MA 02467-9991, USA, drjasongeller@gmail.com: jason.geller@bc.edu

Abstract

Eye-tracking has become a valuable tool for studying cognitive processes in second language (L2) acquisition and bilingualism (Godfroid et al., 2024). While research-grade infrared eye-trackers are commonly used, there are a number of isuses that limit its wide-spread adoption. Recently, consumer-based eye-tracking has emerged as a more affordable alternative, requiring only internet access and a personal webcam. However, webcam eye-tracking presents unique design and preprocessing challenges that must be addressed for valid results. To help researchers overcome these challenges,we developed a comprehensive tutorial focused on using webcam eye-tracking for L2 langauge research (but the information provided can be extended to any online research using the visual world paradigm (VWP)). Our guide will cover all key steps, from experiment design to data preprocessing and analysis, where we highlight the R package `webgazeR`, which is open source and freely available for download and installation: https://github.com/jgeller112/webgazeR. We offer best practices for environmental conditions, participant instructions, and tips for designing VWP experiments with webcam eye-tracking. TO demonstrate these steps, we analyze data collected through the Gorilla platform (Anwyl-Irvine et al., 2020) using a single word Spanish VWP and show competeiton within and between L2/L1. This tutorial aims to empower researchers by providing a step-by-step guide to successfully conduct webcam-based eye-tracking studies. To follow along with this tutorial, please download the entire manuscript and its accompaying code with data from here:

Keywords: VWP, Tutorial, Webcam eye-tracking, R, Gorilla

Oh, the Places You'll Go! A Step-by-Step Guide to Analyzing Webcam Eye-Tracking Data for L2 Research

Eye-tracking technology, which has a history spanning over a century, has seen remarkable advancements. In the early days, eye-tracking required the use of contact lenses fitted with search coils, often requiring anesthesia (Płużyczka, 2018), or the attachment of suction cups to the sclera of the eyes. These methods were not only cumbersome for the researcher, but also uncomfortable and invasive for participants. Over time, such approaches have been replaced by non-invasive, lightweight, and user-friendly systems. Today, modern eye-tracking technology is widely accessible in laboratories worldwide, enabling researchers to tackle critical questions about cognitive processes . This evolution has had a profound impact on fields such as psycholinguistics and bilingualism opening up new possibilities for understanding how language is processed in real time (Godfroid (she/her) et al., n.d.).

Despite its widespread usage, eye-tracking technology faces several obstacles that can limit its accessibility. One significant challenge is the specialized expertise required to operate research-grade eye-trackers. Proper usage often demands many hours of training, meaning most research must be conducted in a lab by a trained student or faculty member. Another major limitation is the cost. Eye-trackers can be prohibitively expensive, ranging from a few thousand dollars (e.g., Gazepoint; www.gazept.com) to tens of thousands of dollars (e.g., Tobii (www.tobii.com; SR Research (www.sr-research.com). As a result, not everyone possess the resources, or the time, to incorporate eye-tracking into their research program.

In addition, eye-tracking research often requires participants to visit a laboratory, which significantly limits the diversity of the sample or population researchers can recruit. Behavioral science research, in general, frequently suffers from a lack of diversity, relying heavily on participants who are predominantly Western, Educated, Industrialized, Rich, Democratic, and able-bodied (WEIRDA). This focus often excludes individuals from geographically dispersed areas, those from lower socioeconomic backgrounds, and people with disabilities who may face barriers to accessing research facilities. In language research, this issue is particularly evident, as it often prioritizes English-speaking, monolingual, populations (Blasi et al., 2022; Bylund et al.,

2024) and largely includes individuals with typical language production and comprehension abilities. These limitations not only narrow the populations available for study but also compromise the generalizability and applicability of research findings.

Eye-tracking outside the lab

Methods that allow participants to use their own equipment from anywhere in the world offer a potential solution to the issues outlined above, enabling researchers to recruit more diverse and disadvantaged samples and explore a broader range of questions (Gosling et al., 2010). The shift toward online behavioral experiments has been gradually increasing in the behavioral sciences and has become every more important since the 2020 pandemic, which forced many of us to run studies online (Anderson et al., 2019; Rodd, 2024). The *onlineification* of behavioral research has prompted the development of eye-tracking methods that do not rely on traditional lab settings. One method, manual eye-tracking (Trueswell, 2008), involves using video recordings of participants, which can be collected through online teleconferencing platforms such as Zoom (www.zoom.com). Post-hoc, eye gaze (direction) is manually analyzed frame by frame from these recordings.

Another method, which is the focus of this tutorial, is automated eye-tracking or webcam eye-tracking. Webcam eye-tracking requires three things: 1. A personal computer. 2. An internet connection and 3. A purchased or pre-installed webcamera. Gaze information can be collected via a web broswer. One common method to perform webcam eye-tracking is through an open source and free to use JavaScript library plugin called WebGazer (Papoutsaki et al., 2016). This plugin is already incorporated into several popular experimental platforms (e.g., *Gorilla*, *jsPsych*, *PsychoPy*, and *PCIbex*; [Anwyl-Irvine et al. (2020) Peirce et al. (2019); Leeuw (2015); Zehr & Schwarz, 2018) making it extremely easy to start webcam eye-tracking. WebGazer.js utilizes a webcam connected to the internet to track eye movements in real time. It employs a facial feature detection algorithm that estimates the position of the pupils in the webcam stream (relative to the

face). By analyzing the relative movement of the eyes, WebGazer.js employs machine learning to estimate the user's gaze location on the screen.To achieve this, WebGazer uses a dynamic calibration process wherein users look and click on random dot locations on the screen, or follow a dot as it moves to different locations on the screen. This calibration enhances the mapping between eye gaze and on-screen coordinates, ensuring more precise tracking during subsequent tasks.

It is important to note that WebGazer is not the only method available. Other methods have been implemented by companies like Tobii (www.tobii.com) and Labvanced (Kaduk et al., 2024) . However, because these methods are proprietary, it is unclear what they are doing under the hood.

The algorithms underlying webcam-based eye tracking differ significantly from those used in research-grade eye trackers. Research-grade systems employ video-based recording and rely on the pupil-corneal reflection (P-CR) method to track gaze with high precision (Carter & Luke, 2020). This method utilizes infrared light to illuminate the eyes, capturing reflections (known as glints) from the cornea and pupil. High-speed cameras simultaneously capture images at rates of hundreds or thousands of frames per second to measure eye position. By combining data from the corneal reflections and pupil location, these systems calculate gaze direction and position. Proprietary algorithms then map this information to specific locations on the screen.

This leads to an important question: how does consumer-grade webcam eye tracking compare to research-grade systems? While validation studies are ongoing, webcam-based eye trackers generally exhibit reduced spatiotemporal accuracy. Studies have reported that these systems achieve spatial accuracy and precision exceeding 1° of visual angle, with latencies ranging from 200 ms to 1000 ms (Kaduk et al., 2024; Semmelmann & Weigelt, 2018; Slim et al., 2024; Slim & Hartsuiker, 2023). Furthermore, the sampling rate of webcam-based systems is much lower, typically capped at 60 Hz, with most studies reporting average or median rates

around 30 Hz (Bramlett & Wiener, 2024; Prystauka et al., 2024). Unlike research-grade systems, webcam eye trackers do not use infrared light; instead, they rely on ambient light from the participant's environment. This dependency introduces additional variability in tracking performance.

To compare, research-grade systems like the Tobii Pro Spectrum provides spatial precision of 0.03°–0.06° RMS, spatial accuracy of <0.3°, and latency of less than 2.5 ms, with a sampling rate of up to 1200 Hz (AB, 2024; Nyström et al., 2021). These advanced metrics make research-grade systems ideal for studies requiring high temporal and spatial resolution.

Bringing the visual world paradigm online

Despite the differences between research-grade and consumer grade eye-tracking, a number of studies have begun to look at if lab-based results replicate online using webcam eye-tracking. Most relevant to this tutorial are online replications using the VWP (Tanenhaus et al., 1995; cf. Cooper, 1974). For the past 25 years, the VWP has been a dominant force in language research, helping researchers tackle a wide range of topics, including sentence processing (Eberhard et al., 1995), word recognition (Allopenna et al., 1998), bilingualism, and the effects of brain damage on language (Mirman & Graziano, 2012).

The VWP is a simple task. In a typical VWP experiment, several objects are shown in a display and participants are instructed to point or click on one of the objects. As participants listen to the spoken phrase or word that specifies the target object, their eye movements are recorded. Looks to a given object map very closely and with high temporal precision onto the mental activation of the word or concept corresponding to that object, providing unique insights into the time course of cognitive processing.

Most of the research with the VWP has been conducted in labs with research grade eye-trackers. However, there have been several attempts to conduct these experiments online with webcam eye-tracking. In one of the first studies to examine the VWP in an online setting using

webcam eye-tracking, Degen et al. (2021) reported a replication of Sun and Breheny (2020) looking at set size and determiners in language processing. In their study, they used a VWP with a carrier phrase that manipulated gender (male vs. female), determiners (i.e., all, some, numeral) and noun. Degen et al. (2021) largely replicated the basic pattern of results from Sun and Breheny (2020) observering an interaction between set size and determiners. Most notable, however, is the considerable time delay. Using webcam eye-tracking significant effects did not arise until 700 ms later. They attribted this to reduced spatial precision and lower sampling rates, AOI size, and number of calibrations performed. Several other studies using the VWP online have noted a similar temporal delay in effects (Slim et al., 2024; Slim & Hartsuiker, 2023).

While the temporal delay reported in these replications presents a significant limitation to using webcam eye-tracking with the VWP, recent developments with WebGazer have improved upon this shortcoming. Vos et al. (2022) demonstrated a significant reduction in delay—approximately 50 ms—when comparing the lab-based and online versions of the VWP using an updated version of WebGazer within jsPsych (Leeuw, 2015). More recently, studies by Prystauka et al. (2024) and Bramlett and Wiener (2024), leveraging Gorilla and the updated WebGazer algorithm, reported comparable effects between the online and lab-based version of the VWP. These findings highlight that the online version of the VWP, powered by webcam eye-tracking, can achieve results similar to those of traditional lab-based studies.

Tutorial

Taken together, it seems that webcam eye-tracking is viable alternative to lab-based eye-tracking.Given this, we aimed to support researchers in their efforts to conduct high-quality webcam eye-tracking studies with the VWP. While a valuable tutorial on webcam eye-tracking in the VWP already exists (Bramlett & Wiener, 2024), we believe there is value in having multiple resources available to researchers. To this end, we sought to expand on the tutorial by Bramlett

and Wiener (2024) by incorporating many of their useful recommendations, but also offering an R package to help streamline data pre-processing.

The purpose of this tutorial is to provide an overview of the basic set-up and design features of an online VWP task using the Gorilla platform (Anwyl-Irvine et al., 2020) and to highlight the pre-processing steps needed to analyze webcam eye-tracking data. Here we use the popular open source programming language R and introduce the `webgazeR` package (Geller & Prystauka, 2024) to facilitate pre-processing of webcam data. To highlight the steps needed to process webcam eye-tracking data we present data from a Spanish spoken word VWP with L2 Spanish speakers. To our knowledge, L2 processing and competitor effects have not been looked at in the online version of the VWP.

The structure of the tutorial will be as follows. We first outline the general methods used to conduct a visual world webcam eye-tracking experiment. Next, we detail the data preprocessing steps required to prepare the data for analysis. Finally, we demonstrate one statistical approach for analyzing our preprocessed data, highlighting its application and implications.

L2 VWP Webcam Eye-tracking

To highlight the preprocessing steps required to analyze webcam eye-tracking data, we examined the competitive dynamics of second-language (L2) learners of Spanish during spoken word recognition. Specifically, we investigated both within-language and cross-language (L2/L1) competition using webcam-based eye-tracking.

It is well established that competition plays a critical role in language processing (Magnuson et al., 2007). In speech perception, as the auditory signal unfolds over time, competitors (or cohorts)—phonological neighbors that differ from the target by an initial phoneme—become activated. To successfully recognize the spoken word, these competitors must be inhibited or suppressed. For example, as the word *wizard* is spoken, cohorts like *whistle*

might also be briefly activated and in order for wizard to be recognized, *whistle* must be suppressed.

A key question in the L2 literature is whether competition can occur cross-linguistically, with interactions between a speaker's first language (L1) and second language (L2). A recent study by Sarrett et al. (2022) explored this question using carefully designed stimuli to examine within- and between linguistic (L2/L1) competition in adult L2 Spanish speakers learners using a Spanish VWP. Their study included two key conditions:

1. Spanish-Spanish condition: A Spanish competitor was presented alongside the target word. For example, if the target word spoken was "cielo" (sky), the Spanish competitor was "ciencia" (science).
2. Spanish-English (cross-linguistic) condition: An English competitor was presented for the Spanish target word. For example, if the target word spoken was "botas" (boots), the English competitor was "border."

Sarrett et al. (2022) also included a no competition condition where the Spanish-English pairs were not cross-linguistic competitors (e.g., *frontera* as the target word and *botas*/*boots* as an unrelated item in the pair). They observed competition effects in both of the critical conditions: within-Spanish competition (e.g., *cielo - ciencia*) and cross-linguistic competition (e.g., *botas - border*). There was no competetion effects in the no competion condition. For this tutorial, we collected data to conceptually replicate their pattern of findings.

There are two key differences between our dataset and the original study by Sarrett et al. (2022) worth noting. First, Sarrett et al. (2022) focused on adult L2 Spanish speakers and posed more fine-grained questions about the time course of competition and resolution and its relationship with L2 language acquisition. Second, unlike McCall et al., who measured Spanish proficiency objectively (e.g., using LexTALE-esp; Izura et al. (2014)), we relied on Prolific's filters to recruit L2 Spanish speakers.

Our primary goal here was to demonstrate the pre-processing steps required to analyze webcam-based eye-tracking data. A secondary goal was to provide evidence of L2 competition within and between or cross-linguistically using this methodology. To our knowledge, no papers have looked at spoken word recognition and competition using online methods. It is our hope that researchers can use this to test more detailed questions about L2 processing using webcam-based eye-tracking.

Method

All tasks herein can be previewed here (https://app.gorilla.sc/openmaterials/953693). The manuscript, data, and R code can be found on Github (https://github.com/jgeller112/webcam_gazeR_VWP).

Participants

A total of 187 participants consented to participate in the study using the Gorilla hosting and experiment platform. Of these, 111 passed the headphone screener checkpoint and proceeded to the task. Among these, 32 participants successfully completed the Visual World Paradigm (VWP) task with at least 100 trials, while 79 participants failed calibration. Ninety-one participants completed the entire experiment, including the final questionnaires. Table 1 provides basic demographic information about the participants who completed the full experiment. After applying additional exclusion criteria (low accuracy (< 80%) and excessive missing eye-data (> 30%) , the final sample consisted of 28 participants with usable eye-tracking data.

Table 1

Demographic variables

Characteristic	N = 91[1]
Age	(20.0, 35.0), 28.2(4.4)
Gender	
Female	42 / 91 (46%)
Male	49 / 91 (54%)
Spoken dialect	
Do not know	11 / 91 (12%)
Midwestern	19 / 91 (21%)
New England	11 / 91 (12%)
Other (please specify)	7 / 91 (7.7%)
Pacific northwest	7 / 91 (7.7%)
Pacific southwest	7 / 91 (7.7%)
Southern	21 / 91 (23%)
Southwestern	8 / 91 (8.8%)
Race	
Decline to state	1 / 91 (1.1%)
Hispanic or Latino	38 / 91 (42%)
Not Hispanic or Latino	52 / 91 (57%)
Browser	
Chrome	77 / 91 (85%)
Edge	3 / 91 (3.3%)
Firefox	7 / 91 (7.7%)
Safari	4 / 91 (4.4%)
Years Speaking Spanish	(0, 35), 15(10)
Percentage Time Speaking Spanish	25(23)

[1] (Min, Max), Mean(SD); n / N (%); Mean(SD)

Materials

VWP. ***Items.*** We adapted materials from Sarrett et al. (2022). In their cross-linguistic VWP, participants were presented with four pictures and a spoken Spanish word and had to select the image that matched the spoken word by clicking on it. The word stimuli for the experiment were chosen from textbooks used by students in their first and second year college Spanish courses.

