eyetrackingR: An R Library for Eyetracking Data Analysis

Brock Ferguson (brock@u.northwestern.edu)

Northwestern University

Last updated: June 25, 2013

1. Introduction

Psychologists have measured eye gaze as a window into cognitive processes for more than a century (Buswell, 1922; Duchowski, 2007; Huey, 1908; Just & Carpenter, 1976), and their measurements have yielded tremendous evolutions in cognitive theory. Likewise, the tools employed to measure eye gaze have evolved: Many researchers now employ commercial eyetrackers from companies such as Tobii and SR Research (EyeLink) to record gaze direction. These systems have advantages over traditional methods such as frame-by-frame video coding. For example, they offer greater temporal and spatial resolution, with some systems sampling gaze direction up to 2000 times per second. However, given the amount of data exported by these systems, comprehensive data analyses can be problematic for psychologists who rely on traditional software to analyze their data (e.g., SPSS, JMP, Excel). Although these packages allow one to model their data, they do not allow for the types of pre-processing and data aggregation required for eyetracking analysis. Preparing, visualizing, and analysing these data require specialized functions that can handle the size of eyetracking datasets and the problems unique to their analysis.

In this paper, I introduce *eyetrackingR*, an open source library for R (R Development Core Team, 2012) that significantly reduces the time and technical knowledge required to prepare and analyze eyetracking datasets. Section 2 provides a high-level overview of the data requirements and functionality of the library. Section 3 introduces the supported data analysis techniques in detail. Section 4 walks through an example analysis of a 'looking while listening' experiment (e.g., Fernald, Zangl, Portillo, & Marchman, 2008). The library's specific R functions – discussed throughout the paper – are documented in detail in the Appendix.

2. Requirements and functionality

eyetrackingR includes a host of functions that help in the preparation and analysis of your dataset. In this section, we will broadly review what eyetrackingR requires and its basic functionality.

2.1. Data format requirements

eyetrackingR takes as its input a dataframe of eyetracking data similar to that exported by most commercial eyetrackers. This dataset must include columns for each participant's subject code, the Area of Interest (AOI) being looked at, whether or not the sample was lost to trackloss, the time (in milliseconds) from some relevant trial zero point (e.g., trial onset), the sample number, the trial (i.e., item) name, and a column for each AOI. This dataset might look like:

Table 1

Subset of an example dataset required for eyetrackingR

SubjID	Age	Trial	TimeMS	SampleNum	AOI	Target	Distractor	Trackloss
SUB1	24.33	Monkey	0	1	Trackloss	0	0	1
SUB1	24.33	Monkey	16.67	2	Trackloss	0	0	1
SUB1	24.33	Monkey	33.34	3	Target	1	0	0
SUB1	24.33	Monkey	50.01	4	Target	1	0	0
SUB1	24.33	Monkey	66.68	5	Target	1	0	0
SUB1	24.33	Monkey	83.35	6	Target	1	0	0
SUB1	24.33	Monkey	100.02	7	Distractor	0	1	0

SUB1	24.33	Monkey	116.69	8	Trackloss	0	0	1
SUB1	24.33	Monkey	133.36	9	Distractor	0	1	0

The dataset above also includes an additional column (Age) that is a factor that will be of interest in the analysis. Datasets can include any number of additional factors.

eyetrackingR does not require that these columns have specific names or positions within your dataset. Rather, the column names are specified prior to analysis (`set_data_options`) and saved as a parameter that is passed to future functions. Users can generate a random, fake dataset using these data format options (`generate_random_data`), for the sake of data file comparison or a mock analysis.

Assessing the current validity of one's dataset can be difficult given the amount of data in a typical eyetracking experiment dataset. eyetrackingR includes several functions that can assist with providing snapshots of your data. Users can take a quick summary about the amount of data (i.e., number of total samples, number of participants) (`summarize_dataset`), as well as display descriptive statistics for a particular dependent variable column aggregated by one or more factors (`describe_data`). The library can also generate smoothed (confidence ribbons) or empirical (error bars) plots of any dataset (`plot_data`)[1]. If the output of these functions is unexpected or incorrect, users can specifically assess the data format of each of the columns in

[1] The plotting code including in *eyetrackingR* is a minor adaptation of Judith Degen's spaghetti plot code published at `http://hlplab.wordpress.com/2012/02/27/creating-spaghetti-plots-of-eye-tracking-data-in-r/`. I adapted the code to aggregate across trials by participants (to appropriately estimate standard error) and to add some minor formatting considerations.

their dataset (`verify_dataset`). This function will report – and attempt to repair – any formatting inconsistencies in the dataset.

These functions can be run on a dataset at any time, and thus enable users to continuously monitor the state of their data.

2.2. Data preparation: Dealing with trackloss and subsetting your data

Eyetracking data often require some form of ‘subsetting’, or cleaning the data of data samples that are non-task related or lost to tracking errors. eyetrackingR includes several functions that can assist in this data preparation.

Datasets can be cleaned by particular factors (i.e., columns; `clean_by_factor`). This allows the user to, for example, return a dataset that includes only participants assigned to a single condition, or data from a particular time phase of the experiment if this phase is denoted by a unique value in a column. Datasets can also be reduced to gaze samples from within a given time period for each trial (`subset_by_window`). For example, in assessing infants’ knowledge of a given word, researchers tend to focus on a window from 267 to 2000 ms post word-onset (Swingley & Aslin, 2000), although this varies across experimental paradigms and semantic classes (Waxman, Lidz, Braun, & Lavin, 2009).

Trackloss and missing data must be considered prior to any eyetracking analysis. This is particularly true when conducting experiments with populations that generate high amounts of trackloss (e.g., infants). The user must make two important decisions: First, should non-task non-task directed looks (i.e., looks outside of all AOIs) count as trackloss (`treat_outside_looks_as_trackloss`)? Second, should trackloss be removed from (`remove_trackloss`) or kept in (`keep_trackloss`) the dataset prior to statistical analysis? Both of these questions bear on the denominator of any looking time proportion extracted from the data. For example, consider a simple experiment where an infant is shown two object images on a screen. Next, she hears one of the names of the objects. If non-AOI directed and trackloss samples are removed from the dataset, the experiment should predict that, if she knows the meaning of the word, she should look to the target object for more than 50% of the response period. However, if these samples are not removed from the dataset, the researchers prediction would be only that she should look to the target object more than the distractor.

