WHAT YOU SEE IS WHAT YOU GET: A CLOSER LOOK AT BIAS IN THE VISUAL WORLD PARADIGM

by

Collin Nolte

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Biostatistics in the
Graduate College of The
University of Iowa

May 2023

Thesis Committee: Patrick Breheny, Thesis Supervisor
Jacob Oleson
Bob McMurray
Grant Brown
Kristi Hendrickson

Copyright by
COLLIN NOLTE
2019
All Rights Reserved

ACKNOWLEDGMENTS

acknowledgment

ABSTRACT

abstract

PUBLIC ABSTRACT

Public abstract

TABLE OF CONTENTS

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER

1 Introduction 1

2 bdots 4

2.1 Introduction 4
2.2 Methodology and Overview 6
2.2.1 Establishing subject-level curves 6
2.2.2 Estimating Group Distributions 7
2.2.3 Hypothesis testing for statistically significant differences 8
2.3 Example Analysis 10
2.3.1 Curve Fitting 10
2.3.2 Bootstrapping 17
2.4 Ancillary Functions 20
2.4.1 Refitting 20
2.4.2 Correlations 23
2.4.3 α Adjustment 23
2.5 Discussion 24
2.6 Appendix 25
2.6.1 Formula for nested groups and difference of differences 25
2.7 Writing Custom Curve Functions 27

3 look onset method 38

3.1 Introduction 38
3.2 Analysis with VWP Data 41
3.2.1 Anatomy of Eye Mechanics 41
3.2.2 Activation 44
3.2.3 VWP data 48
3.3 Bias/Look Onset (better name for section) 50
3.4 Simulations 56
3.4.1 No Delay 59
3.4.2 Normal Delay 61
3.4.3 Weibull Delay 63
3.4.4 Results 64
3.5 Discussion 67
3.6 Appendices 69
3.6.1 No Delay 70
3.6.2 Normal Delay 72
3.6.3 Weibull Delay 74

3.6.4 Results 75

4 bdots methodology 77

4.1 Introduction 77
4.2 Methods 78
4.2.1 Homogeneous bootstrap 78
4.2.2 Heterogeneous Bootstrap 79
4.2.3 Permutation Testing 82
4.3 Type I Error Simulations 83
4.3.1 Data Generation 83
4.3.2 Results 85
4.3.3 Discussion 89
4.4 Power Simulations 89
4.4.1 Results 91
4.5 Discussion and concluding remarks 95

LIST OF TABLES

Table

2.1 Description of the `fitCode` statistic 14

2.2 Example of nested vehicle classes 26

3.1 Summary of mean integrated squared error across simulations 65

3.2 Figures not as striking as logistic, but keep in mind that the scale of this is significantly smaller, peaking at around 0.15 and being close to zero elsewhere. R^2 is another metric available 75

3.3 R^2 for logistic, not as clear a hierarchy 76

3.4 R^2 for double gauss 76

4.1 FWER for empirical parameters (unpaired) 87

4.2 FWER for empirical parameters (paired) 87

4.3 median per comparison error rate (unpaired) 88

4.4 median per comparison error rate (paired) 89

4.5 Power for methods (possibly include AR(1) error and AR(1) specification to compare to last setting) 92

4.6 Summary of methods for Type II error 92

LIST OF FIGURES

Figure

2.1 Illustration of Mouse data in long format 10

2.2 Main arguments to `bfit`, though see `help(bfit)` for additional options . . . 11

2.3 A `bfit` object inheriting from `data.frame` 12

2.4 Abridged output from the summary function (missing summary information for groups B-E. Note that this includes data on the formula used, the quality of fits and mean parameter estimates by group, and a summary of all fits combined 15

2.5 Plot of `mouse_fit`, using `ggplot2` (default) and `data.table` syntax to subset to only the first four observations 16

2.6 Bootstrapped distributions with regions of significant difference determined via permutation testing 19

2.7 before and after refit 22

2.8 A call to the function `bfit` with data `X` and outcome and time variables `"y"` and `"time"`. `bfit` splits the dataset `X` by subject/group and passes each individual `data.frame` into the curve function `f()`, along with time and outcome character vectors as well as any other arguments passed into `...`. In particular, the `...` argument allows the user to specify characteristics of the curve function that apply to all instances, as would be the case, for example, if `curveFun = polynomial(degree = 5)`. Finally, each instance of `f(...)` returns both a formula for `lmer::gnls` as well as subject-specific starting parameters. 29