The item sets consisted of two types of phonologically-related word pairs: one pair of Spanish-Spanish words and another of Spanish-English words. The Spanish-Spanish pairs were unrelated to the Spanish-English pairs. All the word pairs were carefully controlled on a number of dimensions (see (Sarrett et al., 2022)).

There were three experimental conditions: (1) the Spanish-Spanish condition, where one of the Spanish words was the target and the other was the competitor; (2) the Spanish-English condition, where a Spanish word was the target and its English phonological cohort served as the competitor; and (3) the No Competitor condition, where the Spanish word did not overlap with any other word in the set. The Spanish-Spanish condition had twice as many trials as the other conditions due to the interchangeable nature of the target and competitor words in that pair.

There were 15 sets of 4 items (half the number of sets used in (Sarrett et al., 2022)). Each item within a set was repeated 4 times as the target word. This yielded 240 trials (15 sets × 4 items per set × 4 repetitions). Each item set consisted of one Spanish-Spanish cohort pair and one Spanish-English cohort pair. Both items in a Spanish-Spanish pair had a“reciprocal” competitor relationship (that is, we could test activation for *cielo* given *ciencia*, and for *ciencia* given *cielo*). Consequently, there were 120 trials in the Spanish-Spanish condition. In contrast, only one item from the Spanish-English pair had the specified competitor relationship (we could test activation for frontera *border*, given *botas*, but when hearing *frontera*, there was no competitor). Thus, there were only 60 trials for each the Spanish-English competition as well as the No Competitor condition. Items occurred in each of the four corners of the screen on an equal numbers of trials.

Stimuli. In Sarrett et al. (2022) all auditory stimuli were recorded by a female bilingual speaker whose native language was Mexican Spanish and also spoke English. Stimuli were recorded in a sound-attenuated room sampled at 44.1 kHz. Auditory tokens were edited to reduce noise and remove clicks. The auditory tokens were then amplitude normalized to 70 dB SPL. For each target word, there were four separate recordings so each instance was unique.

Visual stimuli were images from a commercial clipart database that were selected by a consensus method involving a small group of students. All .wav files were converted to .mp3 for online data collection. All stimuli can be found here: https://osf.io/mgkd2/.

Headphone screener. Headphones were required for all participants. To make sure participants where wearing headphones, we used a six-trial task taken from Woods et al. (2017). On each trial, three tones of the same frequency and duration were presented sequentially. One tone had a lower amplitude than the other two tones. Tones were presented in stereo, but the tones in the left and right channels were 180 out of phase across stereo channels—in free field, these sounds should cancel out or create distortion, whereas they will be perfectly clear over headphones. The listener picked which of the three tones was the quietest. Performance is generally at the ceiling when wearing headphones but poor when listening in the free field (due to phase cancellation).

Demographics questionnaire. Participants completed a demographic questionnaire as part of the study. The questions covered basic demographic information, including age, gender, spoken dialect, ethnicity, and race.

Participants also answered a series of questions related to their personal health and environmental conditions during the experiment. These questions addressed any history of vision problems (e.g., corrected vision, eye disease, or drooping eyelids) and whether they were currently taking medications that might impair judgment. Participants also indicated if they were wearing eyeglasses, contacts, makeup, false eyelashes, or hats.

The questionnaire inquired about their environment, asking if there was natural light in the room, if they were using a built-in camera or an external one (with an option to specify the brand), and their estimated distance from the camera. Participants were asked to estimate how many times they looked at their phone or got up during the experiment and whether their environment was distraction-free.

Additional questions assessed the clarity of calibration instructions, allowing participants to suggest improvements, and asked if they were wearing a mask during the session. These questions aimed to gather insights into personal and environmental factors that could impact data quality and participant comfort during the experiment.

To gauge L2 experience, we asked participants when they started speaking Spanish, how many years of Spanish speaking experience they had, and to provide, on a scale between 0-100, how often they use Spanish in their daily lives.

Procedure

All tasks were completed in a single session, lasting approximately 45 minutes. The tasks were presented in a fixed order: consent, headphone screener, spoken word VWP, and questionnaire items.

The experiment was programmed in the Gorilla Experiment Platform (Anwyl-Irvine et al., 2019), with personal computers as the only permitted device type. Upon entering the online study, participants received general information to decide if they wished to participate, after which they provided informed consent. Participants were then instructed to adjust the volume to a comfortable level while noise played.

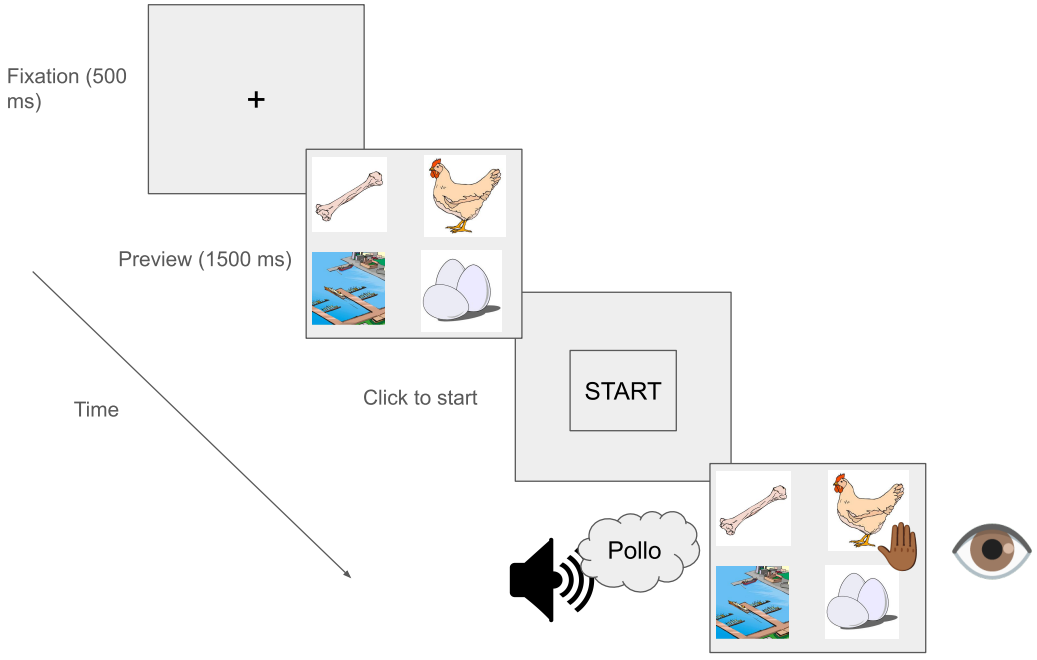
Next, participants completed a headphone screening test. They had three attempts to pass this test. If unsuccessful by the third attempt, participants were directed to an early exit screen, followed by the questionnaire.

For those who passed the screening, the next task was the VWP. This began with instructional videos providing specific guidance on the ideal experiment setup for eye-tracking and calibration procedures. Participants were then required to enter full-screen mode before calibration. Calibration occurred every 60 trials for a total of 3 calibrations. Participants had three attempts to successfully complete each calibration phase. If calibration was unsuccessful, participants were directed to an early exit screen, followed by the questionnaire.

In the main VWP task, each trial began with a 500 ms fixation cross at the center of the screen. This was followed by a preview screen displaying four images, each positioned in a corner of the screen. After 1500 ms, a start button appeared in the center. Participants clicked the button to confirm they were focused on the center before the audio played. Once clicked, the audio was played, and the images remained visible. Participants were instructed to click the image that best matched the spoken target word, while their eye movements were recorded. Eye movements were only recorded on that screen. Figure 1 displays the VWP trial sequence.

Figure 1

VWP trial schematic



After completing the main VWP task, participants proceeded to the final questionnaire, which included questions about the eye-tracking task and basic demographic information. Participants were then thanked for their participation.

Cleaning data

After the data is collected you can begin preprocessing your data. Below we highlight the steps needed to preprocess your webcam eye-tracking data and get it ready for analysis. For some of this preprocessing we will use the newly created `webgazeR`package (v. 0.1.0) which is an extension of the `gazeR` package (Geller et al., 2020) which was created to analyze VWP data in lab-based studies.

For preprocessing webcam eye data, we follow five general steps:

1. Reading in data
2. Combining trial- and eye-level data
3. Assigning areas of interest
4. Time Binning

5. Aggregating (optional)

For each of these steps, we will display R code chunks demonstrating how to perform each step with helper functions (if applicable) from the `webgazeR` package in R.

Reading in Data

Package Installation and Setup. Before turning to the pre-processing code below, we will need to make sure all the necessary packages are installed. The code will not run if the packages are not installed properly. If you have already installed the packages mentioned below, then you can skip ahead and ignore this section. To install the necessary packages, simply run the following code - it may take some time (between 1 and 5 minutes to install all of the libraries so you do not need to worry if it takes some time).

webgazeR installation. The `webgazeR` package can be installed along with helper packages using the remotes package:

```
remotes::install_github("jgeller112/webgazeR")
```

Once this is installed, `webgazeR` can be loaded along with additional useful packages:

```
options(stringsAsFactors = F) # no automatic data transformation
options("scipen" = 100, "digits" = 10) # suppress math annotation
library(tidyverse) # data wrangling
library(remotes) # install github repo
library(here) # relative paths instead of abosoulte aids in reproduce
library(tinytable) # nice tables
library(janitor)# functions for cleaning up your column names
library(webgazeR) # has webcam functions
library(readxl) # read in excel files
library(ggokabeito)
```

```
library(permuco) # permutation analysis

library(foreach) # permiutation analysis

library(geomtextpath) # for plotting labels on lines of ggplot figs

library(cowplot) # combine ggplot figs
```

Once `webgazeR` and other helper packages have been installed and loaded the user is ready to start cleaning your data.

Gorilla Data

Behavioral, trial-level, data. To process eye-tracking data you will need to make sure you have both the behavioral data and the eye-tracking data files. We have all the data needed in the repository by navigating to the L2 subfolder in the data folder (data -> L2). For the behavioral data, Gorilla produces a `.csv` file that includes trial-level information (here contained in the object `L2_data`). The files needed are called `data_exp_196386-v5_task-scf6.csv`. and `data_exp_196386-v6_task-scf6.csv`. We have two files becasue we ran a modified version of the experiment.

The .csv files contain meta-data for each each trial, such as what picture were presented on each trial, which object was the target, reaction times, audio presentation times, what object was clicked on, etc. To load our data files into our R environment, we use the `here` package to set a relative rather than an absolute path to our files. The below object `L2_data` merges both `data_exp_196386-v5_task-scf6.csv` and `data_exp_196386-v6_task-scf6.csv` into one object.

```
# load in trial level data

# combine data from version 5 and 6 of the task

L2_1 <- read_csv(here("data", "L2", "data_exp_196386-v5_task-scf6.csv"))
```

```
L2_2 <- read_csv(here("data", "L2", "data_exp_196386-v6_task-scf6.csv"))

L2_data <- rbind(L2_1, L2_2) # bind the two objects together
```

Eye-tracking data. Gorilla currently saves each participant's eye-tracking data trial by trial. The `raw` subfolder in the data folder in the project repository contains the eye-tracking files by participant for each trial individually. Contained in those files, we have information pertaining to each trial such as participant id,time since trial started, x and y coordinates of looks, convergence (the model's confidence in finding a face (and accurately predicting eye movements), face confidence (represents the support vector machine (SVM) classifier score for the face model fit), and information pertaining to the the AOI screen coordinates (standardized and user-specific). The `vwp_files_L2` object below contains a list of all the files contained in the folder. Because `vwp_files_L2` contains trial data as well as calibration data, we remove the calibration trials and save the files to `vwp_paths_filtered_L2`.


```
# Get the list of all files in the folder

vwp_files_L2  <- list.files(here::here("data", "L2", "raw"), pattern = "\

\.xlsx$", full.names = TRUE)

# Exclude files that contain "calibration" in their filename

vwp_paths_filtered_L2 <- vwp_files_L2[!grepl("calibration", vwp_files_L2)]
```

When data is generated from Gorilla, each trial in your experiment is saved as an individual file. Because of this, we need some way to take all the individual files and merge them together. The `merge_webcam_files()`function merges each trial from each participant into a single tibble or data frame. Before running the `merge_webcam_files()` function, ensure that your working directory is set to where the files are stored. `merge_webcam_files()` reads in all the .xlsx files from the raw subfolder, binds them together into one dataframe, and cleans up the column names. The function then filters the data to include only rows where the type is

"prediction" and the `screen_index` matches the specified value (in our case, screen 4 is where we collected eye-tracking data). If you recorded across multiple screens the `screen_index` argument can take multiple values (e.g., `screen_index`= c(1, 4, 5), will take eye-tacking information from screens, 1, 4, and 5)). `merge_webcam_files()` also renames the `spreadsheet_row` column to trial and sets both `trial` and `subject` as factors for further analysis in our pipeline. As a note, all steps should be followed in order due to the renaming of column names. If you encounter an error it might be because column names have not been changed.

```
setwd(here::here("data", "L2", "raw")) # set working directory to raw data
folder
edat_L2 <- merge_webcam_files(vwp_paths_filtered_L2, screen_index=4) # eye
tracking occured ons creen index 4
```


Subject and trial level data removal. To ensure high-quality data, it is essential to filter out unreliable data based on both behavioral and eye-tracking criteria before merging datasets. In our dataset, participants will be excluded if they meet any of the following conditions: failure to successfully calibrate throughout the experiment (less than 100 trials), low accuracy (< 80%) , low sampling rates (< 5), and a high proportion of gaze data outside the screen coordinates (> 30%). Successful calibration is crucial for capturing accurate eye-tracking measurements, so participants who could not maintain proper calibration may have inaccurate gaze data. Similarly, low accuracy may indicate poor engagement or task difficulty, which can reduce the reliability of the behavioral data and suggest that eye-tracking data may be less precise.

First, we will create a cleaned up version of our use the behavioral, trial-level, data `L2_data` by creating an object named `eye_behav_L2` that selects useful columns from that file and renames stimuli to make them more intuitive. Because most of this will be user-specific, no function is called here. Below we describe the preprocessing done on the behavioral data file. The below code processes and transforms the `L2_data` dataset into a cleaned and structured

format for further analysis. First, the code renames several columns for easier access using `janitor::clean_names()` (Firke, 2023) function. We then select only the columns we need and filter the dataset to include only rows where `zone_type` is "response_button_image", representing the picture selected for that trial. Afterward, the function renames additional columns (`tlpic` to `TL`, `trpic` to `TR`, etc.). We also renamed `participant_private_id` to `subject`, `spreadsheet_row` to `trial`, and `reaction_time` to `RT`. This makes our columns consistent with the `edat` above for merging later on. Lastly, `reaction time` (RT) is converted to a numeric format for further numerical analysis.