The amount of trackloss for a given trial or participant can also determine whether a trial or participant should be included in the analysis. With eyetrackingR, researchers can assess the amount of trackloss for each trial and participant (`get_trackloss_data`), as well as tally, for each participant, in how many trials they contributed gaze samples (`final_trial_counts`). Moreover, trials can be eliminated from a dataset by establishing a maximum trackloss criterion and eliminating all trials in which that criterion is not met (`clean_by_trackloss`).

These functions allow the user to subset their raw eyetracking data into a 'clean' dataset that is ready for statistical analysis.

3. Data analysis

Although all eyetracking experiments measure the same behavioural response, their data can be analysed using several different techniques. Consider again a basic word knowledge task in which infants look at two objects on screen. After a couple of seconds, they hear a label referring to one of the objects. In this experiment, infants are assigned to one of two conditions: (1) 'Knowers', whose parents reported that they know the word and, (2) 'Non-knowers', whose parents reported that they did not know the word. The researcher can ask whether Knowers are more likely than Non-knowers to know the meaning of the word in many different ways and each of these formulations requires a different type of analysis. This section will briefly review these various analyses and introduce eyetrackingR's functions that take your prepared data and return a data set ready for analysis.

3.1. Window analysis

The most basic question a researcher might ask – and the one asked of most two alternative forced-choice looking designs (Bergelson & Swingley, 2012; Golinkoff & Hirsh-Pasek, 2008; Shukla, White, & Aslin, 2011; Waxman et al., 2009; Werker, Cohen, Lloyd, Casasola, & Stager, 1998) – is whether participants in one condition looked more to one scene than another in a certain time window. For example, did the Knowers look more to the target than the Non-knowers in the time after they heard the word? These total looking proportions are often aggregated across subjects (collapsing across trials). However, these subjects-only analyses are less powerful than an analysis that aggregates by trials and subjects, thus allowing for estimation of both random subject and item variance in a mixed-effects model (Baayen, Davidson, & Bates, 2008; Barr, 2008; Jaeger, 2008). For this reason, eyetrackingR includes a function (`window_analysis`) that returns looking proportions to a given AOI aggregated by trials and subjects (i.e., ready for a mixed-effects analysis). The dependent variable (i.e., looking proportion to an AOI) is calculated in its raw form as well as transformed with an empirical logit transformation and arcsine square root transformation to compensate for the boundedness of the proportional (i.e., logistic) scale (Barr, 2008; Jaeger, 2008). This dataset can also include any number of additional participant or trial factors.

3.2. Sequential bins analysis

Notably, the window analysis described above fails to utilize one of the key advances of modern eyetracking system, namely their high temporal resolution. Or, to be more specific, window analyses ignore time altogether except to zoom in on a particular window of interest. This can come at the cost of losing vital clues to the nature of cognitive processing, and therefore recent studies have sought to examine change over time in their analyses (Altmann & Kamide, 1999; Fernald et al., 2008; Swingley & Aslin, 2000). A sequential bins analysis is one way to look for change over time. For example, in the example experiment introduced above, the experimenter might want to determine in what time bin the Knowers first looked more than the Non-knowers to the target image. This would inform the timecourse of processing a known word. With the present library, experimenters can define a bin size (e.g., 250ms) and look for a main effect of a given factor predicting looking towards an arcsine root-transformed dependent variable AOI within each of these bins (`sequential_bins_analysis`).

3.3. Linear and non-linear (growth curve) timecourse analyses

Another technique for looking at change over time is to estimate the influence of one or more factors on fitting participants' data to linear or non-linear polynomial time factors (Barr, 2008; Mirman, Dixon, & Magnuson, 2008). To demonstrate, I will add a new twist to our word knowledge experiment: In one trial, infants saw an image of Elmo alongside an image of an egg while they heard the word *egg*, which should have prompted them to fixate on the egg. Of course, young children love Elmo, and chose to look at him much more than the egg. If one analysed these data using only a window analysis, it might look like both Knowers and Non-knowers did not know the word *egg* despite parents' reporting that Knowers knew this word – both groups appear to think *egg* means Elmo.

A linear time analysis can salvage this trial. The researcher can fit a model with three parameters: time (linear: 0ms, 250ms, 500ms, …), condition (Knowers, Non-knowers) and their interaction (time by condition) predicting looking at the Elmo AOI. What he might find is: (1) in general, all infants' likelihood of looking at Elmo increased over time (significant positive slope coefficient for time), (2) Knowers and Non-knowers did not significantly differ in their total looking to Elmo (insignificant slope coefficient for condition), but, crucially, (3) Knowers were slower in increasing their looking at Elmo over time (i.e., they took longer to disengage from the egg; significant negative slope coefficient for the time by condition interaction term). This would suggest that Knowers did in fact know the meaning of *egg* and this influenced the rate of change over time for their preference of Elmo.

In the same spirit as a linear time analysis, experimenters can include higher order polynomial time factors to look for non-linear (e.g., quadratic, cubic, quartic, …) looking patterns in their data, and how one or more factors influence the estimated coefficients for these polynomial terms (Kukona, Fang, Aicher, & Chen, 2011; Mirman et al., 2008). These higher order polynomial codes are like linear time codes except that they demonstrate non-linear rates of change. For example, the experimenter in the present experiment might predict that Non-knowers' looking patterns should be cubic, i.e., that these infants might oscillate between the two images on the screen while Knowers fixate on the target throughout the response period. By fitting a model with a linear time factor, a quadratic time factor, and a cubic time factor, the experimenter can look for significant differences between conditions in the estimated coefficient for the cubic polynomial. Knowers might not fit at all (a coefficient of 0) while Non-knowers' looking can be predicted from the cubic polynomial (a non-zero coefficient). (In reality, this would be a tough prediction to make if Non-knowers' looking at trial onset is random. In this case, half of the infants would have a positive cubic polynomial coefficient while the other half would have a negative coefficient, leading to an estimated coefficient of 0 for the condition as a whole.)

eyetrackingR includes a single function (`time_analysis`) that yields a dataset ready for linear and non-linear time analysis. Proportions of time looking to an AOI are binned into time bins, aggregated by trials and participants, and included a dataset alongside additional columns that represent potential linear and non-linear time factors. Linear factors are calculated from the timestamps of each sample. Non-linear polynomial codes are orthogonal codes that can be simultaneously entered into a mixed-effect models to calculate complementary estimates of coefficients for various non-linear forms. These estimates allow for nuanced predictions about the structure of looking over time. As with the window analysis, the dependent variable is returned in its raw proportional form as well as empirical logit-transformed and arcsine root-transformed.