2.9 An example curve function with its constituent parts 30

2.10 Example data for a four parameter logistic 33

2.11 Estimate starting parameters for empirical asymmetric Gaussian 34

2.12 Using random distribution to estimate starting parameters 36

3.1 An illustration of the four-parameter logistic and its associated parameters, introduced as a parametric function for fixations to target objects in McMurray 2010. Can describe the parameters in detail, but should also have the formula itself somewhere to be referenced. (Equation 4.10) 42

3.2 redo this image to match anataomy of look image, also for size 43

3.3 redo this image 44

3.4 I want to do this figure again but differently. have saccade be two bars matching anatomy of look, include refractory period of fixation, noting that that and saccade are identical, followed by period of time of voluntary fixation (theoretically relevant) followed by next CM 45

3.5 Alternative to Figure 3.4 with diagram of saccade/fixation more consistent with Figure 3.8. There may be a bit more than is needed here, especially with the pink and blue (and maybe even subsequent CM). Although I don't discuss their use in detail, the pink/blue/second CM highlight the fact that *even when* we argue that using proportion of fixation method indirectly captures strength of activation via length of duration, the assumption of linearity is incorrect. For the sake of argument, assuming that the refractory period is exactly 100ms and the length of oculomotor delay is 200ms, a fixation of length 500ms may seem to be 25% larger than a fixation lasting 400ms, but in terms of the *intentional* fixation period, it is twice as large. This serves both to demonstrate another issue with the proportion method while also paving the way for incorporating fixation length in the future. Still, this may not all be necessary here, or presented in a cleaner way 46

3.6 Stole this from Bob (who apparently stole it from richard aslin), plan on making my own 50

3.7 **(a.)** Example of a nonlinear activation curve $f(t|\theta)$ **(b.)** At some time, t, a saccade is launched with its destination probabilistically determined by $f(t|\theta)$ **(c.)** For a look persisting over n time points, $t+1, \ldots, t+n$, we are recording "observed" data, adding to the proportion of fixations at each time but without having gathered any additional observed data at $f(t+1|\theta), \ldots, f(t+n|\theta)$, thus inflating (or in the case of a monotonically increasing function like the logistic, deflating) the true probability. 54

3.8 Anatomy of a look – a key thing to discuss somewhere is the OM delay, refractory period, and planning time. The latter two go in γ. Worth noting also that while we do need to be able to control for ρ, *information* regarding strength of consideration will be in γ less the refractory period 57

3.9 Parameter bias with no oculomotor delay 59

3.10 Representative curves with no oculomotor delay 60

3.11 Parameter bias with normal oculomotor delay 61

3.12 Representative curves with normal oculomotor delay 62

3.13 Parameter bias with Weibull oculomotor delay 63

3.14 Representative curves with Weibull oculomotor delay 64

3.15 Parameter bias with no oculmotor delay 70

3.16 Representative curves for no oculmotor delay 71

3.17 Parameter bias for normal OM delay 72

3.18 Representative curves for normal oculomotor delay 73

3.19 Parameter bias for weibull OM delay 74

3.20 Representative curves for weibull oculmotor delay 75

4.1 Distributions of various group with the mean curve in bold (a.) 50 samples from the generating distribution of the four-parameter logistic in Equation 4.10 used for testing FWER. (b.) 50 samples from the generating distributions of each group in Equation 4.12. The legend makes size weird, might just explain what they are here. Also need to see if I can change size of the mean lines 84

4.2 Observed power of each of the methods at each time in (-1,1) (not sure if better as subplots?) 94

CHAPTER 1
INTRODUCTION

Last compiled: Friday 3rd March, 2023 at 12:38

This dissertation is made up of three chapters, each addressing some aspect of eye tracking in the Visual World Paradigm (VWP) or `bdots`, the R software created to accompany the analysis. Each of this chapters is also written as a separate manuscript. As such, there will be some overlap and redundancy between chapters, though this is intentional as they are intended to stand alone. And though they are presented necessarily in a linear fashion, they can be read in any order.