It is important to note here that what the behavioral spreadsheet denotes as trial is not in fact the trial number used in the eye-tracking files. Thus it is imperative you use `spreadhseet row` as trial number to merge the two files successfully.

```
#|message: false
#|echo: true
eye_behav_L2 <- L2_data %>%

  janitor::clean_names() %>%

  # Select specific columns to keep in the dataset
  dplyr::select(participant_private_id,  correct, tlpic, trpic, blpic, brpic,
condition, eng_targetword, targetword, typetl, typetr, typebl, typebr,
zone_name, zone_type,reaction_time, spreadsheet_row,  response) %>%

  # Filter the rows where 'Zone.Type' equals "response_button_image"
  dplyr::filter(zone_type == "response_button_image") %>%
```

```
  # Rename columns for easier use and readability
  dplyr::rename(
    "TL" = "tlpic",              # Rename 'tlpic' to 'TL'
    "TR" = "trpic",             # Rename 'trpic' to 'TR'
    "BL" = "blpic",            # Rename 'blpic' to 'BL'
    "BR" = "brpic",                # Rename 'brpic' to 'BR'
    "targ_loc" = "zone_name",       # Rename 'Zone.Name' to 'targ_loc'
    "subject" = "participant_private_id",  # Rename 'Participant.Private.ID'
to 'subject'
    "trial" = "spreadsheet_row",    # Rename 'spreadsheet_row' to 'trial'
    "acc" = "correct",              # Rename 'Correct' to 'acc' (accuracy)
    "RT" = "reaction_time"          # Rename 'Reaction.Time' to 'RT'
  ) %>%


  # Convert the 'RT' (Reaction Time) column to numeric type
  mutate(RT = as.numeric(RT),
         subject=as.factor(subject),
         trial=as.factor(trial))
```

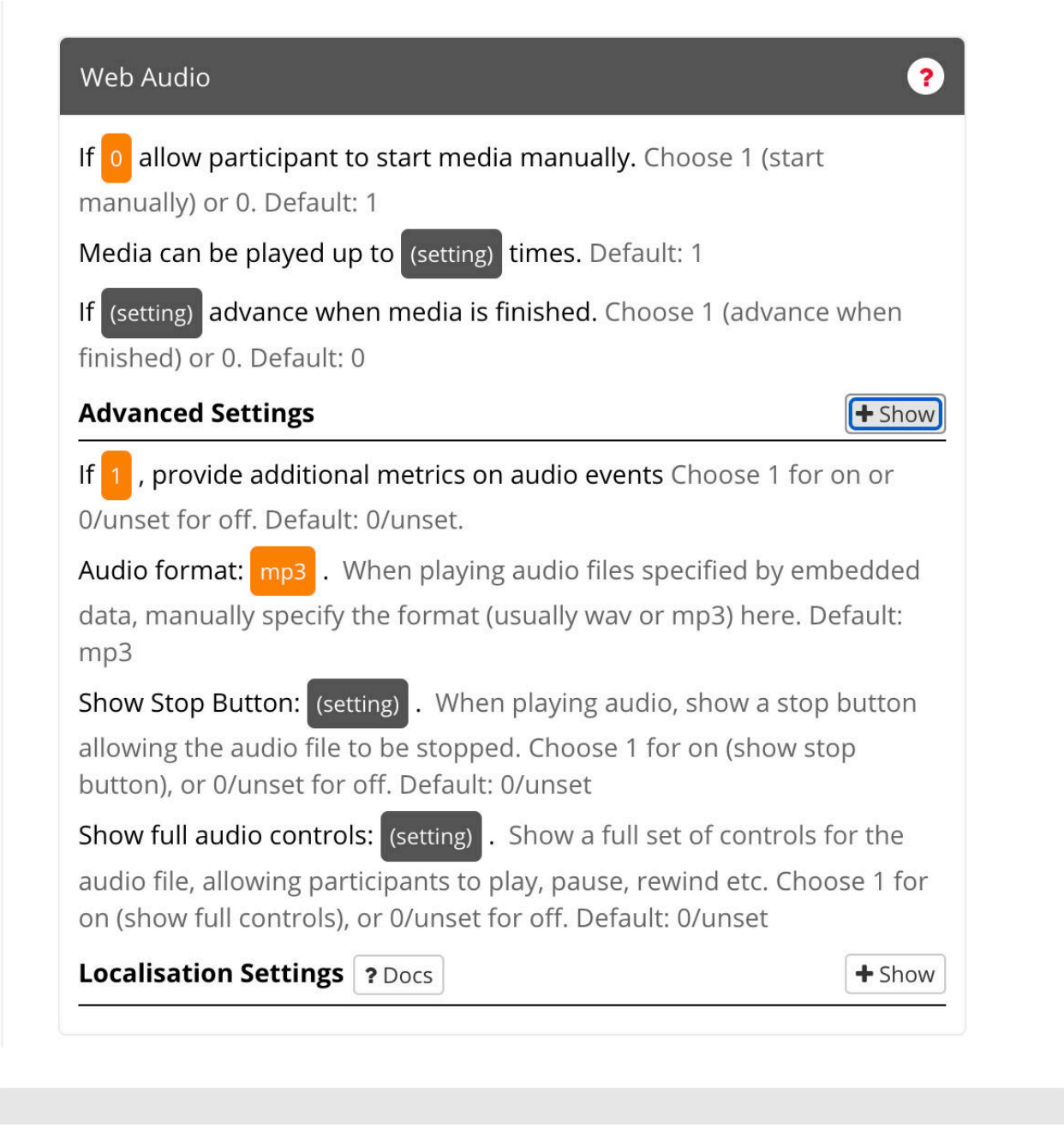
Audio onset

Because we are using spoken audio on each trial and running this experiment from the browser, audio onset is never going to to consistent across participants. In Gorilla there is an option to collect advanced audio features (you must make sure you select this when designing the study) such as when the audio play was requested, fired (played) and when the audio ended. To do so you must click on advanced settings and select 1 (see Figure 2). We will want to incorporate this timing information into our analysis pipeline. Gorilla records the onset of the

audio which varies by participant. We are extracting that in the `audio_rt_L2` object by filtering `zone_type` to `content_web_audio` and response equal to "AUDIO PLAY EVENT FIRED". This will tell us when the audio was triggered in the experiment. We are creating a column called (`RT_audio`) which we will use later on to correct for audio delays.

Figure 2

Advanced audio settings in Gorilla



```
audio_rt_L2 <- L2_data %>%


  janitor::clean_names()%>%


select(participant_private_id,zone_type, spreadsheet_row, reaction_time,

response) %>%


  filter(zone_type=="content_web_audio", response=="AUDIO PLAY EVENT

FIRED")%>%
  distinct() %>%
dplyr::rename("subject" = "participant_private_id",
       "trial" ="spreadsheet_row",
       "RT_audio" = "reaction_time") %>%
select(-zone_type) %>%
mutate(RT_audio=as.numeric(RT_audio))
```

We then merge this information with `eye_behav_L2`.

```
# merge the audio Rt data to the trial level object
trial_data_rt_L2 <- merge(eye_behav_L2, audio_rt_L2, by=c("subject", "trial"))
```

Trial removal

As stated above, participants who did not successfully calibrate 3 times or less were rejected from the experiment. Let's take a look at how many trials each participant had using the `trial_data_rt_L2` object. Deciding to remove trials is ultimately up to the researcher. In our case, we remoced participans who have less than 100 trials. In Table 2 we can see several participants failed some of the calibration attempts and do not have an adequate number of trials.

Again we make no strong recommendations here. If you to decide to do this, we recommend pre-registrating this decision.

```
# find out how many trials each participant had
edatntrials_L2 <-trial_data_rt_L2 %>%
 dplyr::group_by(subject)%>%
 dplyr::summarise(ntrials=length(unique(trial)))
```

Table 2

Trials by participant Experiment 2

subject	ntrials
12102265	2
12102286	237
12102530	239
12110559	239
12110579	173
12110585	239
12110586	239
12110600	104
12110638	55
12110685	239
12110713	228
12110751	240
12110829	59
12110878	59
12110890	116
12110891	230
12110897	60
12111092	114
12111234	57
12111244	58
12111363	58
12111367	239
12111379	238
12111410	178
12111423	237
12111514	238
12111624	238
12111663	57
12111703	58
12111795	240
12111866	240
12111869	60
12111910	114
12111960	46
12112152	59
12210934	235
12210955	238
12211266	118
12211290	240
12211353	236
12211956	226
12212098	238
12212113	56
12212204	237
12212388	118
12212716	240
12212723	239
12212808	239
12213162	236
12213496	239
12213685	238
12213754	233
12213794	239
12213826	99
12213892	237
12213965	59
12213971	237
12214172	238
12214281	235

Let's remove them from the analysis using the below code.

```
trial_data_rt_L2 <- trial_data_rt_L2 %>%
  filter(subject %in% edatntrials_bad_L2$subject)
```

Low accuracy

In our experiment, we want to make sure accuracy is high (> 80%). Again, we want participants that are fully attentive in the experiment. In the below code, we keep participants with accuracy equal to or above 80% and only include correct trials and save it to `trial_data_acc_clean_L2`.

```
# Step 1: Calculate mean accuracy per subject and filter out subjects with
mean accuracy < 0.8
subject_mean_acc_L2 <- trial_data_rt_L2 %>%
  group_by(subject) %>%
  dplyr::summarise(mean_acc = mean(acc, na.rm = TRUE)) %>%
  filter(mean_acc > 0.8)


# Step 2: Join the mean accuracy back to the main dataset and exclude trials
with accuracy < 0.8
trial_data_acc_clean_L2 <- trial_data_rt_L2 %>%
  inner_join(subject_mean_acc_L2, by = "subject") %>%
  filter(acc==1) # only use accurate responses for fixation analysis
```

RTs

There is much debate on what the proper procedure is to use to remove RTs that will bias the data the least, and if they should even be removed. Because of the ambiguity We do not remove RTs in this experiment.

Sampling Rate

While most commercial eye-trackers sample at a constant rate, data captured by webcams are widely inconsistent. Below is some code to calculate the sampling rate of each participant. Ideally, you should not have a sampling rate less than 5 Hz. It has been recommended you drop those values (Bramlett & Wiener, 2024) The below function `analyze_sample_rate()` calculates calculates the sampling rate for each subject and each trial in our eye-tracking dataset (`edat_L2`). The function provides overall statistics, including the median (used by (Bramlett & Wiener, 2024)) and standard deviation of sampling rates in your experiment,and also generates a histogram of median sampling rates by subject. Looking at Figure 3, the sampling rate ranges from 5 to 35 Hz with a median sampling rate of 21.56. This corresponds to previous webcam eye-tracking work (e.g., (Bramlett & Wiener, 2024; Prystauka et al., 2024))

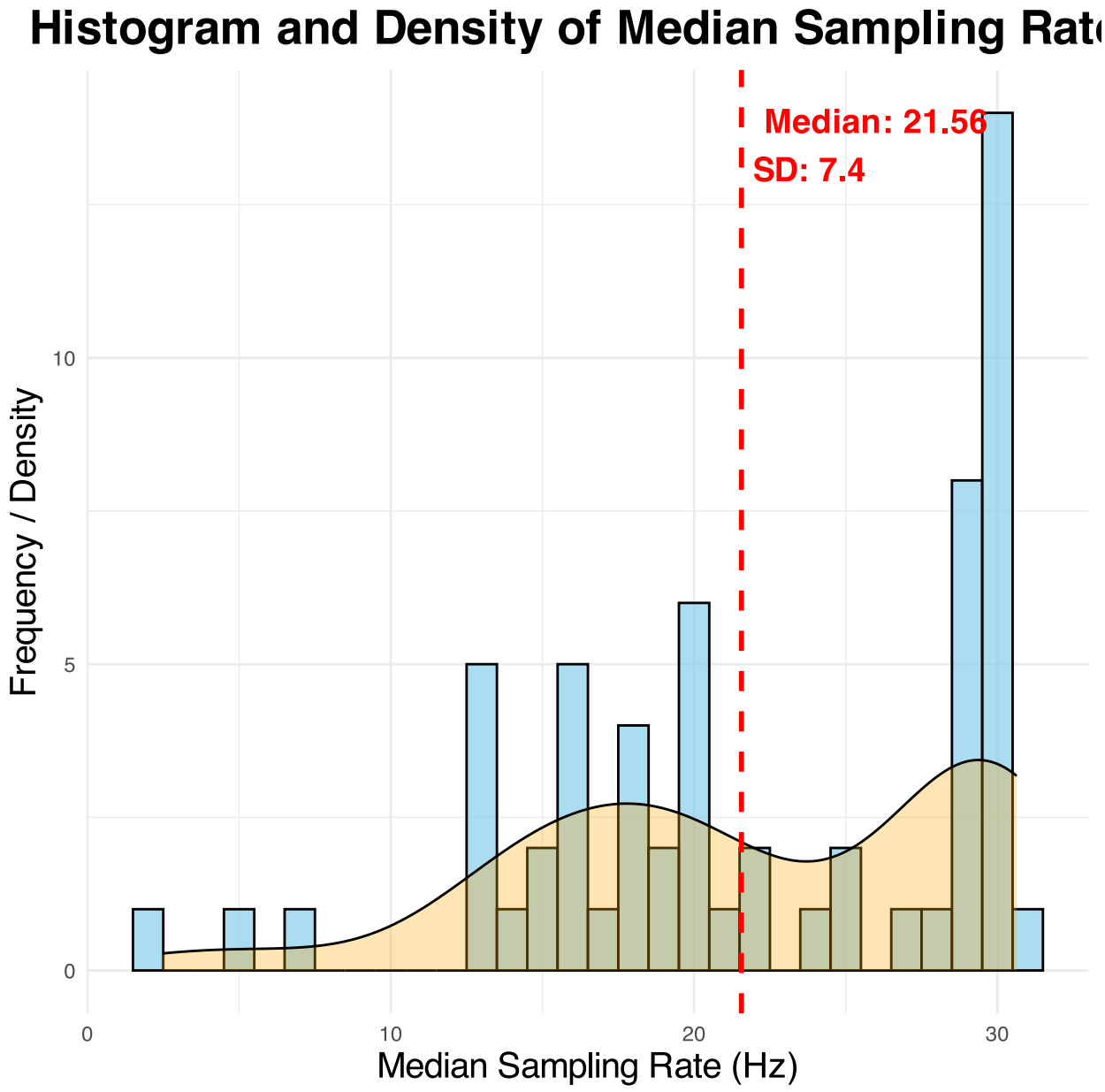
```
samp_rate_L2 <- analyze_sampling_rate(edat_L2)
```

```
Overall Median Sampling Rate (Hz): 21.56171771

Overall Standard Deviation of Sampling Rate (Hz): 7.399937723


Sampling Rate by Trial:

# A tibble: 10,665 × 5

# Groups:   subject [60]

   subject  trial max_time n_times    SR

   <fct>    <fct>    <dbl>   <int> <dbl>

 1 12102265 8        4895      108  22.1

 2 12102265 11       4920.     112  22.8

 3 12102265 15       4911.      79  16.1

 4 12102265 17       4916.     113  23.0
```

```
 5 12102265 20       4903.     112  22.8

 6 12102265 21       1826.      40  21.9

 7 12102265 28       4917.     114  23.2

 8 12102265 31       4913.      79  16.1

 9 12102265 34       4948.      88  17.8

10 12102265 35       4901.      93  19.0

# ℹ 10,655 more rows


Median Sampling Rate by Subject:

# A tibble: 60 × 2

   subject  med_SR

   <fct>     <dbl>

 1 12102265  21.9

 2 12102286  30.6

 3 12102530  19.9

 4 12110559  29.3

 5 12110579  13.3

 6 12110585  30.1

 7 12110586  14.8

 8 12110600   2.47

 9 12110638  29.0

10 12110685  19.5

# ℹ 50 more rows
```

Figure 3

Participant sampling-rate for L2 experiment. A histogram and overlayed density plot shows median sampling rate by participant. the overall median and SD is highlighted in red.



When using the above function, separate dataframes are produced by-participants and by-trial. These can be added to the behavioral dataframe using the below code.

```
# Extract by-subject and by-trial sampling rates from the result
subject_sampling_rate_L2 <- samp_rate_L2$median_SR_by_subject  # Sampling rate
by subject
trial_sampling_rate_L2 <- samp_rate_L2$SR_by_trial  # Sampling rate by trial
trial_sampling_rate_L2$subject<-as.factor(trial_sampling_rate_L2$subject)


# Assuming target_data is your other dataset that contains subject and trial
information
# Append the by-subject sampling rate to target_data (based on subject)
subject_sampling_rate_L2$subject <-
as.factor(subject_sampling_rate_L2$subject)


# Assuming target_data is your other dataset that contains subject and trial
information
# Append the by-subject sampling rate to target_data (based on subject)
trial_sampling_rate_L2$subject<-as.factor(trial_sampling_rate_L2$subject)


# Assuming target_data is your other dataset that contains subject and trial
information
# Append the by-subject sampling rate to target_data (based on subject)
subject_sampling_rate_L2$subject <-
as.factor(subject_sampling_rate_L2$subject)
trial_data_acc_clean_L2$subject <- as.factor(trial_data_acc_clean_L2$subject)


target_data_with_subject_SR_L2 <- trial_data_acc_clean_L2 %>%
```

```
  left_join(subject_sampling_rate_L2, by = "subject")


target_data_with_subject_SR_L2$trial <-
as.factor(target_data_with_subject_SR_L2$trial)


# Append the by-trial sampling rate to target_data (based on subject and
trial)
target_data_with_full_SR_L2 <- target_data_with_subject_SR_L2 %>%
  select(subject, trial, med_SR)%>%
  full_join(trial_sampling_rate_L2, by = c("subject", "trial"))
```

```
trial_data_L2 <- left_join(trial_data_acc_clean_L2,
target_data_with_full_SR_L2, by=c("subject", "trial"))
```


Now we can use this information to filter out data with poor sampling rates. Users can use the `filter_sampling_rate()` function to either (1) throw out data, by-participant, by-trial, or both, or (2) label sampling rates below a certain threshold as bad (TRUE or FALSE). Let's use the `filter_sampling_rate()` function to do this. We will use our `trial_data_L2` object.