3.4. Ranked looks analysis

Window analyses, sequential bins analyses, and time analyses all consider participants' looking in a way that disregards the nature of individual looks or fixations during a trial. However, properties of participants' looking such as their longest look to an AOI can sometimes yield more robust evidence of, for example, word learning than estimates of total looking time (Schafer & Plunkett, 1998). Considering the example experiment, the experimenter might predict that Knowers would show on average longer looks towards the target than the distractor image, or longer looks to the target than Non-knowers looks to the target. This analysis requires that looks be collapsed across individual fixations (i.e., consistent looking towards an AOI) and then ranked from the longest to shortest look. eyetrackerR includes a function for returning such a dataset (`get_looks`) with looks sorted by trials and participants. The AOI of interest of each look, its length in data samples and time, rank (by length), and rank sequence (by time) are returned.

3.5. Looking switch analysis

A final analysis that has been effectively used in two-alternative forced choice looking experiments is what the present library calls a looking switch analysis (Fernald, Thorpe, & Marchman, 2010). These analyses compare two subsets of trials – (1) trials in which the participant was looking on the distractor image at trial- or window-onset, and (2) trials in which the participant was looking at the target at the same timepoint – and compares the relative speed with which these infants 'switched' looking from AOI (i.e., disengaged from the AOI to fixate on another AOI). In the present example, the prediction would be that Knowers would be faster to switch when they were looking at the distractor image than when they were looking at the target image. Moreover, they would be faster to switch from the distractor image than the Non-knowers. In service of this type of analysis, the present library can return a dataset that includes, for each trial and participant, the AOI they first looked at and for how long, the AOI they looked at second and for how long, and the 'switch time' between these two looks (`first_looks_analysis`).

4. Example analysis: A 'looking while listening' experiment

To understand how eyetrackerR can assist in a statistical analysis, I will review an example analysis of the 'looking while listening' experiment (Fernald et al., 2008) used in the previous sections. To recap, in this experiment, children look at images of two objects on a screen (e.g., a monkey and a car) while they hear a label for one of the objects depicted. The experimenter is interested in whether 'Knowers', whose parents reported that they knew the word, do in fact know the referent of the word more than 'Non-knowers', whose parents reported that they did not know the word. It uses a within-subject design that has 6 total trials, half of which are trials in which the child is a 'Knower' and the other half in which the child is a 'Non-knower' (i.e., each child conveniently knew 3 of the 6 words). Each trial has two phases: the 'Baseline' phase and 'Response' phase (each 4s in duration). If parents can accurately assess their young child's word knowledge, Knowers should look more towards the target than Non-knowers in the Response phase. Our analysis will particularly focus on a window 367ms to 1500ms during the Response phase (i.e., from 4367ms to 5500ms in each trial).

4.1. Download eyetrackingR and its dependencies

First, let us begin by downloading eyetrackingR, available from `https://www.github.com/brockf/eyetrackingR`. Second, because the `plot_data()` and `describe_data()` functions have external dependencies, we will make sure that the 'psych', 'ggplot2', 'ggthemes', and 'reshape2' R packages are installed. If they are already installed, no action is required; eyetrackingR will load them quietly when these functions are called.

4.2. Create analysis folder and load eyetrackerR

Create a new folder and place the `analyze-eyetracker.R` source file into it. Then, set this folder as your current working directory, and source eyetrackerR:

```
> source('analyze-eyetracker.R')
```

4.3. Set data configuration options and generate data

These options will typically be set using the column names and parameters of the eyetracker being used. However, in this case, we can define these options to any reasonable value. They will govern the properties of the dataset generated and be passed to most future eyetrackerR functions.

```
> data_options <- set_data_options(sample_rate = 60, participant_factor =
'ParticipantName', active_aoi_factor = 'SceneType', trackloss_factor =
'TrackLoss', time_factor = 'TimeFromTrialOnset', sample_factor =
'FramesFromTrialOnset', trial_factor = 'Trial', default_dv = 'Target',
default_factors = c('Condition','Age'))
```

These data options are compatible with the data format illustrated in Table 1, as generated by a 60hz eyetracker.

We can now generate a random dataset that matches these data options. In doing so, we will pass a `seed` argument that governs the random number generation. Using the seed number below will generate a dataset exactly like the one used in preparing this walkthrough. Using a different seed number will generate another random dataset that can act as a conceptual – but not statistical – equivalent.

```
> data <- generate_random_data(data_options, seed = 42, num_participants = 40)
```

The 'raw' eyetracking data is now stored in a `data` dataframe. We can verify that the data in each column is in an acceptable format (e.g., numeric where it should numeric) by verifying the dataset:

```
> data <- verify_dataset(data, data_options)
```

This should not produce any errors but, instead, display the columns being verified:

```
Participant factor: ParticipantName
Time factor: TimeFromTrialOnset
Sample factor: FramesFromTrialOnset
Trial factor: Trial
```

If there were errors, they would be reported. But this is not the case and so we can move on.

We can take our first look at the dataset by describing the values of our dependent variable AOI (the 'Target' image) by the condition factor:

```
> describe_data(data, dv = 'Target', factors = c('Condition'))
```

```
DV: Target
========================================================================
X1: Knower
[Mean]: 0.583   [SD]:   0.493   [Var]:  0.243   [Min]:  0   [Max]:   1
------------------------------------------------------------------------X1:
NonKnower
[Mean]: 0.528   [SD]:   0.499   [Var]:  0.249   [Min]:  0   [Max]:   1
```

These statistics are encouraging – Knowers looked to the target 58.3% of the time during the trials while NonKnowers looked to the target only 52.8% of the time. However, we need to a bit of data preparation before we can do the sorts of statistical analyses that can tell us if this is a reliable difference.

4.4. Clean by factor

Recall that each trial has two phases: Baseline and Response. The baseline phase was meant only to familiarize children with the objects in the trial, and we make no predictions about Knowers' and Non-knowers' looking during this phase. We can therefore exclude this data from the dataset:

```
> data <- clean_by_factor(data, data_options, factor = 'Window', allowed_levels
= c('Response'))
```

Our `data` now includes only samples where the 'Window' column is equal to 'Response'. If we had other phases that we wanted to include, we could have appended them to the `allowed_levels` character vector.