The first chapter is bdots and address major changes in the `bdots` package, including user interface, functionality, and general quality-of-life improvements. Included also is an use-case example performing an analysis with non-VWP data, an analysis of tumor growth in mice across a number of experimental treatment groups. This chapter also includes important changes to the underlying methodology, including a change to the way the original bootstrapping algorithm is constructed, as well as the introduction of a permutation test to control the family wise error rate. These methods are included both for completeness for new users and to demonstrate important changes for existing users. Justification for these changes is provided in Chapter 3.

The second chapter address data in the VWP itself. It begins with a general review of the VWP, eye tracking, and how these are used in relation to one another to assess lexical activation in time. This chapter examines critical issues in the current "proportion of fixation" method, presents a more comprehensive model for describing the underlying cognitive mechanisms of interest with the observed physiological behavior, and introduces a novel "look onset" method for using data in the VWP. We justify this both with a theoretical argument as to why this method would be superior for the recovery of the underlying cognitive mechanisms and verify the veracity of our claims with a number of simulations. We

conclude this chapter by suggesting a number of ways this new method can be incorporated into current theories of lexical activation.

Finally, the third chapter of this dissertation addresses the underlying methodology presented in the original 2017 bootstrapping differences of time series paper. In particular, it identifies a critical issue in the underlying assumptions as they relate to empirically observed data and the impacts on the family-wise error rate when these assumptions do not hold. Presented in this chapter is a modification of the original bootstrapping algorithm that still uses the modified Bonferonni correction based on the autocorrelation of the test statistics in time, as well as the introduction of a permutation test. We demonstrate that both the modified bootstrap and the permutation test maintain a FWER close to the nominal rate across a number of differing assumptions and conclude with a comparison of power for each of the methods discussed. The results of Chapter 3 provide justification for the underlying changes in Chapter 1.

Though parts of this dissertation are still in progress, the primary arguments being made in each chapter are complete and prepared to stand on their own. There are several places where information may be repeated or areas that transition rather abruptly from one section to the next or may be otherwise incomplete. Some images have become slightly outdated as the arguments being made materialized (particularly in Chapter 2), though these are being actively addressed each day. Given the computational intensity of some of the simulations, there are a few that have only run 100 times (for example, in Chapter 3), though as we increase the iterations we have little expectation of the results fundamentally changing. We intend on rerunning each of these 1000 times prior to final submission. Lastly, there are a few places where a number of questions have been asked with only the relevant results presented. This includes a number of settings for the simulations in Chapter 3, a more in depth treatment of user-specified curves in Chapter 1, or the relating of the look onset method to TRACE data to demonstrate theoretical consistency in Chapter 2. Where relevant enough to be included but not central to the arguments in the paper, these topics

will be moved to an appendix. When of less relevance, they will be omitted altogether. To ensure that you have the most up to date version, you can compare the time stamp at the top of this section with that on the version hosted online.

Finally, I would like to thank you in advance the time and attentiveness that you give to reading this manuscript, your patience in the portions that are less refined, and for the feedback that you are able to provide.

CHAPTER 2
BDOTS

2.1 Introduction

In 2017, Oleson et al. introduced a method for detecting time-specific differences in the trajectory of outcomes between experimental groups [11]. Particularly in the case of a densely sampled time series, the construction of evaluating differences at each point in time results in a series of highly correlated test statistics expanding the family-wise error rate, accommodated with an adjustment to the nominal alpha based on this autocorrelation. This was followed up with in 2018 with the introduction of the `bdots` package to CRAN [12]. Here, we introduce the second version of `bdots`, an update to the package that broadly expands the capabilities of the original.

This manuscript is not intended to serve as a complete guide for using the `bdots` package. Instead, the purpose is to showcase major changes and improvements, with those seeking a more comprehensive treatment directed to the package vignettes. Rather than taking a "compare and contrast" approach, we will first enumerate the major changes, followed by a general demonstration of the package use:

1. Major changes to underlying methodology with implications for prior users of the package
2. Simplified user interface
3. Introduction of user defined curves
4. Permit fitting for arbitrary number of groups
5. Automatic detection of paired tests based on subject identifier
6. Allows for non-homogeneous sampling of data across subjects and groups
7. Introduce formula syntax for bootstrapping difference function
8. Interactive refitting process

We start by clearly delineating the type of problem that `bdots` has been created to solve.