We leave it up to the user to decide what to do with low sampling rates and make no specific recommendations here. In our case we are going to remove the data by-participant and by-trial (setting `action` = "both") if sampling frequency is below 5hz (`threshold`=5). The `filter_sampling_rate()` function is designed to process a dataset containing participant-level and trial-level sampling rates. It allows the user to either filter out data that falls below a certain sampling rate threshold or simply label it as "bad". The function gives flexibility by allowing the threshold to be applied at the participant-level, trial-level, or both. It also lets the user decide

whether to remove the data or flag it as below the threshold without removing it. If `action` = remove, the function will output how many subjects and trials were removed by on the threshold.

```
filter_edat_L2 <- filter_sampling_rate(trial_data_L2,threshold = 5,
                                       action = "remove",
                                       by = "both")
```

The message produced states that 1 subject is thrown out along with 107 trials (trials associated with the 1 subject).

Out-of-bounds (outside of screen)

It is important that we do not include points that fall outside the standardized coordinates (0,1). The `gaze_oob()` function calculates how many of the data points fall outside the standardized range. Here we need our eye-tracking data (`edat_L2`). Running the `gaze_oob()` function returns a table listing how many data points fall outside this range (total, X and Y), and also provides percentages (see Table 3). This information would be useful to include in the final paper.


```
oob_data_L2 <- gaze_oob(edat_L2)
```

Table 3

Out of bounds gaze statistics

subject	total_points	outside_count	total_missing_percentage	x_outside_count	y_outside_count	x_outside_percentage	y_outside_percentage
12102265	6197	1132	18.26690334033	202	947	3.25964176214	15.28158786510
12102286	11765	354	3.00892477688	267	181	2.26944326392	1.53846153846
12102530	9025	385	4.26592797784	244	147	2.70360110803	1.62880886427
12110559	11890	416	3.49873843566	194	222	1.63162321278	1.86711522288
12110579	5822	1063	18.25833047063	697	436	11.97183098592	7.48883545173
12110585	13974	776	5.55317017318	83	694	0.59396021182	4.96636610849
12110586	5281	216	4.09013444423	176	77	3.33270213975	1.45805718614
12110600	731	1	0.13679890561	1	0	0.13679890561	0.00000000000
12110638	4647	1056	22.72433828276	996	159	21.43318269852	3.42156229826
12110685	6356	1099	17.29074889868	1006	241	15.82756450598	3.79169288861
12110713	12647	1107	8.75306396774	787	538	6.22281964102	4.25397327429
12110751	10028	995	9.92221779019	475	570	4.73673713602	5.68408456322
12110829	4016	671	16.70816733068	510	223	12.69920318725	5.55278884462
12110878	847	27	3.18772136954	13	16	1.53482880756	1.88902007084
12110890	7683	454	5.90915007159	204	251	2.65521280750	3.26695301315
12110891	8746	2277	26.03475874686	867	1579	9.91310313286	18.05396752801
12110897	295	0	0.00000000000	0	0	0.00000000000	0.00000000000
12111092	5039	1053	20.89700337369	556	606	11.03393530462	12.02619567374
12111234	4018	524	13.04131408661	90	440	2.23992035839	10.95072175212
12111244	3136	104	3.31632653061	74	30	2.35969387755	0.95663265306
12111363	3648	1020	27.96052631579	382	753	10.47149122807	20.64144736842
12111367	10325	337	3.26392251816	259	90	2.50847457627	0.87167070218
12111379	11410	2832	24.82033304119	2083	1104	18.25591586328	9.67572304996
12111410	6480	135	2.08333333333	82	53	1.26543209877	0.81790123457
12111423	9877	891	9.02095778070	684	207	6.92517971044	2.09577807026
12111501	32569	3712	11.39734102981	2003	1800	6.15001995763	5.52672787006
12111514	12729	2495	19.60091130489	2111	666	16.58417786158	5.23214706576
12111624	6761	197	2.91377015234	139	67	2.05590888922	0.99097766603
12111663	5598	1354	24.18720971776	233	1153	4.16220078599	20.59664165773
12111703	3577	1572	43.94744199049	527	1507	14.73301649427	42.13027676824
12111795	9129	1465	16.04775988608	1408	201	15.42337605433	2.20177456457
12111866	4627	550	11.88675167495	360	284	7.78041927815	6.13788631943
12111869	1485	55	3.70370370370	54	1	3.63636363636	0.06734006734
12111910	6221	653	10.49670470985	486	211	7.81224883459	3.39173766276
12111960	5902	1780	30.15926804473	705	1121	11.94510335479	18.99356150457
12112152	4062	1289	31.73313638602	448	1172	11.02904972920	28.85278188085
12210934	11338	2472	21.80278708767	1021	1709	9.00511554066	15.07320515082
12210955	4229	1744	41.23906360842	142	1668	3.35776779380	39.44194845117
12211266	7552	880	11.65254237288	313	654	4.14459745763	8.65995762712
12211290	11740	592	5.04258943782	336	361	2.86201022147	3.07495741056
12211353	9337	1218	13.04487522759	414	1008	4.43397236800	10.79575880904
12211956	2467	349	14.14673692744	256	101	10.37697608431	4.09404134576
12212098	6278	2812	44.79133482001	315	2586	5.01752150366	41.19146224912
12212113	4321	434	10.04397130294	384	72	8.88683175191	1.66628095348
12212204	8321	657	7.89568561471	406	291	4.87922124745	3.49717582021
12212388	6771	810	11.96278245459	695	128	10.26436272338	1.89041500517
12212716	8013	516	6.43953575440	303	216	3.78135529764	2.69561961812
12212723	7748	747	9.64119772845	60	717	0.77439339184	9.25400103252
12212808	7011	593	8.45813721295	422	191	6.01911282271	2.72429040080
12213162	13211	1060	8.02361668307	674	428	5.10180909848	3.23972447203
12213496	14536	4540	31.23280132086	2052	3270	14.11667583930	22.49587231701
12213685	2329	1	0.04293688278	1	0	0.04293688278	0.00000000000
12213754	9463	927	9.79604776498	79	848	0.83483039205	8.96121737293
12213794	7515	501	6.66666666667	336	228	4.47105788423	3.03393213573
12213826	4518	698	15.44931385569	356	414	7.87959274015	9.16334661355
12213892	7293	375	5.14191690662	248	133	3.40052104758	1.82366652955
12213965	1646	194	11.78614823815	60	145	3.64520048603	8.80923450790
12213971	10532	2210	20.98366881884	1872	457	17.77440182302	4.33915685530
12214172	9006	1178	13.08016877637	921	336	10.22651565623	3.73084610260
12214281	5457	678	12.42440901594	373	339	6.83525746747	6.21220450797

We can also add add by-participant and by-trial out of bounds data to our behavioral, trial-level, data (`filter_edat_L2`) and finally exclude participants and trials with more than 30% missing data. The value of 30 is just a suggestion and should not be used as a rule of thumb for all studies nor are we endorsing this value.

```
remove_missing <- oob_data_L2 %>%                                  # Start with the
`oob_data` dataset and assign the result to `remove_missing`
  select(subject, total_missing_percentage) %>%            # Select only the
`subject` and `total_missing_percentage` columns from `oob_data`
  left_join(filter_edat_L2, by = "subject") %>%               # Perform a left
join with `filter_edat` on the `subject` column, keeping all rows from
`oob_data`
  filter(total_missing_percentage < 30)  %>%                   # Filter the
data to keep only rows where `total_missing_percentage` is less than 30 %>%
na.omit()
```

Eye-tracking data

Convergence and confidence

In the eye-tracking data we need to remove rows with poor convergence and confidence scores in our eye-tracking data. The `convergence` column refers to WebGazer.js confidence in finding a face (and accurately predicting eye movements). Confidence values vary from 0 to 1, and numbers less than 0.5 suggest that the model has probably converged. `face_conf` represents the support vector machine (SVM) classifier score for the face model fit. This score indicates how strongly the image under the model resembles a face. Values vary from 0 to 1, and here numbers greater than 0.5 are indicative of a good model fit. In our `edat_L2` object we filter out convergence less than 0.5 and face confidence greater than 0.5 and save it to `edat_1_L2`

```
edat_1_L2 <- edat_L2 %>%

 dplyr::filter(convergence <= .5, face_conf >= .5) # remove poor convergnce

and face confidence
```

Combining eye and trial-level data

Next, we will combine the eye-tracking data and behavioral data. In this case, we'll use right_join to add the behavioral data to the eye-tracking data. This ensures that all rows from the eye-tracking data are preserved, even if there isn't a matching entry in the behavioral data (missing values will be filled with NA). The resulting object is called dat_L2. We use the `distinct()` function afterward to remove any duplicate rows that may arise during the join

```
dat_L2 <- right_join(edat_1_L2,remove_missing,  by = c("subject","trial"))


dat_L2 <- dat_L2 %>%

  distinct() # make sure to remove duplicate rows
```

Areas of Interest

Zone coordinates

In the lab, we can control every aspect of the experiment. Online we cant do this. Participants are going to be completing the experiment under a variety of conditions. This includes using different computers, with very different screen dimensions. To control for this, Gorilla outputs standardized zone coordinates (labeled as `x_pred_normalised` and `y_pred_normalised` in the eye-tracking file) . As discussed in the Gorilla documentation, the Gorilla lays everything out in a 4:3 frame and makes that frame as big as possible. The normalized coordinates are then expressed relative to this frame; for example, the coordinate 0.5, 0.5 will always be the center of the screen, regardless of the size of the participant's screen. We used the normalized coordinates in our analysis (in general, you should always use normalized

coordinates). However, there are a few different ways to specify the four coordinates of the screen, which I think are worth highlighting here.

Quadrant approach. One way is to make the AOIs as big as possible, dividing the screen into four quadrants. This approach has been used in several studies (e.g., (Bramlett & Wiener, 2024; Prystauka et al., 2024)). lists coordinates for the quadrant approach. Figure 4 shows how each quadrant looks in standardized space.

loc	x_normalized	y_normalized	width_normalized	height_normalized	xmin	ymin	xmax	ymax
TL	0.0	0.5	0.5	0.5	0.0	0.5	0.5	1.0
TR	0.5	0.5	0.5	0.5	0.5	0.5	1.0	1.0
BL	0.0	0.0	0.5	0.5	0.0	0.0	0.5	0.5
BR	0.5	0.0	0.5	0.5	0.5	0.0	1.0	0.5

We plot all the fixations in each of the quadrants highlighted in different colors (Figure 4), removing points outside the standardized screen space.

Figure 4

AOI coordiantes in standardized space using the quandrant appraoch

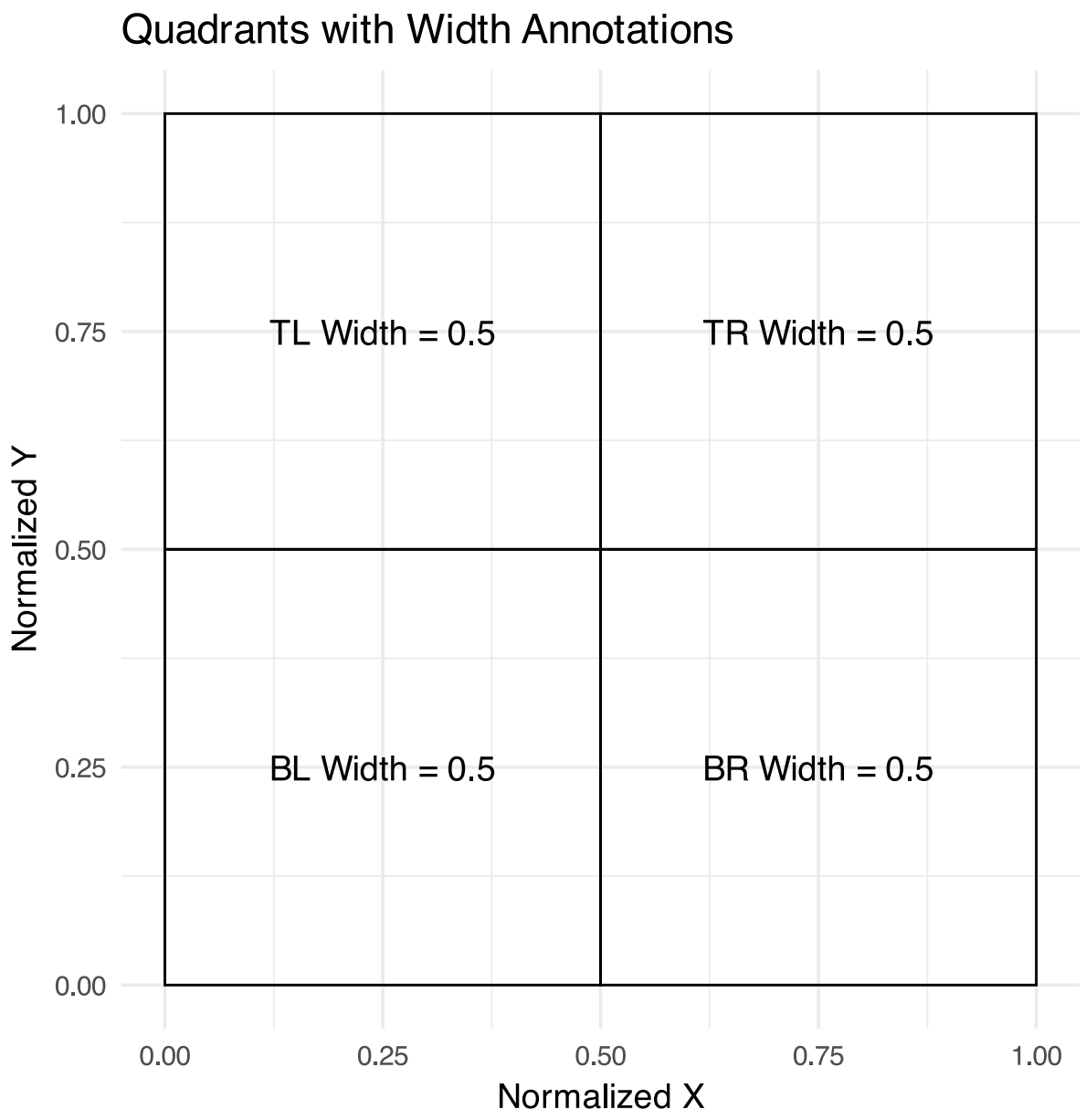
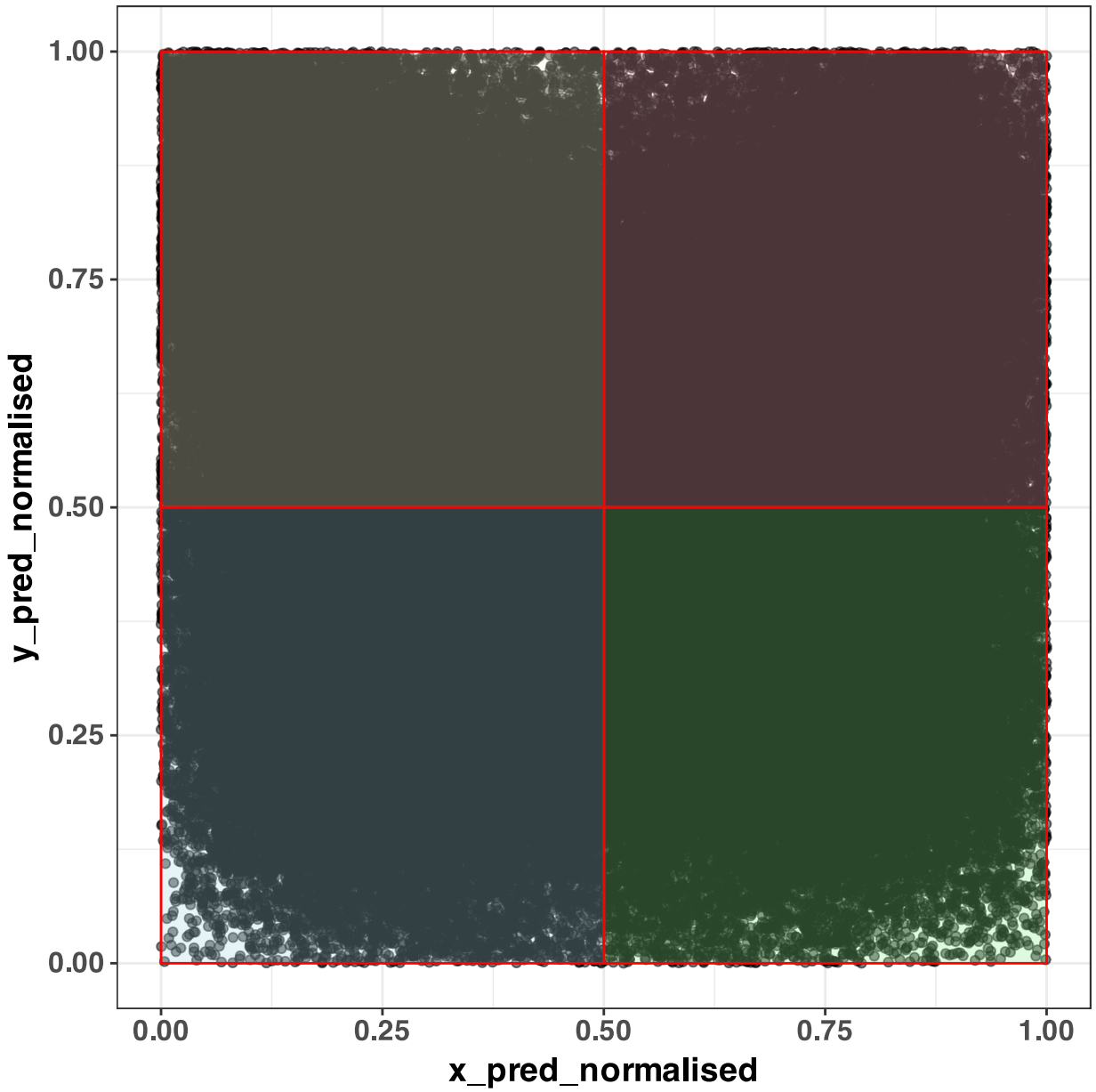