4.5. Clean by trackloss

As the experiment was running, we recognized that there was a significant amount of trackloss occurring during some sessions and trials. This is not surprising given that we are studying children – who like to shift and look away from the screen during the experiment. However, we do not want to include this data in our final analysis and thus have decided to eliminate data samples lost to trackloss from the dataset as part of our data preparation.

First, we must assess the amount of trackloss in the data. We can analyze the samples lost trackloss within our window of interest[2]:

```
> trackloss <- get_trackloss_data(data, data_options, window_start = 4367,
window_end = 5500)
```

This returns a `list` (`trackloss`, in this case) that has two elements: `trackloss`, a trial-by-trial analysis of trackloss, as a dataframe, and `trials_committed`, a dataframe showing the number of trials that had at least 1 non-trackloss sample for each participant:

```
> head(trackloss$trackloss)
ParticipantName   Trial      SamplesLooking TotalSamples TracklossSamples TracklossProportion
1 SUBJ_1          Basketball 58                      68              10          0.14705882
2 SUBJ_10         Basketball 62                      68               6          0.08823529
3 SUBJ_11         Basketball 52                      68              16          0.23529412
4 SUBJ_12         Basketball 53                      68              15          0.22058824
5 SUBJ_13         Basketball 57                      68              11          0.16176471
6 SUBJ_14         Basketball 46                      68              22          0.32352941

> head(trackloss$trials_committed)
ParticipantName          Trials
1         SUBJ_1         6
2         SUBJ_10        6
3         SUBJ_11        6
4         SUBJ_12        6
```

---

[2] Normally, we would first have to decide whether non-AOI looks should be treated like trackloss and excluded from the analysis. If this is the case, we would run `data <- treat_outside_looks_as_trackloss(data, data_options)` before analyzing trackloss.

```
5         SUBJ_13           6
6         SUBJ_14           6
```

Some basic statistics can reveal the nature of the trackloss:

```
> mean(trackloss$trackloss$TracklossProportion)
[1] 0.214277

> sd(trackloss$trackloss$TracklossProportion)
[1] 0.08270246

> mean(trackloss$trials_committed$Trials)
[1] 6

> sd(trackloss$trials_committed$Trials)
[1] 0
```

Every participant contributed 6 trials (not unexpected, considering they were random number generating robots). Across all trials, we averaged 21.4% trackloss (*SD* = 8.27%).

We would like to eliminate all trials that had trackloss greater than 2 standard deviations above the mean. We do this by calculating the minimum number of samples required in order to keep a trial as the mean number of good samples – 2*SD(number of good samples):

```
> minimum_samples <- mean(trackloss$trackloss$SamplesLooking) -
2*sd(trackloss$trackloss$SamplesLooking)

> minimum_samples
[1] 42.18163
```

All trials that contribute fewer than ~42.18 (i.e., 43, rounded up) good samples of data within the 4367ms to 5500ms time window can now be excluded:

```
> data <- clean_by_trackloss(data, data_options, window_start = 4367,
window_end = 5500, minimum_samples = minimum_samples)
Trimming data outside of window...
Calculating trials to be dropped...
Dropping 9 trials...
```

Nine trials were dropped from the dataset. We can confirm this by looking at the final trial counts for trials in which the participant contributed data:

```
> trial_counts <- final_trial_counts(data, data_options, window_start = 4367,
window_end = 5500)
Trimming data outside of window...

> head(trial_counts)
ParticipantName      Trials
1 SUBJ_1             6
2 SUBJ_10            6
3 SUBJ_11            6
4 SUBJ_12            6
5 SUBJ_13            6
6 SUBJ_14            5

> mean(trial_counts$Trials)
[1] 5.775

> sd(trial_counts$Trials)
[1] 0.5767949
```

With one trial dropped, we average 5.78 trials per participant (*SD* = .57).

Given that we now have the data we need to report regarding trackloss, and we have no more use for samples lost to trackloss, we can eliminate trackloss from the dataset entirely:

```
> data <- remove_trackloss(data, data_options)
```

This results in an increase in looking to the Target AOI across both conditions, as illustrated in this updated breakdown of looking to the target AOI:

```
> describe_data(data, dv = 'Target', factors = c('Condition'))
DV: Target
========================================================================
X1: Knower
[Mean]: 0.663   [SD]:   0.473   [Var]:  0.224   [Min]:  0    [Max]:  1

------------------------------------------------------------------------
X1: NonKnower
[Mean]: 0.557   [SD]:   0.497   [Var]:  0.247   [Min]:  0   [Max]:  1
```

The reason for this increase should be clear: By eliminating samples lost to trackloss, we have reduced the denominator used to calculate the proportion of looking towards the Target.

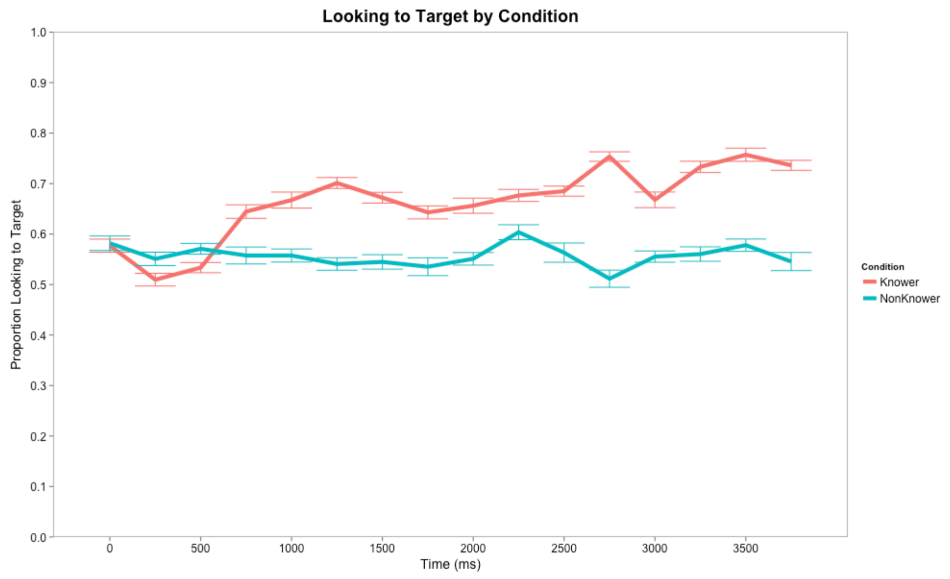
4.6. Plot data

The descriptive statistics above reveal little about the patterns of looking within the Knower and Non-knower conditions. A careful look at the proportion of looking towards the Target AOI over time can answer some interesting questions about the infants' responses in this task. For example, did the Non-knowers ever orient towards the target? How quickly did the Knowers recognize the target? Did Knowers sustain their attention on the target, or did they oscillate between the target and distractor? Answers to these questions will become more clear when we visualize our data.