Bootstrapped differences in time series

A typical problem in the analysis of differences of time series, and the kind that `bdots` is intended to solve, involves that of two or more experimental groups containing subjects whose responses are measured over time. This may include the growth of tumors in mice or the change in the proportion of fixations over time in the context of the VWP. In either case, we assume that each of the subjects $i = 1, \ldots, n$ in the groups being considered has observed data of the following form:

$$y_{it} = f(t|\theta_i) + \epsilon_{it} \tag{2.1}$$

where f represents a functional mean structure while the error structure of ϵ_{it} is open to being either IID or possess an AR(1) structure. At present, `bdots` requires that each of the subjects being compared have the same parametric function f, though is not strictly necessary at the theoretical level and future directions of the package include accommodating non-parametric functions. While each of the subjects are required to be of the same parametric form $f(\cdot|\theta)$, each differs in their instance of their own subject-specific parameters, θ_i.

An explicit assumption of the current iteration of `bdots` is that each subject is' parameters within a group $g = 1, \ldots, G$ is drawn from a group level distribution

$$\theta_i \sim N(\mu_g, V_g). \tag{2.2}$$

Bootstrapping parameters to estimate this distribution and evaluating the function f at these values gives us an estimate of the distribution of functions. As these function are *in time*, this in turn gives a representation of the temporal changes in group characteristics. It is precisely the identification of if and when these temporal changes differ between groups that `bdots` seeks to perform.

Homogeneous Means Assumption

The assumption presented in Equation 2.2 differs from the original iteration of `bdots` in a critical way. In [11], there was no assumption of variability between subject parameters, congruent the assumption that $\theta_i = \theta_j$ for all subjects i, j within an experimental group. The most relevant consequence of the homogeneous means assumption is that only within-subject variability is estimated, inflating the type I error when this assumption does not hold (see Chapter 3 for a discussion). For now, we will give a methodological overview of the process used by `bdots` along with a presentation of an updated bootstrapping process and the introduction of a permutation test.

2.2 Methodology and Overview

A standard analysis using `bdots` consists of two steps: fitting the observed data to a specified parametric function, $f(\cdot|\theta)$, and then using the observed variability to construct estimates of the distributions of groups whose differences we wish to investigate. Here, we briefly detail how this is implemented in practice and introduce the new methodologies in `bdots`. A more comprehensive treatment of these new methods, along with their justifications, is offered in Chapter 3.

2.2.1 Establishing subject-level curves

We begin with the assumption that for subject i in group g, we have collected observed data of the form given in Equation 4.2.2, with the subject specific parameter θ_i following the distribution in Equation 2.2. Each subject is then fit in `bdots` via the nonlinear curve fitting function `nlme::gnls`, returning for each set of observed data an estimated set of parameters $\hat{\theta}_i$ and their associated standard errors. Assuming large sample normality, we are able to construct a sampling distribution for each subject:

$$\hat{\theta}_i \sim N(\theta_i, s_i^2). \tag{2.3}$$

This provides us with an estimate of within-subject variability of the observed parameters.

2.2.2 Estimating Group Distributions

Once sampling distributions are created for each subject, we are prepared to begin estimating group distributions given in Equation 2.2. We then propose the following algorithm for creating bootstrapped estimates of the group distributions:

1. For a group of size n, select n subjects from the group *with replacement*. This allows us to construct an estimate of V_g.

2. For each selected subject i in bootstrap b, draw a set of parameters from the distribution

$$\theta^*_{ib} \sim N(\hat{\theta}_i, s_i^2). \tag{2.4}$$

3. The the mean of each of the bootstrapped θ^*_{ib} in group g to construct the bth group bootstrap, θ^*_{gb} where

$$\theta^*_{gb} \sim N\left(\mu_g, \frac{1}{n}V_g + \frac{1}{n^2}\sum s_i^2\right) \tag{2.5}$$

4. Perform steps (1)-(3) B times, using each θ^*_b to construct a distribution of population curves, $f(\cdot|\mu_g)$. (not sure I like this notation)

Note that the distribution in Equation 2.5 differs from that under the homogeneous means assumption by the factor of V_g in the variance term.