Figure 5

All looks in each of the screen quadrants



We plot all the fixations in each of the quadrants highlighted in different colors (Figure 4), removing points outside the standardized screen space. As a note, we have decided to use an outer edge approach here (eliminating eye fixations that extend beyond the screen coordinates). Bramlett and Wiener (2024) have suggested an inner-edge approach and we may add this functionality once more testing is done. For now, we believe that the otter edge approach leads to the least amount of bias in the eye-tracking pipeline.

Matching conditions with screen locations. The goal of the provided code is to assign condition codes (e.g., Target, Unrelated, Unrelated2, and Cohort) to each image in the dataset based on the screen location where the image is displayed (e.g., TL, TR, BL, BR).

For each trial, the images are dynamically placed at different screen locations, and the code maps each image to its corresponding condition based on these locations.

```
# Assuming your data is in a data frame called dat_L2
dat_L2 <- dat_L2 %>%
  mutate(
    Target = case_when(
      typetl == "target" ~ TL,
      typetr == "target" ~ TR,
      typebl == "target" ~ BL,
      typebr == "target" ~ BR,
      TRUE ~ NA_character_  # Default to NA if no match
    ),
    Unrelated = case_when(
      typetl == "unrelated1" ~ TL,
      typetr == "unrelated1" ~ TR,
      typebl == "unrelated1" ~ BL,
      typebr == "unrelated1" ~ BR,
      TRUE ~ NA_character_
    ),
    Unrelated2 = case_when(
      typetl == "unrelated2" ~ TL,
      typetr == "unrelated2" ~ TR,
```

```
      typebl == "unrelated2" ~ BL,
      typebr == "unrelated2" ~ BR,
      TRUE ~ NA_character_
    ),
    Cohort = case_when(
      typetl == "cohort" ~ TL,
      typetr == "cohort" ~ TR,
      typebl == "cohort" ~ BL,
      typebr == "cohort" ~ BR,
      TRUE ~ NA_character_
    )
  )
```


In addition to tracking the condition of each image during randomized trials, a custom function, `find_location()`, determines the specific screen location of each image by comparing it against the list of possible locations. This function ensures that the appropriate location is identified or returns NA if no match exists. Specifically, `find_location()` first checks if the image is NA (missing). If the image is NA, the function returns NA, meaning that there's no location to find for this image. If the image is not NA, the function creates a vector called loc_names that lists the names of the possible locations. It then attempts to match the given image with the locations. If a match is found, it returns the name of the location (e.g., TL, TR, BL, or BR) of the image.

```
# Apply the function to each of the targ, cohort, rhyme, and unrelated columns
dat_colnames_L2 <- dat_L2 %>%
  rowwise() %>%
```

```
  mutate(
    targ_loc = find_location(c(TL, TR, BL, BR), Target),
    cohort_loc = find_location(c(TL, TR, BL, BR), Cohort),
    unrelated_loc = find_location(c(TL, TR, BL, BR), Unrelated),
    unrealted2_loc= find_location(c(TL, TR, BL, BR), Unrelated2),
  ) %>%
  ungroup()
```

Once we do this we can use `assign_aoi()` to loop through our object called `dat_colnames_L2` and assign locations (i.e., TR, TL, BL, BR) to where participants looked at on the screen. This requires the `x` and `y` coordinates and the location of our aois `aoi_loc`. Here we are using the quadrant approach. This function will label non-looks and off screen coordinates with NA. To make it easier to read we change the numerals assigned by the function to actual screen locations (e.g., TL, TR, BL, BR).


```
assign_L2 <- gazer::assign_aoi(dat_colnames_L2,X="x_pred_normalised",
Y="y_pred_normalised",aoi_loc = aoi_loc)



AOI_L2 <- assign_L2 %>%

  mutate(loc1 = case_when(

    AOI==1 ~ "TL",

    AOI==2 ~ "TR",
```

```
    AOI==3 ~ "BL",

    AOI==4 ~ "BR"

  ))
```

In `AOI_L2` we label looks to Targets, Unrelated, and Cohort items with 1 (looked) and 0 (no look).

```
AOI_L2$target <- ifelse(AOI_L2$loc1==AOI_L2$targ_loc, 1, 0) # if in
coordinates 1, if not 0.

AOI_L2$unrelated <- ifelse(AOI_L2$loc1 == AOI_L2$unrelated_loc, 1, 0)# if in
coordinates 1, if not 0.

AOI_L2$unrelated2 <- ifelse(AOI_L2$loc1 == AOI_L2$unrealted2_loc, 1, 0)# if in
coordinates 1, if not 0.

AOI_L2$cohort <- ifelse(AOI_L2$loc1 == AOI_L2$cohort_loc, 1, 0)# if in
coordinates 1, if not 0.
```

The locations of looks need to be "gathered" or pivoted into long format—that is, converted from separate columns into a single column. This transformation makes the data easier to visualize and analyze. We use the `pivot_longer()` function from the `tidyverse` to combine the columns (Target, Unrelated, Unrelated2, and Cohort) into a single column called `condition1`.

Additionally, we create another column called `Looks`, which contains the values from the original columns (e.g., 0 or 1 for whether the area was looked at).

```
dat_long_aoi_me_L2 <- AOI_L2 %>%
  select(subject, trial, condition, target, cohort, unrelated, unrelated2,
time, x_pred_normalised, y_pred_normalised, RT_audio) %>%
  pivot_longer(
    cols = c(target, unrelated, unrelated2, cohort),
    names_to = "condition1",
    values_to = "Looks"
  )
```

We frirther clean up the object by first cleaning up the condition codes. They have a numeral appended to them and that should be removed. We then adjust the timing in the `gaze_sub_L2_comp` object by aligning time to the actual audio onset. To achieve this, we subtract `RT_audio` from time for each trial. In addition, we subtract 300 ms from this to account for the 100 ms of silence at the beginning of each audio clip and 200 ms to account for the oculomotor delay when planning an eye movement Viviani (1990). Additionally, we set our interest period between 0 ms (audio onset) and 2000 ms. This was chosen based on the time course figures in Sarrett et al. (2022) . It is important that you choose your interest area carefully and preferably you preregister it. The interest period you choose can have drastic We also filter out gaze coordinates that fall outside the standardized window, ensuring only valid data points are retained. The resulting object `gaze_sub_long_L2` provides the corrected time column spanning from –200 ms to 2000 ms relative to stimulus onset with looks outside the screen removed.


```
# repalce the numbers appended to conditions that somehow got added
dat_long_aoi_me_comp <- dat_long_aoi_me_L2 %>%
```

```
  mutate(condition = str_replace(condition, "TCUU-SPENG\\d*", "TCUU-SPENG"))
%>%
  mutate(condition = str_replace(condition, "TCUU-SPSP\\d*", "TCUU-SPSP"))%>%
  na.omit()
```

```
# dat_long_aoi_me_comp has condition corrected


gaze_sub_L2_long <-dat_long_aoi_me_comp%>%
group_by(subject, trial, condition) %>%
  mutate(time = (time-RT_audio)-300) %>% # subtract audio rt onset and account
for occ motor planning and silence in audio
 filter(time >= -200, time < 2000) %>%
   dplyr::filter(x_pred_normalised > 0,
                 x_pred_normalised < 1,
                 y_pred_normalised > 0,
                 y_pred_normalised < 1)
```

Samples to bins

Downsampling

Downsampling into smaller time bins is a common practice in gaze data analysis, as it helps create a more manageable dataset and reduces noise. When using research grade eye-trackers, it is sometimes not needed to downsample the data. However, with consumer-based webcam eye-tracking it is recommended you downsample your data so participants have consistent bin sizes (e.g., (Slim & Hartsuiker, 2023)). In `webgazeR` we included the `downsample_gaze()` function to assist with this process. We apply this function to the `gaze_sub_L2_long` object,and set the `bin.length` argument to 100, which groups the data into

100-millisecond intervals. This adjustment means that each bin now represents a 100 ms passage of time. We specify time as the variable to base these bins on, allowing us to focus on broader patterns over time rather than individual millisecond fluctuations.There is no agreed upon downsampling value, but with webcam data larger bins are preferred (Slim & Hartsuiker, 2023).

In addition, we indicate other variables for aggregation, such as condition, and use the newly created timebins variable, which represents the time intervals over which we aggregate data. There is no universal rule for selecting bin sizes, but Bramlett and Wiener (2024) found that differences in bin size did not significantly affect results as long as the sampling rate was above 5 Hz. The resulting downsampled dataset, output as Table 4, provides a simplified and more concise view of gaze patterns, making it easier to analyze and interpret broader trends.

```
gaze_sub_L2 <- downsample_gaze(gaze_sub_L2_long, bin.length=100,
timevar="time", aggvars=c("condition", "condition1", "time_bin"))
```


Table 4

Aggregated proportion looks for each condition in each 100 ms time bin

condition	condition1	time_bin	Fix
TCUU-ENGSP	cohort	−200	0.2615870787
TCUU-ENGSP	cohort	−100	0.2588888889
TCUU-ENGSP	cohort	0	0.2508226691
TCUU-ENGSP	cohort	100	0.2574108818
TCUU-ENGSP	cohort	200	0.2298850575
TCUU-ENGSP	cohort	300	0.2244212099

To simplify the analysis, we combine the two unrelated conditions and average them (this is for the proportional plots).

```
# Average Fix for unrelated and unrelated2, then combine with the rest
gaze_sub_L2 <- gaze_sub_L2 %>%
```

```
    group_by(condition, time_bin) %>%
    summarise(
        Fix = mean(Fix[condition1 %in% c("unrelated", "unrelated2")], na.rm =
TRUE),
        condition1 = "unrelated",  # Assign the combined label
        .groups = "drop"
    ) %>%
    # Combine with rows that do not include unrelated or unrelated2
    bind_rows(gaze_sub_L2 %>% filter(!condition1 %in% c("unrelated",
"unrelated2")))
```

The above will not include the subject variable. If you want to keep participant-level data we need to add `subject` to the `aggvars` argument. If you do not plan to analyze proportion data, and instead what time binned data with binary outcomes preserved please set the `aggvars` argument to "none." This will return a time binned column but will not aggregate over other variables.


```
# get back trial level data with no aggregation


gaze_sub_id <- downsample_gaze(gaze_sub_L2_long, bin.length=100,
timevar="time", aggvars="none")
```

We need to make sure we only have one unrelated value.

```
gaze_sub_id <- gaze_sub_id %>%
  mutate(condition1 = ifelse(condition1=="unrelated2", "unrelated",
condition1))
```

In `AOI_L2` we label looks to Targets, Unrelated, and Cohort items with 1 (looked) and 0 (no look).

```
AOI_L2$target <- ifelse(AOI_L2$loc1==AOI_L2$targ_loc, 1, 0) # if in
coordinates 1, if not 0.


AOI_L2$unrelated <- ifelse(AOI_L2$loc1 == AOI_L2$unrelated_loc, 1, 0)# if in
coordinates 1, if not 0.


AOI_L2$unrelated2 <- ifelse(AOI_L2$loc1 == AOI_L2$unrealted2_loc, 1, 0)# if in
coordinates 1, if not 0.


AOI_L2$cohort <- ifelse(AOI_L2$loc1 == AOI_L2$cohort_loc, 1, 0)# if in
coordinates 1, if not 0.
```


The locations of looks need to be "gathered" or pivoted into long format—that is, converted from separate columns into a single column. This transformation makes the data easier to visualize and analyze. We use the pivot_longer() function to combine the columns (Target, Unrelated, Unrelated2, and Cohort) into a single column called condition1. Additionally, we create another column called Looks, which contains the values from the original columns (e.g., 0 or 1 for whether the area was looked at).

```
dat_long_aoi_me_L2 <- AOI_L2  %>%
  select(subject, trial, condition, target, cohort, unrelated, unrelated2,
time, x_pred_normalised, y_pred_normalised, RT_audio) %>%
    pivot_longer(
        cols = c(target, unrelated, unrelated2,cohort),
```

```
        names_to = "condition1",
        values_to = "Looks"
    )
```

Samples to bins

We can further clean up the data to improve its accuracy and relevance. Since there is a delay in audio presentation, captured as `RT_audio` in our dataset, we adjust the timing in the `gaze_sub_L2_comp` dataset by aligning time to the actual audio onset. To achieve this, we subtract `RT_audio` from time for each trial. In addition, we subtract 300 ms from this to account for the 100 ms of silence at the beginning of each audio clip and 200 ms to account for the oculomotor delay when planning an eye movement Viviani (1990). Additionally, we set our interest period between 0 ms (audio onset) and 2000 ms. This was chosen based on the time course figures in Sarrett et al. (2022) . It is important that you choose your interest area carefully and preferably you preregister it. We also filter out gaze coordinates that fall outside the standardized window, ensuring only valid data points are retained. The resulting dataset `gaze_sub_long_L2` provides the corrected time column spanning from –200 ms to 2000 ms relative to stimulus onset with looks outside the screen removed.


```
# repalce the numbers appended to conditions that somehow got added
dat_long_aoi_me_comp <- dat_long_aoi_me_L2 %>%
  mutate(condition = str_replace(condition, "TCUU-SPENG\\d*", "TCUU-SPENG"))
%>%
  mutate(condition = str_replace(condition, "TCUU-SPSP\\d*", "TCUU-SPSP"))%>%
  na.omit()
```

```
# dat_long_aoi_me_comp has condition corrected
```

```
gaze_sub_L2_long <-dat_long_aoi_me_comp%>%
group_by(subject, trial, condition) %>%
  mutate(time = (time-RT_audio)-300) %>% # subtract audio rt onset and account
for occ motor planning and silence in audio
 filter(time >= -200, time < 2000) %>%
   dplyr::filter(x_pred_normalised > 0,
                 x_pred_normalised < 1,
                 y_pred_normalised > 0,
                 y_pred_normalised < 1)
```

Downsampling

Downsampling into smaller time bins is a common practice in gaze data analysis, as it helps create a more manageable dataset and reduces noise. When using research grade eye-trackers, it is sometimes not needed to downsample the data. However, with consumer-based webcam eye-tracking it is recommended you downsample your data so participants have consistent bin sizes (e.g., (Slim & Hartsuiker, 2023)). In `webgazeR` we included the `downsample_gaze()` function to assist with this process. We apply this function to the `gaze_sub_L2_long` object,and set the `bin.length` argument to 100, which groups the data into 100-millisecond intervals. This adjustment means that each bin now represents a 100 ms passage of time. We specify time as the variable to base these bins on, allowing us to focus on broader patterns over time rather than individual millisecond fluctuations.There is no agreed upon downsampling value, but with webcam data larger bins are preferred (Slim & Hartsuiker, 2023).