First, let us plot proportion looking towards the Target AOI over time for the entire `data` dataframe (i.e., the entire timecourse of the Response phase):

```
> plot_data(data, data_options, output_file = 'graph-entire-trial.png', dv =
'Target', factor = 'Condition', title = 'Looking to Target by Condition',
y_title = 'Proportion Looking to Target', x_title = 'Time (ms)', bin_time =
250)
```

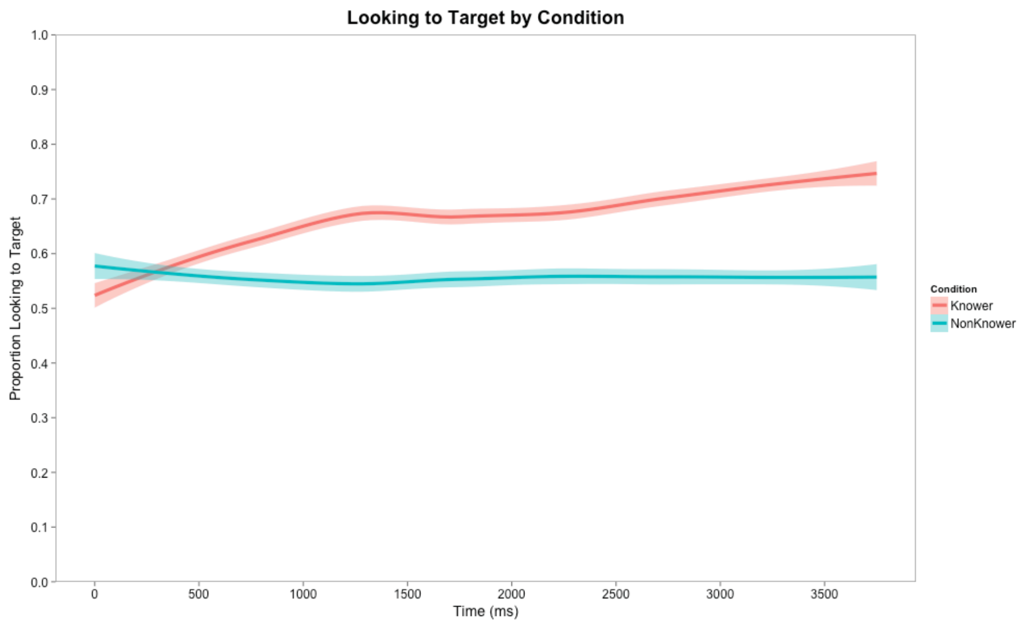
This command generates a plot with two lines (for Knowers and Non-knowers, respectively) and saves it as `graph-entire-trial.png` in the working directory.



Data samples are binned by 250ms time windows – a relatively large bin size, but appropriate for infant studies that have lots of trackloss and periods of inattention. Data in these bins is collapsed to the minimum time value within the bin (e.g., the 0 timepoint includes data from 0ms-249ms; the 250ms timepoint includes data from 250ms to 499ms). The plots, by default, are 'empirical' blots with means and error bars (aggregated across trials by participants) connected by lines.

We can also create the same plot with 'smoothed' spaghetti plots by specifying the `type` argument in `plot_data`. These plots smooth the empirical data (using *loess*; locally weighted scatterplot smoothing) and are accompanied by 95% confidence interval ribbons:

```
> plot_data(data, data_options, output_file = 'graph-entire-trial-
smoothed.png', dv = 'Target', factor = 'Condition', title = 'Looking to Target
by Condition', y_title = 'Proportion Looking to Target', x_title = 'Time (ms)',
bin_time = 250, type = 'smoothed')
```

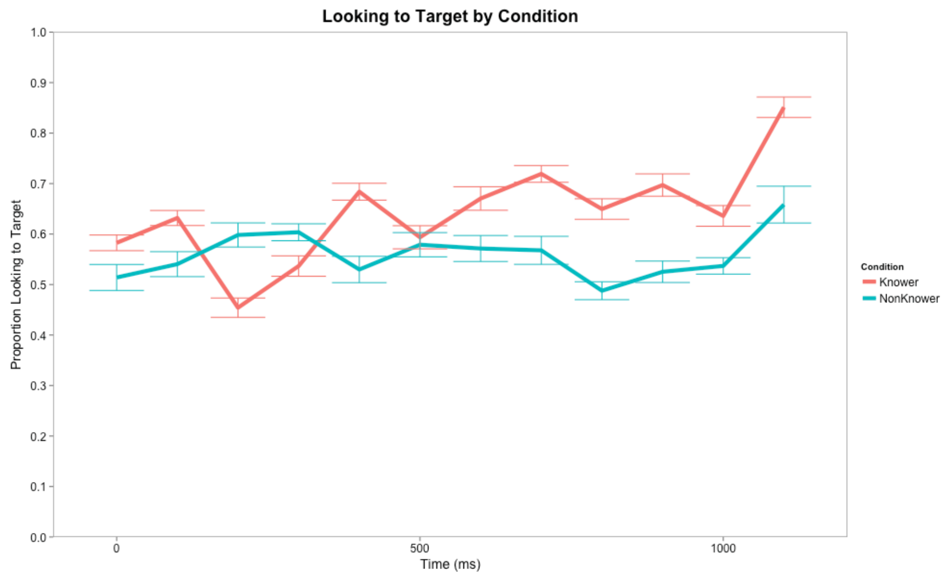


These smoothed plots are an abstraction from the empirical data but can be beneficial in visualization.

Recall that our window of interest is from 367ms to 1500ms during the Response phase. We can plot only this window by first subsetting the data and then passing this subsetted `dataframe` to `plot_data`, like above:

```
> data_window <- subset_by_window(data, data_options, window_start = 4367,
window_end = 5500)

> plot_data(data_window, data_options, output_file = 'graph-window.png', dv =
'Target', factor = 'Condition', title = 'Looking to Target by Condition',
y_title = 'Proportion Looking to Target', x_title = 'Time (ms)', bin_time =
250)
```



From these visualizations, it looks as if the Knowers were more likely to look to the Target than the Non-knowers (comparison across conditions), were more likely to look to the Target than the Distractor (comparison to chance, 50%), responded after approximately 500ms to the known word, and sustained their attention to the Target throughout the remainder of the response period. However, before we can be confident that these observations will generalize to new data, we must assess the reliability of these results with statistical analyses.