The final population curves from (4) can be used to create estimates of the mean response and an associated standard deviation at each time point for each of the groups bootstrapped. These estimates are used both for plotting and in the construction of confidence intervals. They also can be, but do not necessarily have to be, used to construct a test statistic, which is the topic of our next section.

2.2.3 Hypothesis testing for statistically significant differences

We now turn our attention to the primary goal of an analysis in `bdots`, the identification of time windows in which the distribution of curves of two groups differ significantly. A problem unique to the ones addressed by `bdots` is that of multiple testing; and especially in densely sampled time series, we must account for multiple testing while controlling the family-wise error rate (FWER). There are primarily two ways by which we are able to do this which we detail below.

2.2.3.1 α Adjustment

Just as in the original iteration of `bdots`, we are able to construct test statistics from the bootstrapped estimates described in the previous section. These bootstrapped test statistics $T_t^{(b)}$ can be written as

$$T_t^{(b)} = \frac{(\overline{p}_{1t} - \overline{p}_{2t})}{\sqrt{s_{1t}^2 + s_{2t}^2}}, \tag{2.6}$$

where $\overline{p}_{gt}$ and s_{gt}^2 are mean and standard deviation estimates at each time point t and for groups 1 and 2, respectively. As was demonstrated in [11], these test statistics can be highly correlated in the presence of densely sampled test statics, leading to an inflated type I error. The FWER in this case can be controlled with the Oleson adjustment proposed in original bootstrap paper. In addition to this, adjustments to the nominal alpha can also be made using all of the adjustments present in `p.adjust` from the R `stats` package.

2.2.3.2 Permutation testing

In addition to modified correction based on the bootstrapped test statistics, `bdots` provides a permutation test for controlling the FWER without any additional assumptions of autocorrelation.

In doing so, we begin by creating an observed test statistic in the following way: first, taking each subject's estimated parameter $\hat{\theta}_i$, we find the subject's corresponding parametric curve $f(t|\hat{\theta}_i)$. Within each group, we use *these* curves to create estimates of the mean population curves and associated standard errors at each each point[1]. Letting p_{gt} and s^2_{gt} represent the mean population curve and standard error for group g at time t, we define our observed permutation test statistic,

$$T_t^{(p)} = \frac{|\overline{p}_{1t} - \overline{p}_{2t}|}{\sqrt{s^2_{1t} + s^2_{2t}}}. \tag{2.7}$$

.

We then going about using permutations to construct a null distribution against which to compare the observed statistics from Equation 2.7. We do so with the following algorithm:

1. Assign to each subject a label indicating group membership
2. Randomly shuffle the labels assigned in (1.), creating two new groups
3. Recalculate the test statistic $T_t^{(p)}$, recording the maximum value from each permutation
4. Repeat (2.)-(3.) P times. The collection of P statistics will serve as our null distribution, denoted $\widetilde{T}$. Let $\widetilde{T}_\alpha$ be the $1-\alpha$ quantile of $\widetilde{T}$. Areas where the observed $T_t^{(p)} > \widetilde{T}_\alpha$ are designated significant.

Paired statistics can also be constructed in both the bootstrap and permutation methods. This is implemented by ensuring that at each bootstrap the same subjects are selected for each group or by ensuring that each permuted group contains one observation from each subject.

A demonstration of power and FWER control for both the heterogeneous bootstrap and permutation test are given in Chapter 3.

[1]This differs from the bootstrapped test statistic in which the mean of the subjects' parameters was used to fit a population curve, i.e., $\frac{1}{n}\sum f(t|\theta_i)$ compared with $f\left(t|\frac{1}{n}\sum\theta_i\right)$. The implications of this have not been investigated any further

2.3 Example Analysis

In this next section we are going to review a worked example of a typical use of the `bdots` package. We will use as our illustration a study (source?) comparing tumor growth for the 451LuBr cell line in mice data with repeated measures in five treatment groups.

```
> head(mouse, n = 10)
      Volume Day Treatment ID
 1:   47.432   0         A  1
 2:   98.315   5         A  1
 3:  593.028  15         A  1
 4:  565.000  19         A  1
 5: 1041.880  26         A  1
 6: 1555.200  30         A  1
 7:   36.000   0         B  2
 8:   34.222   4         B  2
 9:   45.600  10         B  2
10:   87.500  16         B  2
```

Figure 2.1: Illustration of Mouse data in long format

A new feature of `bdots` is the ability to fit and analyze subjects with non-homogeneous time samples, which we see in the `Day` variable values in the mouse data in Figure 2.1. For the present analysis, we are interested in determining if and when the trajectory of tumor growth, measured in (Volume?) changes between any two treatment groups.