In addition, we indicate other variables for aggregation, such as condition, and use the newly created timebins variable, which represents the time intervals over which we aggregate data. There is no universal rule for selecting bin sizes, but Bramlett and Wiener found that differences in bin size did not significantly affect results as long as the sampling rate was above 5

Hz. The resulting downsampled dataset, output as Table 4, provides a simplified and more concise view of gaze patterns, making it easier to analyze and interpret broader trends.

```
gaze_sub_L2 <- downsample_gaze(gaze_sub_L2_long, bin.length=100,
timevar="time", aggvars=c("condition", "condition1", "time_bin"))
```

To simplify the analysis, we combine the two unrelated conditions and average them (this is for the proportional plots).

```
# Average Fix for unrelated and unrelated2, then combine with the rest
gaze_sub_L2 <- gaze_sub_L2 %>%
    group_by(condition, time_bin) %>%
    summarise(
        Fix = mean(Fix[condition1 %in% c("unrelated", "unrelated2")], na.rm =
TRUE),
        condition1 = "unrelated",  # Assign the combined label
        .groups = "drop"
    ) %>%
    # Combine with rows that do not include unrelated or unrelated2
    bind_rows(gaze_sub_L2 %>% filter(!condition1 %in% c("unrelated",
"unrelated2")))
```

The above will not include the subject variable. If you want to keep participant-level data we need to add `subject` to the `aggvars` argument. If you do not plan to analyze proportion data, and instead what time binned data with binary outcomes preserved please set the `aggvars` argument to "none." This will return a time binned column but will not aggregate over other variables.

```
# get back trial level data with no aggregation
```

```
gaze_sub_id <- downsample_gaze(gaze_sub_L2_long, bin.length=100,
timevar="time", aggvars="none")
```

We need to make sure we only have one unrelated value.

```
gaze_sub_id <- gaze_sub_id %>%
  mutate(condition1 = ifelse(condition1=="unrelated2", "unrelated",
condition1))
```

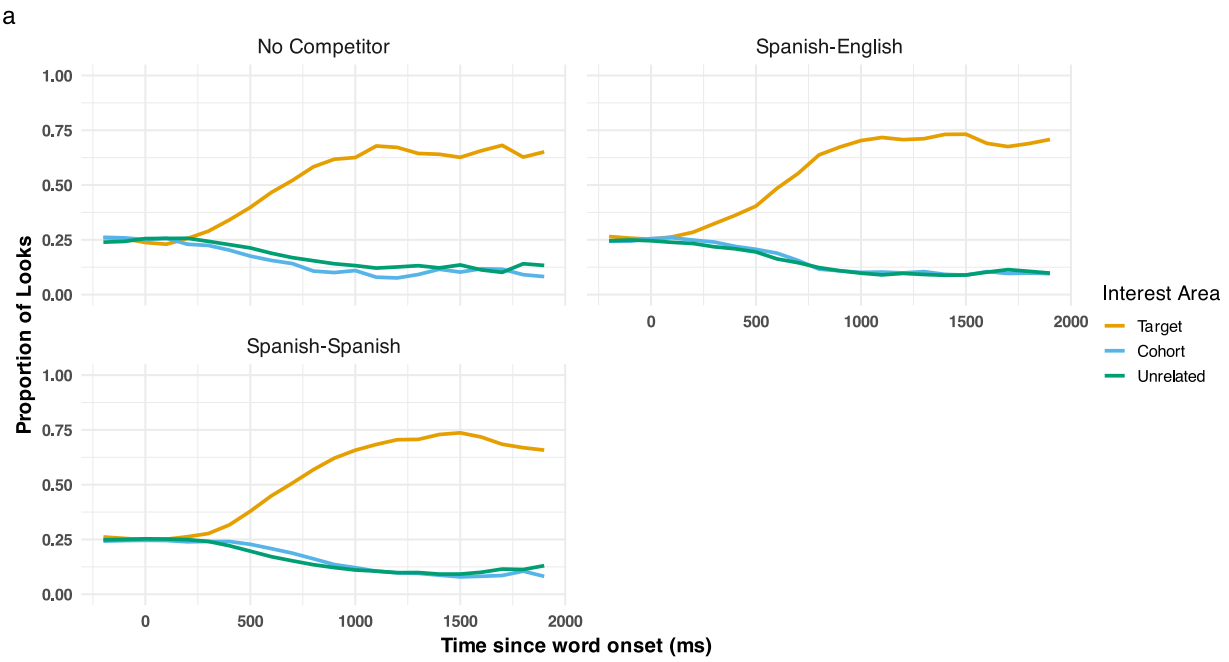
Visualizing time course data

To simplify plotting your time-course data, we have created the `plot_IA_proportions()` function. This function takes several arguments. The `ia_column` argument specifies the column containing your Interest Area (IA) labels. The `time_column` argument requires the name of your time bin column, and the `proportion_column` argument specifies the column containing fixation or look proportions. Additional arguments allow you to specify custom names for each IA in the `ia_mapping` argument, enabling you to label them as desired. In order to use this function, you must use the `downsample_gaze()` function.

Below we have plotted the time course data in Figure 6 for each condition. By default the graphs are using a color-blind friendly palette from the `ggokabeito` package (Barrett, 2021).

Figure 6

Comparison of L2 competition effects.



Gorilla provided coordinates. Thus far, we have used the coordinates representing the four quadrants of the screen. However, Gorilla provides their own quadrants representing image location on the screen. To the authors' knowledge, these quadrants have not been looked at in any studies reporting eye-tracking results. Let's examine how reasonable our results are with these coordinates.

We will use the function `extract_aois` to get the standardized coordinates for each quadrant on screen. You can use the `zone_names` argument to get the zones you want to use. In our case we want the `TL`, `BR`, `BL` `TR` coordinates. We input our object that contains all our eye-tracking files and this extract the coordinates. These are labled in Table 5. In Figure 7 we can see that the AOIs are a bit smaller than then when using the quadrant approach. We can take these coordinates and use them in our analysis.

We are not going to highlight the steps here as they are the same as above. we are just repalcing hre coordinates.

```
# apply the extract_aois fucntion

aois_L2 <- extract_aois(vwp_paths_filtered_L2, zone_names =  c("TL", "BR",

"TR", "BL"))
```

```
knitr::include_graphics(

  "gorilla_cords.png")
```

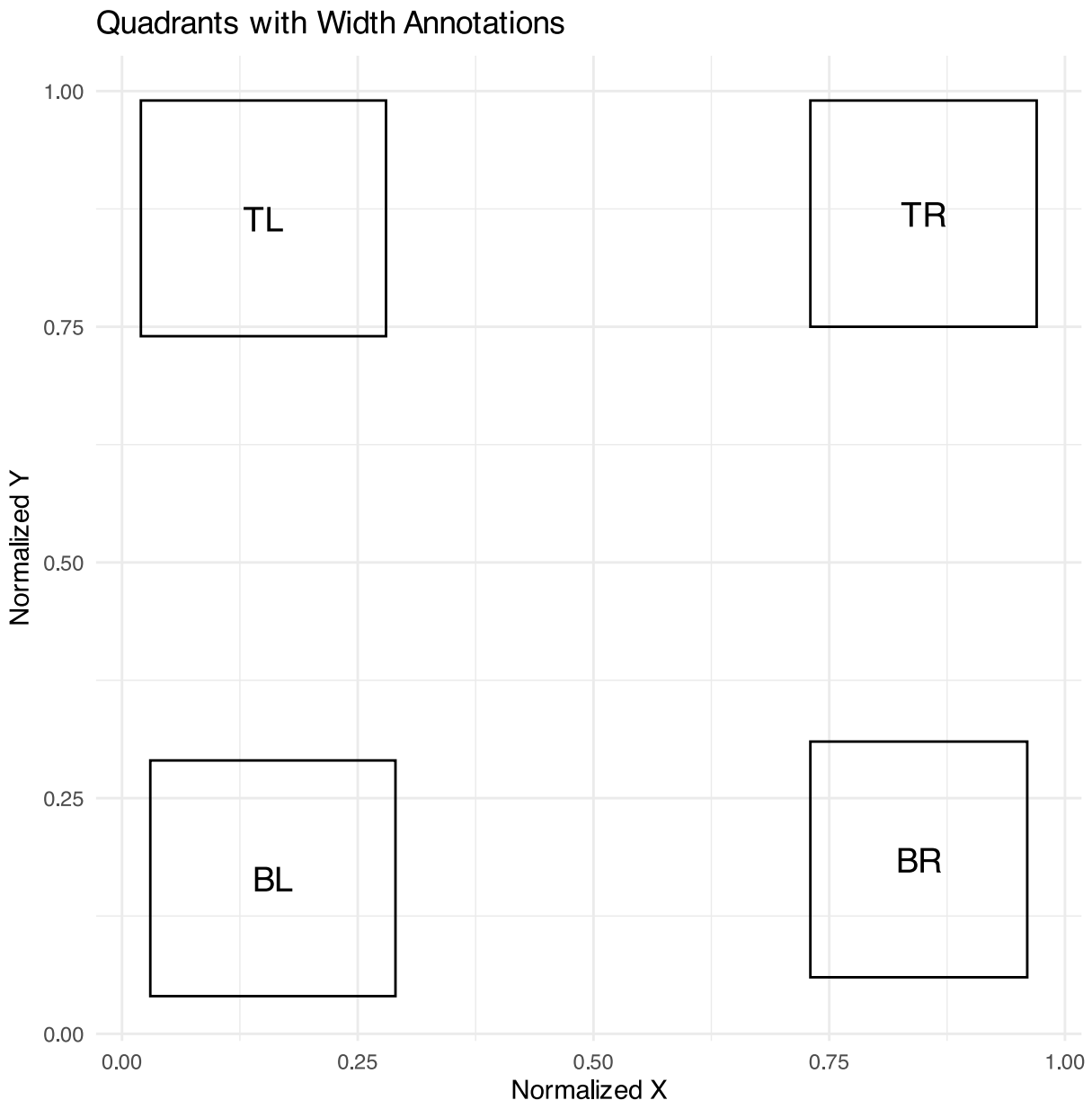
Table 5

Gorilla provided gaze coordinates

loc	x_normalized	y_normalized	width_normalized	height_normalized	xmin	ymin	xmax	ymax
BL	0.03	0.04	0.26	0.25	0.03	0.04	0.29	0.29
TL	0.02	0.74	0.26	0.25	0.02	0.74	0.28	0.99
TR	0.73	0.75	0.24	0.24	0.73	0.75	0.97	0.99
BR	0.73	0.06	0.23	0.25	0.73	0.06	0.96	0.31

Figure 7

Gorilla provided standardized coordinates for the four qudrants on the screen




```
assign_L2_gor <- webgazeR::assign_aoi(dat_colnames_L2,X="x_pred_normalised",

Y="y_pred_normalised",aoi_loc = aois_L2)



AOI_L2_gor <- assign_L2_gor %>%
```

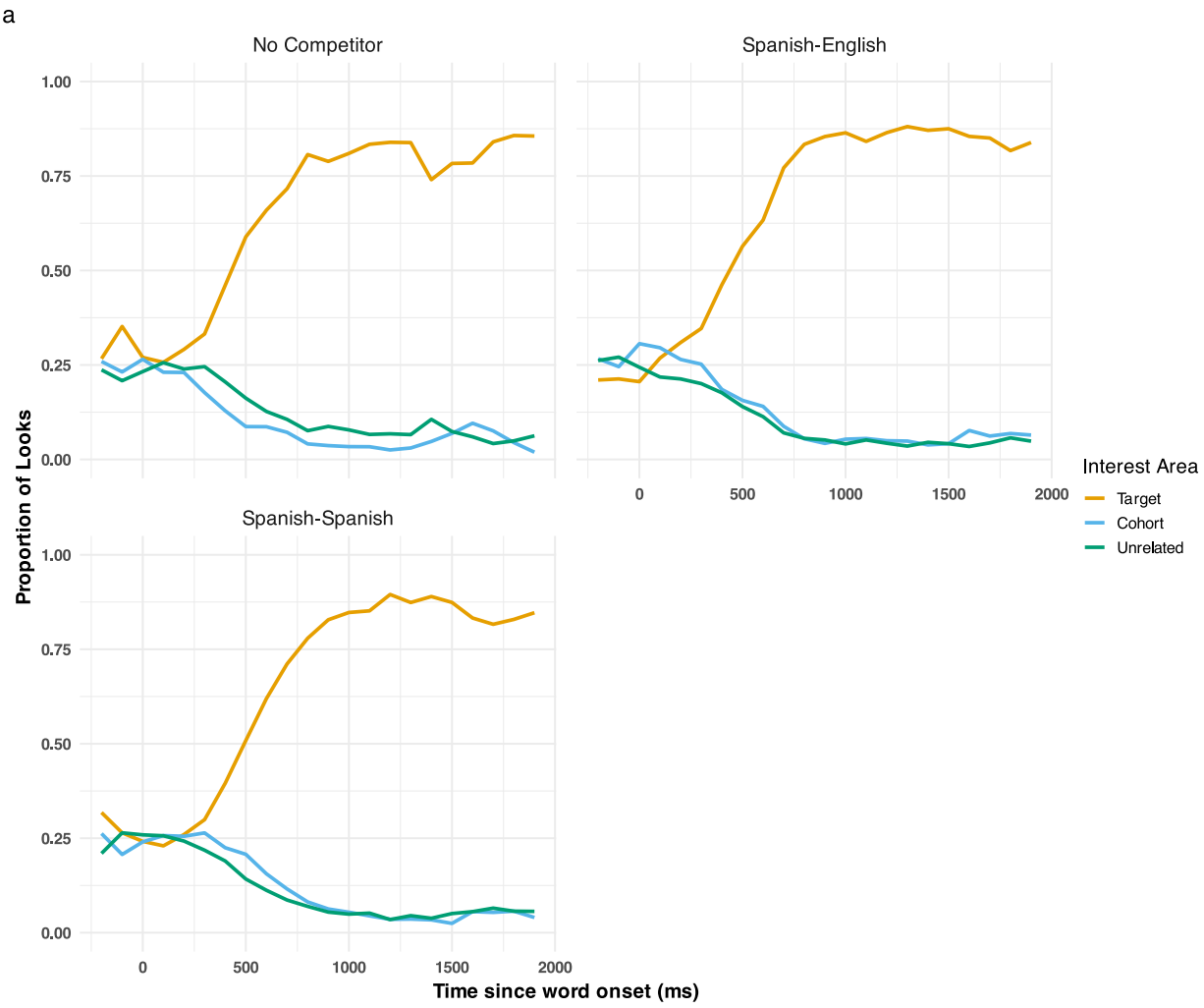
```
  mutate(loc1 = case_when(


    AOI==1 ~ "BL",


    AOI==2 ~ "TL",


    AOI==3 ~ "TR",


    AOI==4 ~ "BR"


  ))
```

```
# Average Fix for unrelated and unrelated2, then combine with the rest
gaze_sub_L2_gor  <- gaze_sub_L2_gor %>%
    group_by(condition, time_bin) %>%
    summarise(
        Fix = mean(Fix[condition1 %in% c("unrelated", "unrelated2")], na.rm =
TRUE),
        condition1 = "unrelated",  # Assign the combined label
        .groups = "drop"
    ) %>%
    # Combine with rows that do not include unrelated or unrelated2
    bind_rows(gaze_sub_L2_gor %>% filter(!condition1 %in% c("unrelated",
"unrelated2")))
```

Visualizing time course data with Gorilla coordinates

Figure 8

Comparison of competition effects with Gorilla standardized cooridnates



The Gorilla provided coordinates show a similar pattern to the quadrant approach.