4.7. Analyses

For the sake of demonstration, we will perform 4 analyses on these data. In an actual analysis, it would be best to motivate and select the analyses that best answer your questions.

4.7.1. Window analysis

In order to assess whether Knowers looked more to the Target than Non-knowers, we will perform a window analysis. A window analysis collapses across time within a given window (or the entire dataset) and yields observations by trial and participant (and any other factors specified, e.g., Condition), as is appropriate for a mixed-effects analysis.

Let us create our windowed dataset from our raw data:

```
> window <- window_analysis(data, data_options, dv = 'Target', factors =
c('Condition'))
```

This yields a dataset with 231 observations (40 participants with 6 trials each, minus 9 trials that were dropped due to trackloss). We can fit a model predicting raw proportions, empirical logit-transformed values, or arcsine-root transformed proportions. We will use empirical logit transformed values and fit a mixed-effects model with the `lme4` package and pass the `wts` column to do a weighted linear regression. This tells the model fitting procedure to prioritize minimizing error on observations that have the most samples, which can be important in datasets with significant trackloss.

```
> library(lme4)

> window_model <- lmer(elog ~ Condition + (1 | ParticipantName) + (1 | Trial),
data = window, weights = 1/wts)

> summary(window_model)

Linear mixed model fit by REML
Formula: elog ~ Condition + (1 | ParticipantName) + (1 | Trial)
   Data: window
AIC   BIC logLik deviance REMLdev
674 691.2   -332      651     664

Random effects:
 Groups          Name        Variance   Std.Dev.
ParticipantName (Intercept) 1.1706e-05 0.0034214
Trial           (Intercept) 2.7309e-06 0.0016525
Residual                    2.1601e-02 0.1469733
Number of obs: 231, groups: ParticipantName, 40; Trial, 6

Fixed effects:
                    Estimate Std. Error t value
(Intercept)         0.668177   0.002311   289.2
ConditionNonKnower -0.442323   0.003098  -142.8
```

```
Correlation of Fixed Effects:
            (Intr)
CndtnNnKnwr -0.682
```

Non-knowers were significantly less likely to orient towards the Target than Knowers (t = -142.8). To attain a p-value for Condition as a main effect, we will compare this model with Condition as a factor to a null model without this factor of interest:

```
> window_model_null <- lmer(elog ~ 1 + (1 | ParticipantName) + (1 | Trial),
data = window, weights = 1/wts)

> anova(window_model, window_model_null)
Data: window
Models:
window_model_null: elog ~ 1 + (1 | ParticipantName) + (1 | Trial)
window_model: elog ~ Condition + (1 | ParticipantName) + (1 | Trial)
                  Df    AIC    BIC  logLik  Chisq Chi Df Pr(>Chisq)
window_model_null  4 761.89 775.66 -376.94
window_model       5 660.96 678.17 -325.48 102.93      1  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The main effect of Condition can be reported as, β = -.44, $\chi^2(1)$ = 102.93, p < .00001. Beta is estimated as part of the mixed-effects model fitting (`lmer`) and the p-value is established with an ANOVA comparison of the two models using a -2 log-likelihood ratio test.

This window analysis would likely be enough to conclude that Knowers were more likely than Non-knowers to recognize the word, but we will continue on with other analyses to look for corroborating evidence.

4.7.2. Sequential bins analysis

Given that Knowers ultimately looked to the target more than Non-knowers, we can ask a subsequent question: When did looking to the target between these two conditions begin to diverge? This can begin to address questions regarding the efficiency of word recognition, i.e., the response time. To answer this question, we will compare sequential time bins and look for the first time bin in which these groups reliably diverge:

```
> bin_test <- sequential_bins_analysis(data, data_options, bin_time = 100, dv =
'Target', factor = 'Condition')

> bin_test
Bin StartTime EndTime Participants SameDifferent p-value df_num df_den       F
0           0     100           40          same    0.35      1     38   0.892
1         100     200           40     different   0.001      1     38  15.492
2         200     300           40     different   0.001      1     38  24.172
3         300     400           40          same   0.209      1     38   1.630
4         400     500           40          same   0.588      1     38   0.300
5         500     600           40     different    0.04      1     38   4.529
…
```

The output of the bins analysis (`bin_test`) is truncated here, but hopefully the format is clear. Within each bin, a simple one-way ANOVA is performed looking for a main effect of the `factor`. The results of this test are reported and we can see whether the levels of the factor significantly differ or not within that bin. Researchers have a few degrees of freedom here in the way they interpret these results. For example, the researcher must decide how many 'different' sequential bins represent a meaningful divergence between the conditions. This would be a function of the task, the time within each bin, and the predicted effect size.

4.7.3. Linear time analysis

The sequential bins analysis above gave us some idea about how looking in our Knowers and Non-knowers conditions changed over time. However, by applying a linear model in a linear time analysis, we can estimate the differences between our groups vis-à-vis the intercept (i.e., at timepoint zero) and slope (i.e., change in looking towards the AOI as a function of time).

First, we need to prepare a dataset that is appropriate for a time analysis. This will aggregate our data into bins, generate several new factor columns (one of which is a linear time factor used here), and transform our dependent variable into various analysis-ready variables (we will use the empirical logit transformation):

```
> time_data <- time_analysis(data, data_options, bin_time = 250, dv = 'Target',
factors = c('Condition'))
```

We have the option of fitting a linear model by subjects (`time_data$bySubj`), by items (`time_data$byItem`), or with crossed random effects (`time_data$crossed`). Here we will use the crossed dataset to estimate item and subjects random effects simultaneously, predicting the empirical logit transformed dependent variable from the linear time factor (`time_data$crossed$t1`) and Condition factor (`time_data$crossed$Condition`):

```
> linear_model <- lmer(elog ~ t1*Condition + (1 | ParticipantName) + (1 |
Trial), data = time_data$crossed, weights = 1/wts)

> summary(linear_model)
Linear mixed model fit by REML
Formula: elog ~ t1 * Condition + (1 | ParticipantName) + (1 | Trial)
Data: time_data$crossed

 AIC  BIC logLik deviance REMLdev
 9777 9821  -4882     9724    9763

Random effects:
Groups          Name        Variance   Std.Dev.
ParticipantName (Intercept) 0.0000e+00 0.0000000
Trial           (Intercept) 4.3517e-05 0.0065967
Residual                    2.8891e-01 0.5375057
Number of obs: 3696, groups: ParticipantName, 40; Trial, 6