There are two primary functions in the `bdots` package: one for fitting the observed data to a parametric function and another for estimating group distributions and identifying time windows where they differ significantly. The first of these, `bfit`, is addressed in the next section.

2.3.1 Curve Fitting

The curve fitting process is performed with the `bfit` function (previously `bdotsFit`), taking the following arguments:

The `data` argument takes the name of the dataset being used. `subject` is the subject

```
bfit(data, subject, time, y, group, curveType, ar, ...)
```

Figure 2.2: Main arguments to `bfit`, though see `help(bfit)` for additional options

identifier column in the data and should be passed as a character. The `time` and `y` arguments are column names of the time variable and outcome, respectively. Similarly, `group` takes as an argument a character vector of each of the group columns that are meant to be fit, accommodating the fact that `bdots` is now able to fit an arbitrary number of groups at once, provided that the outcomes in each group adopt the same parametric form. This brings us to the `curveType` argument, which is addressed in the next section. [need to describe `ar`]

[say this somewhere] It is important to note here that the identification of paired data is done automatically; in determining if two experimental groups are paired, `bdots` checks that the intersection of subjects in each of the groups are identical with the subjects in each of the groups individually.

Curve functions

Whereas the previous iteration of `bdots` had a separate fitting function for each parametric form (i.e., `logistic.fit` for fitting data to a four-parameter logistic), we are now able to specify the curves we wish to fit independent of the fitting function `bfit`. This is done with the `curveType` argument. Unlike the previous arguments which took either a `data.frame` or character vector, `curveType` takes as an argument a function call, for example, `logistic()`. The motivation for this is detailed elsewhere (the appendix, maybe?), but in short, it allows the user to pass additional arguments to further specify the curve. For example, among the parametric functions included in `bdots` is now the `polynomial` function, taking as an additional argument the number of degrees we wish to use. To fit the observed data with a five parameter polynomial in `bfit`, one would then pass the argument `curveType = polynomial(degree = 5)`. Curve functions currently included in `bdots` include `logistic()`, `doubleGauss()`, `expCurve()`, and `polynomial()`. `bfit` can also accept user-created curves; detailed vignettes for writing your own can be found with

`vignette("bdots")`. [and maybe the appendix]

We fit the mouse data to an exponential curve with `expCurve()` and using the column names found in Figure 2.1:

```
mouse_fit <- bfit(data = mouse, subject = "ID", time = "Day",
                  y = "Volume", group = "Treatment", curveType = expCurve())
```

Return object and generics

The function `bfit` returns an object of class `bdotsObj`, inheriting from class `data.table`. As such, each row uniquely identifies one permutation of subject and group values. Included in this row are the subject identifier, group classification, summary statistics regarding the curves, and a nested `gnls` object. Inheriting from `data.table` also permits us to use `data.table` syntax to subset the object as in Figure 2.5, for example, where we elect to only plot the first four subjects.

```
> class(mouse_fit)
[1] "bdotsObj"   "data.table" "data.frame"

> head(mouse_fit)
   ID Treatment        fit      R2   AR1 fitCode
1:  1         A <gnls[18]> 0.97349 FALSE       3
2:  2         B <gnls[18]> 0.83620 FALSE       4
3:  3         E <gnls[18]> 0.96249 FALSE       3
4:  4         C <gnls[18]> 0.96720 FALSE       3
5:  5         D <gnls[18]> 0.76156 FALSE       5
6:  7         B <gnls[18]> 0.96361 FALSE       3
```

Figure 2.3: A `bfit` object inheriting from `data.frame`

The number of columns will depend on the total number of groups specified, with the subject and group identifiers always being the first columns. Following this is the `fit` column, which contains the fitted object returned from `gnls`, as well as `R2` indicating the R^2 statistic. The `AR1` column indicates whether or not the observed data was able to be fit with an AR(1) error assumption. Finally, there is the `fitCode` column, which we will describe in

more detail shortly.