Modeling Data

When analyzing VWP data there are many analytic approaches to choose from (e.g., growth curve analysis (GCA), cluster permutation tests (CPT), generalized additive mixed models (GAMMS), logistic multilevel models, divergent point analysis, etc.), and a lot has already been written describing these methods and applying them to visual world fixation data

from the lab (see [Stone et al. (2021); Ito and Knoeferle (2023); McMurray and Kutlu (n.d.)) and online ((Bramlett & Wiener, 2024)). This tutorial's goal, however, is to not evaluate different analytic approaches and tell readers which is the best method. All methods have there strengths and weaknesses (Ito & Knoeferle, 2023). Nevertheless, statistical modeling should be guided by the questions researchers using the VWP have and thus serious thought needs to be given to the proper analysis. In the VWP, there are two general questions one might be interested in: (1) Are there any overall difference in fixations between conditions and (2) Are there any time course differences in fixations between conditions.

With our data, one question we might want to answer is if there are any fixation differences between the cohort and unrelated conditions across the time course. One statistical approach we chose to highlight to answer this question is a cluster permutation analysis (CPA). The CPA is suitable for testing differences between two conditions or groups over an interest period while controlling for multiple comparisons and autocorrelation.

CPT

CPA is a technique that has become increasingly popular, particularly in the field of cognitive neuropsychology, for analyzing MEG and EEG data (Maris & Oostenveld, 2007). While its adoption in VWP studies has been relatively slow, it is now beginning to appear more frequently (Huang & Snedeker, 2020; Ito & Knoeferle, 2023). Notably, its use is growing in online eye-tracking studies (see [Slim and Hartsuiker (2023); Slim et al. (2024); Vos et al. (2022)).

Before I show you how to apply this method to the current dataset, I want to briefly explain what CPT is. The CPT is a data-driven approach that increases statistical power while controlling for Type I errors across multiple comparisons—exactly what we need when analyzing fixations across the time course.

The clustering procedure involves three main steps:

1. Cluster Formation: With our data, a multilevel logistic models is conducted for every data point (condition by time). Adjacent data points that surpass the mass univariate significance threshold (e.g., p < .05) are combined into clusters. The cluster-level statistic, typically the sum of the t-values (or F-values) within the cluster, is computed. By clustering adjacent significant data points, this step accounts for autocorrelation by considering temporal dependencies rather than treating each data point as independent.
2. Null Distribution Creation: A surrogate null distribution is generated by randomly permuting the conditions within subjects. This randomization is repeated n times (here 1000), and the cluster-level statistic is computed for each permutation. This step addresses multiple comparisons by constructing a distribution of cluster statistics under the null hypothesis, ensuring that family-wise error rates (FWER) are controlled.
3. Significance Testing: The cluster-level statistic from the observed (real) comparison is compared to the null distribution. Clusters with statistics falling in the highest or lowest 2.5% of the null distribution are considered significant (e.g., $p < 0.05$).

To preform CPT, we will load in the `permutaes` (Voeten, 2023), `permuco` (Frossard & Renaud, 2021), and (& Weston, 2022) packages in R so we can use the `cluster.glmer()` function to run a cluster permutation model with our `gaze_sub_id` dataset where each row in `Looks` denotes whether the AOI was fixated, with values of zero (not fixated) or one (fixated).

Below is sample code to perform multilevel CPA in R.

```
library(permutes) # cpa

library(permuco) # cpa

library(foreach) # for par


cpa.lme = permutes::clusterperm.glmer(Looks~ condition1_code + (1|subject) +
```

```
(1|trial), data=gaze_sub_L2_cp1, series.var=~time_bin, nperm = 1000,

parallel=TRUE)
```

```
knitr::include_graphics("cluster_spsp.png")
```

Table 6

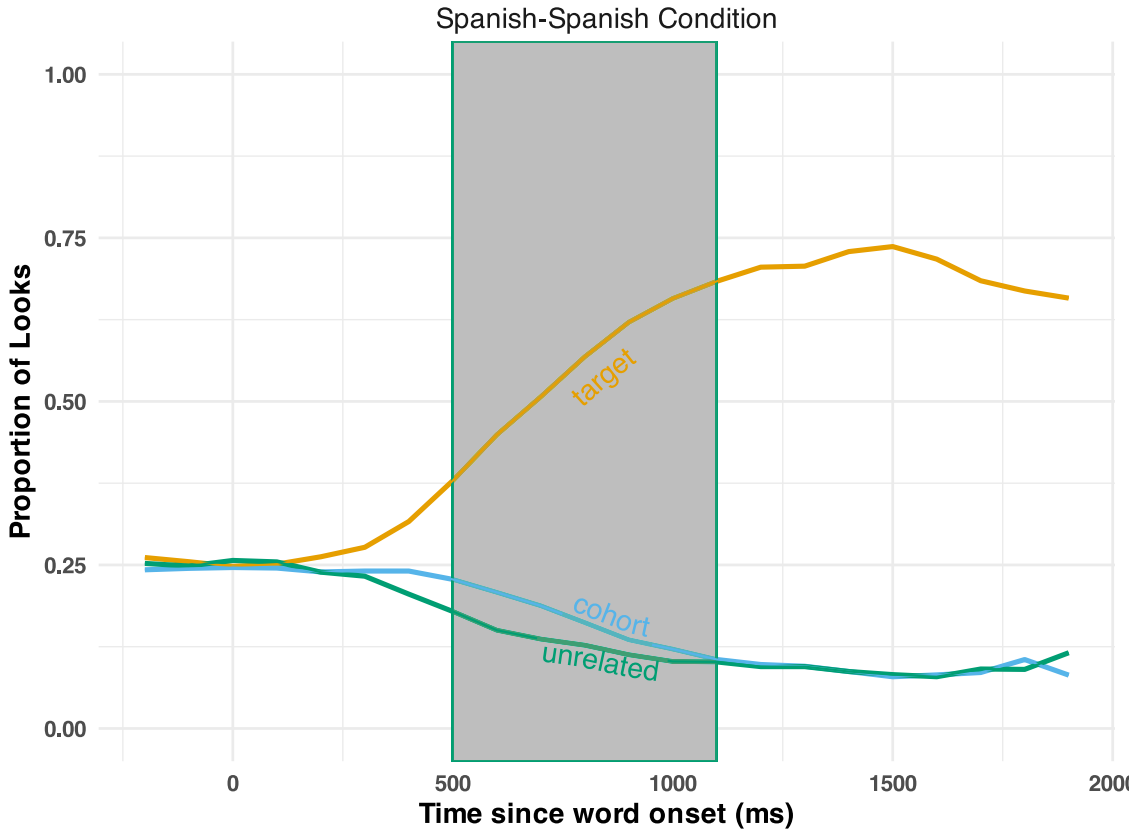
Clustermass statistics for the Spanish-Spanish condition

cluster	cluster_mass	p.cluster_mass	bin_start	bin_end	t	sign	time_start	time_end
1	210.0252112	0.000999000999	7	13	5.14215438	1	500	1100

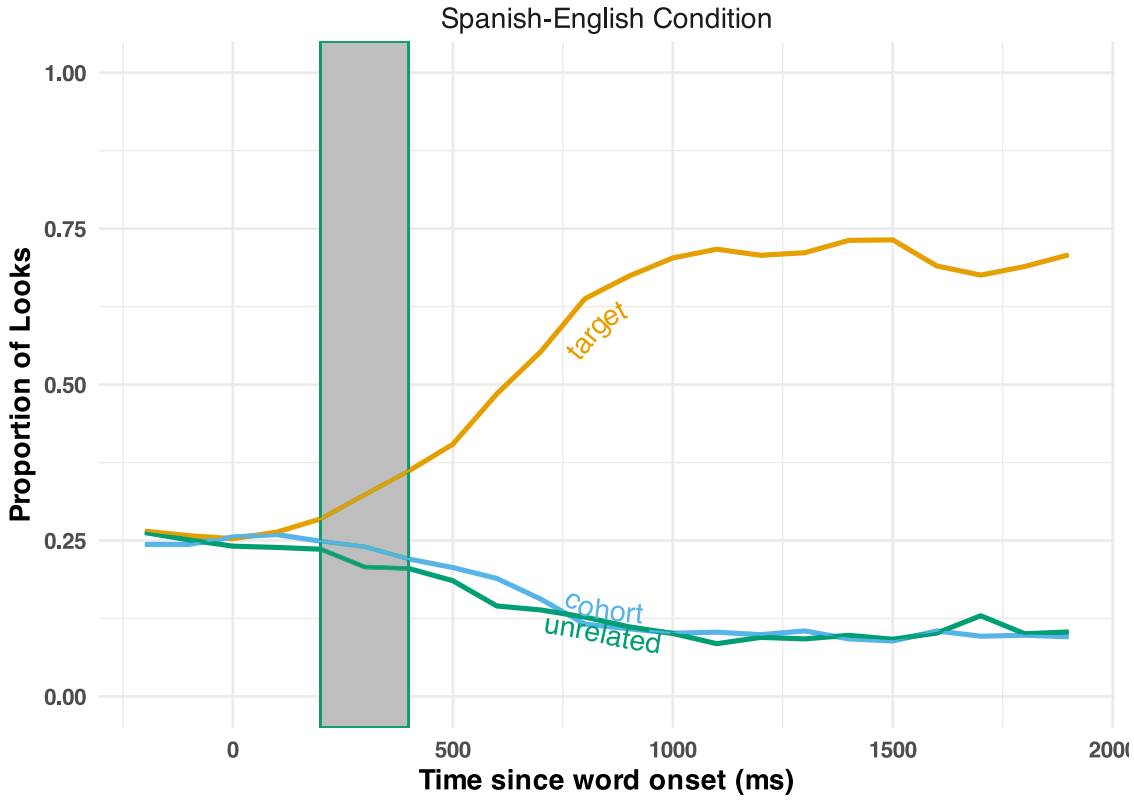
Figure 9 shows significant clusters (shaded) for both the Spanish-Spanish and Spanish-English conditions. We see there is one significant cluster in both conditions.

Figure 9

Average looks in the cross-lingustic VWP task over time for the Spanish-Spanish condition (a) and the Spanish-English condition (b). The shaded rectangles indicate when cohort looks were greater than chance baed on the CPA.


Spanish-Spanish Condition
1.00
0.75
0.50
0.25
0.00
Proportion of Looks
target
cohort
unrelated
0
500
1000
1500
2000
Time since word onset (ms)

b


Spanish-English Condition
1.00
0.75
0.50
0.25
0.00
Proportion of Looks
target
cohort
unrelated
0
500
1000
1500
2000
Time since word onset (ms)

Discussion

Webcam eye-tracking is a relatively nascent technology, and as such, there is limited guidance available for researchers. To ameliorate this, we created a tutorial to assist new users of webcam eye-tracking in analyzing their data, using some of the best practices laid out in other work (e.g., Bramlett and Wiener (2024) . To further facilitate this process, we created the `webgazeR` package, which contains several helper functions designed to streamline data preprocessing, analysis, and visualization.

In this tutorial, we covered the basic steps of running a VWP experiment online using webcam-based eye-tracking. We highlighted these steps by using data from a cross-linguistic VWP looking at competitive processes in L2 speakers of Spanish. Specifically, we attempted to replicate the experiment by Sarrett et al. (2022) where they observed within- and between L2/L1 competition using carefully crafted materials.

While the main purpose of this tutorial was to highlight the steps needed to analyze webcam eye-tracking data, replicating Sarrett et al. (2022) allowed us to not only assess whether competition can be found in a spoken word recognition VWP experiment online (one of the first studies to do so), but also provide insight in how to run VWP studies online and the issues associated with it.

Turning first to our conceptual replication attempt, the findings are very encouraging. We demonstrated competition effects both within (Spanish-Spanish condition) and across languages (Spanish-English condition), paralleling what Sarrett found. However, there are some important differences between our studies worth highlighting.

First, while Sarrett used a non-linear curve-fitting approach to examine the time course of competition, we used a cluster-based permutation approach. This methodological difference meant we could not address similar questions, though the overall temporal pattern of findings is strikingly similar. Second, we employed a truncated stimulus set, with half the number of trials

(250 vs. 450). While the number of trials is smaller than the original study, the number of trials is much larger than any of the other webcam online studies to date. Even with the fewer number of trials, we were able to obersve a similar pattern of competition in both conditions. Third, we collected all our data on Prolific with a limited set of filters, whereas Sarrett recruited students from a Spanish college course and used the LexTALE-Spanish assessment to evaluate Spanish proficiency. Lastly, Sarrett's sample consisted of adult L2 learners, whereas our sample included a broader range of L2 speakers with limited checks on their language abilities.The filters available on Prolific only allowed us to screen for native language and experience with another language, limiting our ability to refine participant selection further.

While we do not wish to downplay our findings, a more systematic study is needed to ensure their generalizability. Nonetheless, our findings underscore the potential of webcam-based eye-tracking for investigating competition effects in L2 users. We encourage researchers to continue exploring competitive processes in L2 learners.

Table 7

Eye-tracking Questionaire Items

Question
1.Do you have a history of vision problems (e.g., corrected vision, eye disease, or drooping eyelids)?
2.Are you on any medications currently that can impair your judgement?
If yes, please list below:
4.Does your room currently have natural light?
5.Are you using the built in camera?
If no, what brand of camera are you using?
6.Please estimate how far you think you were sitting from the camera during the experiment (an arm's length from your monitor is about 20 inches (51 cm).
7.Approximately how many times did you look at your phone during the experiment?
8.Approximately how many times did you get up during the experiment?
9.Was the environment you took the experiment in distraction free?
10. When you had to calibrate, were the instructions clear?
11. What additional information would you add to help make things easier to understand?
12. Are you wearing a mask?

Table 8

Responses to eye-tracking questions for participants who successfully calibrated vs. participants who had trouble calibratrin

Question	Response	percentage_good	percentage_bad
1. Do you have a history of vision problems (e.g., corrected vision, eye disease, or drooping eyelids)?	No	65.714	64.286
1. Do you have a history of vision problems (e.g., corrected vision, eye disease, or drooping eyelids)?	Yes	34.286	35.714
2. Are you on any medications currently that can impair your judgement?	No	100.000	98.214
2. Are you on any medications currently that can impair your judgement?	Yes	0.000	1.786
4. Does your room currently have natural light?	No	40.000	26.786
4. Does your room currently have natural light?	Yes	60.000	73.214
5. Are you using the built in camera?	No	14.286	8.929
5. Are you using the built in camera?	Yes	85.714	91.071
9. Was the environment you took the experiment in distraction free?	No	11.429	3.571
9. Was the environment you took the experiment in distraction free?	Yes	88.571	96.429

Limitations

While the above suggests that webcam eye-tracking is a promising avenue for language research, there are some issues that we ran into that need to be addressed. One issue that plauges webcam eye-tracking is data loss due to poor calibration. In our study, we had to throw out ~40%

of our data due to poor calibration. Other studies have shown numbers much higher (e.g.,) and lower (Yanina). Given this it is still an open question as to what contributes to better vs. poor data quality in webcam eye-tracking. To this end, we included an assessment after the VWP that included questions on the participants' experimental set-ups and overall experiences with the eye-tracking experiment. All questions are included @tbl-question.