Fixed effects:
                       Estimate Std. Error t value
(Intercept)            0.2275565  0.0142382  15.982
t1                     0.0116257  0.0004080  28.491
ConditionNonKnower    -0.0183465  0.0198405  -0.925
t1:ConditionNonKnower -0.0121506  0.0005722 -21.234
```

In a real analysis, we would use fit a model without each of these terms and compare models using `anova()` to estimate p-values, like we did for the window analysis. However, in this case, we will interpret these effects using the reported t-values. We can consider these effects in terms of the differences between our Knowers and Non-knowers and their looking towards the target object. First, the significant main effect of the `t1` variable means that there was a general increase of looking towards the Target over time across both conditions. Second, there is no main effect of Condition. This suggests there was no difference when `t1 == 0`, i.e., at trial onset. Of course, this makes sense in our study. A significant difference between conditions here would suggest that Knowers were anticipating the referent target before even hearing the word. Third, we see a significant interaction between `t1` and Condition, i.e., the change over time differed between our conditions: The negative estimate here reveals that Non-knowers were significantly less likely than Knowers in orienting towards the Target over time.

4.7.4. Growth curve analysis

In the linear time analysis above, we assume that the change over time will be linear, i.e., the differences between conditions will be reflected by a consistent increase or decrease in looking towards an AOI over time. Growth curve analyses substitute the linear time factor (t1) in the time_data datasets produced above by time_analysis() with multiple orthogonal polynomial timecodes. This allows us to estimate the non-linear shape of change over time, and the way this shape may differ between our conditions (or any other factors).

We begin with the `time_data` dataframe generated above, but fit a model with polynomial codes:

```
> growth_model <- lmer(elog ~ ot1 + ot2 + ot3 + Condition + ot1:Condition +
ot2:Condition + ot3:Condition + (1 | Trial) + (1 | ParticipantName), data =
time_data$crossed, weights = 1/wts)

> summary(growth_model)
Linear mixed model fit by REML
Formula: elog ~ ot1 + ot2 + ot3 + Condition + ot1:Condition + ot2:Condition +
ot3:Condition + (1 | Trial) + (1 | ParticipantName)
```

```
Data: time_data$crossed
 AIC  BIC logLik deviance REMLdev
 9765 9833  -4872     9704    9743

Random effects:
 Groups          Name        Variance   Std.Dev.
 ParticipantName (Intercept) 0.0000e+00 0.0000000
 Trial           (Intercept) 4.4763e-05 0.0066906
 Residual                    2.8769e-01 0.5363666

Number of obs: 3696, groups: ParticipantName, 40; Trial, 6

Fixed effects:
                        Estimate   Std. Error t value
(Intercept)              0.576648   0.008007   72.02
ot1                      0.847327   0.030088   28.16
ot2                     -0.161802   0.030110   -5.37
ot3                      0.123111   0.030174    4.08
ConditionNonKnower      -0.383013   0.010528  -36.38
ot1:ConditionNonKnower -0.886814   0.042156  -21.04
ot2:ConditionNonKnower  0.201758   0.042197    4.78
ot3:ConditionNonKnower -0.196866   0.042231   -4.66
```

The significant main effects of ot1, ot2, and ot3 suggest that our data (across both conditions) can – with some reliability – be modelled by linear, quadratic, and cubic functions, although the linear fit is clearly superior. The linear fit, as in the linear time analysis, suggests an increase in looking towards the Target over time. The quadratic fit has a negative estimate and, because the quadratic timecode represents a U-shaped curve, we can deduce that the quadratic fit here approximates an inverted U-shape.

The main effect of Condition is interpreted differently than in the linear time analysis because orthogonal polynomial codes' zero points do not represent the start of the trial, they represent the trial mean. Here this effect is significant because, as we have seen in other analyses, Knowers looked significantly more towards the Target than Non-knowers

Finally, the interactions between our polynomial timecodes and Condition suggest differential fits to linear, quadratic, and cubic functions between our two conditions.

4.7.5. Looking switch analysis

Here we want to compare the speed with which participants who are looking at the *wrong* object at trial onset, orient towards the Target object. We predict that this should be different between our two conditions: Knowers (who recognize the word and search for its referent) should be faster to correct their looking than Non-knowers (who will eventually look towards the Target, but only due to random chance or undirected oscillation between objects). Conversely, Knowers should be slower to switch away from the Target towards the Distractor than Non-knowers.

First, we will generate a new dataset that breaks down our gaze data into individual 'looks.' This collapses across sequential fixations towards the same AOI.

```
> first_looks <- first_looks_analysis(data, data_options, factors =
c('Condition'))

> head(first_looks)
```

```
P..Name Condition Trial      FirstAOI   FirstStartTime  SecondAOI   SecondStartTime SecondEndTime SwitchTime
SUBJ_1  Knower    Basketball Target     0               Distractor  16.67           33.33         16.67
SUBJ_1  Knower    Bowl       Target     0               Distractor  16.67           100           16.67
SUBJ_1  Knower    Chimpanzee Distractor 0               Target      33.33           66.67         33.33
SUBJ_1  Knower    Pen        Distractor 0               Target      33.33           66.67         33.33
SUBJ_1  Knower    Speaker    Distractor 0               Target      16.67           50            16.67
SUBJ_1  Knower    Woman      Target     0               Distractor  33.33           50            33.33
```

For example, in the first row above, we can see that subject #1 (who is a Knower of the word 'basketball') looked towards the Target before the Distractor in this trial. His first look towards the Target was at trial onset (0ms), but he switched to the Distractor AOI at 16.67ms. This creates a 'switch' time of 16.67ms.

For this analysis, we will be doing simple t-tests. So, to avoid inflating our degrees of freedom and statistical power, we must aggregate across subjects:

```
> first_looks_agg <- aggregate(first_looks$SwitchTime, by =
list(first_looks$ParticipantName, first_looks$Condition, first_looks$FirstAOI),
FUN = mean)

> colnames(first_looks_agg) <-
c('ParticipantName','Condition','FirstAOI','SwitchTime')
```

We now have an aggregated dataset with only the columns we require to test our predictions.

Let's test our prediction that Knowers should be faster to switch away from the Distractor than Non-knowers in trials where they happened to make their first look towards the Distractor:

```
> t.test(first_looks_agg[which(first_looks_agg$Condition == 'Knower' &
first_looks_agg$FirstAOI == 'Distractor'), 'SwitchTime'],
first_looks_agg[which(first_looks_agg$Condition == 'NonKnower' &
first_looks_agg$FirstAOI == 'Distractor'), 'SwitchTime'], var.equal = T)

  Two Sample t-test
data:  first_looks_agg[which(first_looks_agg$Condition == "Knower" &  and
first_looks_agg[which(first_looks_agg$Condition == "NonKnower" &
first_looks_agg$FirstAOI == "Distractor"), "SwitchTime"] and
first_looks_agg$FirstAOI == "Distractor"), "SwitchTime"]

t = 2.5386, df = 36, p-value = 0.0156

alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  2.369112 21.192617
sample estimates:
mean of x mean of y
37.80864  26.02778
```

Here we see a significant pattern that is actually the opposite of our predicted pattern. Non-knowers are slightly faster (M = 26.03ms) to orient away from the Distractor than Knowers (M = 37.81), $p < .02$.

Now let’s test the prediction that Knowers should be slower to orient away from the Target in trials in which they happen to make the first look towards the Target:

```
> t.test(first_looks_agg[which(first_looks_agg$Condition == 'Knower' &
first_looks_agg$FirstAOI == 'Target'), 'SwitchTime'],
first_looks_agg[which(first_looks_agg$Condition == 'NonKnower' &
first_looks_agg$FirstAOI == 'Target'), 'SwitchTime'], var.equal = T)


  Two Sample t-test
data:  first_looks_agg[which(first_looks_agg$Condition == "Knower" &  and
first_looks_agg[which(first_looks_agg$Condition == "NonKnower" &
first_looks_agg$FirstAOI == "Target"), "SwitchTime"] and
first_looks_agg$FirstAOI == "Target"), "SwitchTime"]


t = -0.4739, df = 38, p-value = 0.6383


alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -11.862237   7.362237
sample estimates:
mean of x mean of y
 38.19444  40.44444
```

Now we see a non-significant trend that again opposes our predicted pattern. Knowers are slightly faster to orient away from the Target (*M* = 38.19ms) than Non-knowers (*M* = 40.44ms), *p* = .64.

Why might we be finding these strange results? In this case, it is likely due to the nature of the random data generator. Instead of producing typical fixation data with long looks towards AOIs, the random data generator has thousands of fixations per trial. This results in a number of ‘switches’ between AOIs and here results in spurious correlations between these switches and condition.

5. Conclusion

The eyetrackingR library discussed and demonstrated in this paper allows for comprehensive and multifaceted analyses of eyetracking data with only minimal knowledge of R.

References

Altmann, G. T. M., & Kamide, Y. (1999). Incremental interpretation at verbs: restricting the domain of subsequent reference. *Cognition*, *73*(3), 247–264. doi:10.1016/S0010-0277(99)00059-1

Baayen, R. H., Davidson, D. J., & Bates, D. M. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, *59*(4), 390–412. doi:10.1016/j.jml.2007.12.005

Barr, D. J. (2008). Analyzing “visual world” eyetracking data using multilevel logistic regression. *Journal of Memory and Language*, *59*, 457–474.

Bergelson, E., & Swingley, D. (2012). At 6-9 months, human infants know the meanings of many common nouns. *Proceedings of the National Academy of Sciences of the United States of America*, *109*(9), 3253–3258. doi:10.1073/pnas.1113380109

Buswell, G. T. (1922). *Fundamental reading habits: a study of their development*.

Duchowski, A. T. (2007). Eye tracking methodology: Theory and practice.

Fernald, A., Thorpe, K., & Marchman, V. A. (2010). Blue car, red car: Developing efficiency in online interpretation of adjective–noun phrases. *Cognitive Psychology*, *60*(3), 190–217. doi:10.1016/j.cogpsych.2009.12.002

Fernald, A., Zangl, R., Portillo, A. L., & Marchman, V. A. (2008). Looking while listening: Using eye movements to monitor spoken language comprehension by infants and young children. In I. A. Sekerina, E. M. Fernández, & H. Clahsen (Eds.), *Developmental Psycholinguistics: On-line methods in children's language processing* (pp. 97–135).

Amsterdam: John Benjamins.

Golinkoff, R. M., & Hirsh-Pasek, K. (2008). How toddlers begin to learn verbs. *Trends in Cognitive Sciences*.

Huey, E. B. (1908). *The Psychology and Pedagogy of Reading*.

Jaeger, T. F. (2008). Categorical data analysis: Away from ANOVAs (transformation or not) and towards logit mixed models. *Journal of Memory and Language*, *59*(434-446). doi:10.1016/j.jml.2007.11.007

Just, M. A., & Carpenter, P. A. (1976). *The Role of Eye-fixation Research in Cognitive Psychology*.

Kukona, A., Fang, S., Aicher, K., & Chen, H. (2011). The time course of anticipatory constraint integration. *Cognition*.

Mirman, D., Dixon, J., & Magnuson, J. S. (2008). Statistical and computational models of the visual world paradigm: Growth curves and individual differences. *Journal of Memory and Language*, *59*, 474–494.

R Development Core Team. (2012). R: A language and environment for statistical computing. Vienna, AU: R Foundation for Statistical Computing. Retrieved from http://www.r-project.org

Schafer, G., & Plunkett, K. (1998). Rapid word learning by fifteen-month-olds under tightly controlled conditions. *Child Development*, *69*(2), 309–320.

Shukla, M., White, K. S., & Aslin, R. N. (2011). Prosody guides the rapid mapping of auditory word forms onto visual objects in 6-mo-old infants. *Proceedings of the National Academy of Sciences of the United States of America*, *108*(15), 6038–6043. doi:10.1073/pnas.1017617108

Swingley, D., & Aslin, R. N. (2000). Spoken word recognition and lexical representation in very young children. *Cognition*, *76*(2), 147–166.

Waxman, S. R., Lidz, J., Braun, I. E., & Lavin, T. (2009). Twenty four-month-old infants' interpretations of novel verbs and nouns in dynamic scenes. *Cognitive Psychology*, *59*(1), 67–95. doi:10.1016/j.cogpsych.2009.02.001

Werker, J. F., Cohen, L. B., Lloyd, V. L., Casasola, M., & Stager, C. L. (1998). Acquisition of word–object associations by 14-month-old infants. *Developmental Psychology*, *34*(6), 1289.

Appendix - Function Reference