Poor vs. Good calibrators

In our experimental design, participants were branched based on whether they successfully completed the experiment or failed calibration at any point. Table 2 highlights the comparisons between good and poor calibrators. For the sake of brevity, we do not include responses to all questions. You can look at all the responses here. However, two key differences emerge that may provide insight into factors influencing successful calibration.

One notable difference is the type of webcam used. Participants who failed calibration predominantly reported using built-in webcams, whereas those who successfully calibrated reported using a variety of external webcams. This suggests that built-in webcams may not provide sufficient resolution for calibration in the experiment. Slim and Hartsuiker (2023) performed some correlations looking at calibration score and webcam quality and noticed that high frame rate correlated with a calibration scores.

Another difference lies in the participants' environmental setup. Individuals who failed calibration were more likely to be in environments with natural light. Since natural light is known to interfere with eye-tracking, it may have contributed to their inability to calibrate successfully.

We did not notice any other differences between those that successful calibrated vs. those who did not. For researchers wanting to use webcam eye-tracking, they should try to make sure participants are in rooms without natural light, and use good web cameras. While we tried to emphasize this in our instructional videos, more explicit instruction may be needed. An avenue

for research research would be to compare lab based webcam eye-tracking to online based webcam eye tracking to see if control of the environment can produce better results.

It is important to note here that Gorilla, the experimental platform, we used uses webgazer.JS to perform it's eye tracking. It is unclear if poor calibration results from the noise introduced by participant's environment/equipment or if it is a function of the method itself, or both. We have listed some equipment and enviroemntal factors that may contribite to the poor performance; however it could be the algorthmy itself that is poor. There are other experimental platforms out there that use different eye-tracking ML algorithms to perform webcam eye-tracking (e.g., labvanced; XXX). In labanced they aslo use head motion tracking that measures the distance of the participant in front he screen to ensure head movement is restricted to an acceptable range. Together this might make for a better eye-tracking experience with less data thrown out. This should be investigated further.

Generalizability to Other Platforms In this study, we demonstrated how to analyze data from a Gorilla experiment using WebGazer.js. While we were unable to validate this pipeline on other experimental platforms, such as PCIbex (Zehr & Schwarz, 2014) or jsPsych (de Leeuw et al.), we believe that this basic pipeline will generalize to those platforms, as WebGazer.js underlies them all and provides consistent output. We encourage researchers to test this pipeline in their own studies and report any issues on our GitHub repository. We are committed to continuing improvements to **webgazeR**, ensuring that users can effectively analyze webcam eye-tracking data with our package.

Power

While we successfully demonstrated competition effects similar to Sarrett's study, we did not conduct an a priori power analysis to determine the smallest effect size of interest. Some recommendations suggest running twice the number of participants from the original sample or powering the study to detect an effect size half as large as the original (Slim). We attempted to

double the sample size but were unable to recruit enough participants through Prolific. However, our sample size is similar to the lab based studies and the effects are very similar.

We strongly urge researchers to perform power analyses and justify their sample sizes (Lakens). While tools like G*Power are available for this purpose, we recommend power simulations using Monte Carlo or resampling methods on pilot or sample data (Yanina, Slim). Several excellent R packages, such as **mixedPower**, make such simulations straightforward and accessible.

Recommendations

Based on our findings and limitations, we propose the following recommendations for researchers conducting webcam-based eye-tracking studies:

1. **Prioritize External Webcams**

   Our data showed that participants using external webcams had significantly better calibration success compared to those relying on built-in webcams. External webcams generally provide higher resolution and frame rates, which are critical for accurate eye-tracking. Researchers should encourage participants to use external webcams whenever possible.

2. **Optimize Environmental Conditions**

   Natural light was a common factor in environments where calibration failed. Researchers should advise participants to conduct experiments in rooms with controlled lighting—ideally, artificial lighting with minimal glare or shadows—to reduce interference with eye-tracking accuracy.

3. **Conduct A Priori Power Analysis and Simulations**

   To ensure adequate statistical power, researchers should conduct a priori power analyses either via GUI like GPower or use perform Monte Carlo simulations on pilot data. This step is particularly important for online studies, where sample variability can be higher than in

controlled lab environments. Tools like **G*Power** and R packages such as **mixedPower** can simplify these calculations.

4. **Collect Detailed Post-Experiment Feedback**

   Including post-experiment questionnaires about participants' setups (e.g., webcam type, browser, lighting conditions) can provide valuable insights into calibration success factors. These data can help refine participant instructions and inclusion criteria for future studies.

By adhering to these recommendations, researchers can enhance the reliability and generalizability of their webcam eye-tracking studies, ensuring the potential of this technology is fully realized.

Conclusions

This work highlighted the steps required to process webcam eye-tracking data collected via Gorilla, showcasing the potential of webcam-based eye-tracking for robust psycholinguistic experimentation. With a standarized pipeline for processing eye-tracking data we hope we have given researchers a clear path forward when collecting and analyze webcam eye-tracking data.

Moreoever, our findings demonstrate the feasibility of conducting high-quality online experiments, paving the way for future research to address more nuanced questions about L2 processing and language comprehension more broadly. Additionally, further refinement of webcam eye-tracking methodologies could enhance data precision and extend their applicability to more complex experimental designs. This is an exciting time for eye-tracking research, with its boundaries continuously expanding. We eagerly anticipate the advancements and possibilities that the future of webcam eye-tracking will bring.

References

AB, T. (2024). *Tobii pro spectrum: Technical specifications*. https://www.tobii.com/products/eye-trackers/screen-based/tobii-pro-spectrum

Allopenna, P. D., Magnuson, J. S., & Tanenhaus, M. K. (1998). *Tracking the time c ourse of spoken word recognition using eye movements: Evidence for c ontinuous mapping models* (pp. 419–439).

Anderson, C. A., Allen, J. J., Plante, C., Quigley-McBride, A., Lovett, A., & Rokkum, J. N. (2019). The MTurkification of Social and Personality Psychology. *Personality & Social Psychology Bulletin*, *45*(6), 842–850. https://doi.org/10.1177/0146167218798821

Anwyl-Irvine, A. L., Massonnié, J., Flitton, A., Kirkham, N., & Evershed, J. K. (2020). Gorilla in our midst: An online behavioral experiment builder. *Behavior Research Methods*, *52*(1), 388–407. https://doi.org/10.3758/s13428-019-01237-x

Barrett, M. (2021). *Ggokabeito: 'Okabe-ito' scales for 'ggplot2' and 'ggraph'*. https://CRAN.R-project.org/package=ggokabeito

Blasi, D. E., Henrich, J., Adamou, E., Kemmerer, D., & Majid, A. (2022). Over-reliance on english hinders cognitive science. *Trends in Cognitive Sciences*, *26*(12), 1153–1170. https://doi.org/10.1016/j.tics.2022.09.015

Bramlett, A. A., & Wiener, S. (2024). The art of wrangling. *Linguistic Approaches to Bilingualism*. https://doi.org/https://doi.org/10.1075/lab.23071.bra

Bylund, E., Khafif, Z., & Berghoff, R. (2024). Linguistic and geographic diversity in research on second language acquisition and multilingualism: An analysis of selected journals. *Applied Linguistics*, *45*(2), 308–329. https://doi.org/10.1093/applin/amad022

Carter, B. T., & Luke, S. G. (2020). Best practices in eye tracking research. *International Journal of Psychophysiology*, *155*, 49–62. https://doi.org/10.1016/j.ijpsycho.2020.05.010

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

Degen, J., Kursat, L., & Leigh, D. D. (2021). Seeing is believing: Testing an explicit linking assumption for visual world eye-tracking in psycholinguistics. *Proceedings of the Annual Meeting of the Cognitive Science Society*, *43*.

Eberhard, K. M., Spivey-Knowlton, M. J., Sedivy, J. C., & Tanenhaus, M. K. (1995). Eye movements as a window into real-time spoken language comprehension in natural contexts. *Journal of Psycholinguistic Research*, *24*(6), 409–436. https://doi.org/10.1007/BF02143160

Firke, S. (2023). *Janitor: Simple tools for examining and cleaning dirty data*. https://CRAN.R-project.org/package=janitor

Frossard, J., & Renaud, O. (2021). *Permutation tests for regression, ANOVA, and comparison of signals: The permuco package*. *99*. https://doi.org/10.18637/jss.v099.i15

Geller, J., & Prystauka, Y. (2024). *webgazeR: Tools for processing webcam eye tracking data*. https://github.com/jgeller112/webgazeR

Godfroid (she/her), A., Finch (she/her), B., & Koh (she/her), J. (n.d.). Reporting Eye-Tracking Research in Second Language Acquisition and Bilingualism: A Synthesis and Field-Specific Guidelines. *Language Learning*, *n/a*(n/a). https://doi.org/10.1111/lang.12664

Gosling, S. D., Sandy, C. J., John, O. P., & Potter, J. (2010). Wired but not WEIRD: The promise of the Internet in reaching more diverse samples. *Behavioral and Brain Sciences*, *33*(2-3), 94–95. https://doi.org/10.1017/S0140525X10000300

Huang, Y., & Snedeker, J. (2020). Evidence from the visual world paradigm raises questions about unaccusativity and growth curve analyses. *Cognition*, *200*, 104251. https://doi.org/10.1016/j.cognition.2020.104251

Ito, A., & Knoeferle, P. (2023). Analysing data from the psycholinguistic visual-world paradigm: Comparison of different analysis methods. *Behavior Research Methods*, *55*(7), 3461–3493. https://doi.org/10.3758/s13428-022-01969-3

Izura, C., Cuetos, F., & Brysbaert, M. (2014). Lextale-Esp: a test to rapidly and efficiently assess the Spanish vocabulary size. *PSICOLOGICA*, *35*(1), 49–66. http://hdl.handle.net/1854/LU-5774107

Kaduk, T., Goeke, C., Finger, H., & König, P. (2024). Webcam eye tracking close to laboratory standards: Comparing a new webcam-based system and the EyeLink 1000. *Behavior Research Methods*, *56*(5), 5002–5022. https://doi.org/10.3758/s13428-023-02237-8

Leeuw, J. R. de. (2015). jsPsych: A JavaScript library for creating behavioral experiments in a Web browser. *Behavior Research Methods*, *47*(1), 1–12. https://doi.org/10.3758/s13428-014-0458-y

Magnuson, J. S., Dixon, J. A., Tanenhaus, M. K., & Aslin, R. N. (2007). The Dynamics of Lexical Competition During Spoken Word Recognition. *Cognitive Science*, *31*(1), 133–156. https://doi.org/10.1080/03640210709336987

Maris, E., & Oostenveld, R. (2007). Nonparametric statistical testing of EEG- and MEG-data. *Journal of Neuroscience Methods*, *164*(1), 177–190. https://doi.org/10.1016/j.jneumeth.2007.03.024

McMurray, B., & Kutlu, E. (n.d.). *From real-time measures to real world differences new [and old] statistical approaches to individual differences in real-time language processing*. https://doi.org/10.31234/osf.io/2c5b6

Microsoft, & Weston, S. (2022). *Foreach: Provides foreach looping construct*. https://CRAN.R-project.org/package=foreach

Mirman, D., & Graziano, K. M. (2012). Individual differences in the strength of taxonomic versus thematic relations. *Journal of Experimental Psychology: General*, *141*(4), 601–609. https://doi.org/10.1037/a0026451

Nyström, M., Niehorster, D. C., Andersson, R., & Hooge, I. (2021). The Tobii Pro Spectrum: A useful tool for studying microsaccades? *Behavior Research Methods*, *53*(1), 335–353. https://doi.org/10.3758/s13428-020-01430-3

Papoutsaki, A., Sangkloy, P., Laskey, J., Daskalova, N., Huang, J., & Hays, J. (2016). *Webgazer: Scalable webcam eye tracking using user interactions*. 38393845.

Peirce, J., Gray, J. R., Simpson, S., MacAskill, M., Höchenberger, R., Sogo, H., Kastman, E., & Lindeløv, J. K. (2019). PsychoPy2: Experiments in behavior made easy. *Behavior Research Methods*, *51*(1), 195–203. https://doi.org/10.3758/s13428-018-01193-y

Płużyczka, M. (2018). The First Hundred Years: a History of Eye Tracking as a Research Method. *Applied Linguistics Papers*, *25/4*, 101–116. http://cejsh.icm.edu.pl/cejsh/element/bwmeta1.element.desklight-98576d43-39e3-4981-8c1c-717962cf29da

Prystauka, Y., Altmann, G. T. M., & Rothman, J. (2024). Online eye tracking and real-time sentence processing: On opportunities and efficacy for capturing psycholinguistic effects of different magnitudes and diversity. *Behavior Research Methods*, *56*(4), 3504–3522. https://doi.org/10.3758/s13428-023-02176-4

Rodd, J. M. (2024). Moving experimental psychology online: How to obtain high quality data when we can't see our participants. *Journal of Memory and Language*, *134*, 104472. https://doi.org/10.1016/j.jml.2023.104472

Sarrett, M. E., Shea, C., & McMurray, B. (2022). Within- and between-language competition in adult second language learners: Implications for language proficiency. *Language, Cognition and Neuroscience*, *37*(2), 165–181. https://doi.org/10.1080/23273798.2021.1952283

Semmelmann, K., & Weigelt, S. (2018). Online webcam-based eye tracking in cognitive science: A first look. *Behavior Research Methods*, *50*(2), 451–465. https://doi.org/10.3758/s13428-017-0913-7

Slim, M. S., & Hartsuiker, R. J. (2023). Moving visual world experiments online? A web-based replication of Dijkgraaf, Hartsuiker, and Duyck (2017) using PCIbex and WebGazer.js. *Behavior Research Methods*, *55*(7), 3786–3804. https://doi.org/10.3758/s13428-022-01989-z

Slim, M. S., Kandel, M., Yacovone, A., & Snedeker, J. (2024). Webcams as windows to the mind? A direct comparison between in-lab and web-based eye-tracking methods. *Open Mind*, *8*, 1369–1424. https://doi.org/10.1162/opmi_a_00171

Stone, K., Lago, S., & Schad, D. J. (2021). Divergence point analyses of visual world data: applications to bilingual research. *Bilingualism: Language and Cognition*, *24*(5), 833–841. https://doi.org/10.1017/S1366728920000607

Sun, C., & Breheny, R. (2020). Another look at the online processing of scalar inferences: an investigation of conflicting findings from visual-world eye-tracking studies. *Language, Cognition and Neuroscience*. https://www.tandfonline.com/doi/abs/10.1080/23273798.2019.1678759

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. http://www.ncbi.nlm.nih.gov/pubmed/7777863

Trueswell, J. C. (2008). *Using eye movements as a developmental measure within psycholinguistics* (I. A. Sekerina, E. M. Fernández, & H. Clahsen, Eds.; pp. 73–96). John Benjamins Publishing Company. https://doi.org/10.1075/lald.44.05tru

Viviani, P. (1990). Eye movements in visual search: cognitive, perceptual and motor control aspects. *Reviews of Oculomotor Research*, *4*, 353–393.

Voeten, C. C. (2023). *Permutes: Permutation tests for time series data*. https://CRAN.R-project.org/package=permutes

Vos, M., Minor, S., & Ramchand, G. C. (2022). Comparing infrared and webcam eye tracking in the Visual World Paradigm. *Glossa Psycholinguistics*, *1*(1). https://doi.org/10.5070/G6011131

Woods, K. J. P., Siegel, M. H., Traer, J., & McDermott, J. H. (2017). Headphone screening to facilitate web-based auditory experiments. *Attention, Perception, and Psychophysics*, *79*(7), 2064–2072. https://doi.org/10.3758/s13414-017-1361-2