Several methods exist for this object, including `plot`, `summary`, and `coef`, returning a matrix of fitted coefficients obtained from `gnls`.

Fit Codes

The `bdots` package was originally introduced to address a very narrow scope of problems, and the `fitCode` designation is an artifact of this original intent. Specifically, it assumed that all of the observed data was of the form given in Equation 4.2.2 where the observed time series was dense and the errors were autocorrelated. Autocorrelated errors can be specified in the `gnls` package (used internally by `bdots`) when generating subject fits, though there were times when the fitter would be incapable of converging on a solution. In that instance, the autocorrelation assumption was dropped and constructing a fit was reattempted.

R^2 proved a reliable metric for this kind of data, and preference was given to fits with an autocorrelated error structure over those without. From this, the hierarchy given in Table 2.1 was born. `fitCode` is a numeric summary statistic ranked from 0 to 6 detailing information about the quality of the fitted curve, constructed with the following pseudo-code:

```
AR1 <- # boolean, determines AR1 status of fit
fitCode <- 3*(!AR1) + 1*(R2 < 0.95)*(R2 > 0.8) + 2*(R2 < 0.8)
```

A fit code of 6 indicates that `gnls` was unable to successfully fit the subject's data.

`bdots` today stands to accommodate a far broader range of data for which the original `fitCode` standard may no longer be relevant. The presence of autocorrelation cannot always be assumed, and users may opt for a metric other than R^2 for assessing the quality of the fits. Even the assessments of fits on a discretized scale may be something of only passing interest. Even then, however, this is how the current implementation of `bdots` categorizes the quality of its fits, with the creation of greater flexibility in this regard being a large priority for future directions. Outside of general summary information, the largest impact

of this system is in the refitting process, which organizes the fits by `fitCode`. There is still flexibility in how this is handled, though we will reserve discussion for the relevant section.

`fitCode`	AR(1)	R^2
0	TRUE	$R^2 > 0.95$
1	TRUE	$0.8 < R^2 < 0.95$
2	TRUE	$R^2 < 0.8$
3	FALSE	$R^2 > 0.95$
4	FALSE	$0.8 < R^2 < 0.95$
5	FALSE	$R^2 < 0.8$
6	NA	NA

Table 2.1: Description of the `fitCode` statistic

Summaries and Plots

Users are able to quickly summarize the quality of the fits with the `summary` method now provided.

```
> summary(mouse_fit)

bdotsFit Summary

Curve Type: expCurve
Formula: Volume ~ x0 * exp(Day * k)
Time Range: (0, 106) [31 points]

Treatment: A
Num Obs:  10
Parameter Values:
        x0          k
172.232953   0.056843
########################################
############### FITS ###################
########################################
AR1,       0.95 <= R2        -- 2
AR1,       0.80 < R2 <= 0.95 -- 1
AR1,       R2 < 0.8          -- 0
Non-AR1,   0.95 <= R2        -- 0
Non-AR1,   0.8 < R2 <= 0.95  -- 3
Non-AR1,   R2 < 0.8          -- 4
No Fit                       -- 0

[...]

All Fits
Num Obs:  42
Parameter Values:
        x0          k
102.487118   0.053662
########################################
############### FITS ###################
########################################
AR1,       0.95 <= R2        -- 4
AR1,       0.80 < R2 <= 0.95 -- 2
AR1,       R2 < 0.8          -- 0
Non-AR1,   0.95 <= R2        -- 9
Non-AR1,   0.8 < R2 <= 0.95  -- 16
Non-AR1,   R2 < 0.8          -- 11
No Fit                       -- 0
```

Figure 2.4: Abridged output from the summary function (missing summary information for groups B-E. Note that this includes data on the formula used, the quality of fits and mean parameter estimates by group, and a summary of all fits combined

It is also recommended that users visually inspect the quality of fits for their subjects, which includes a plot of both the observed and fit data. There are a number of options available in `?plot.bdotsObj`, including the option to fit the plots in base R rather than `ggplot2`. This is especially helpful when looking to quickly assess the quality of its (rather than reporting) because `ggplot2` can be notoriously slow with large data sets. Figure 2.5 includes a plot of the first four fitted subjects.

```
plot(mouse_fit[1:4, ])
```

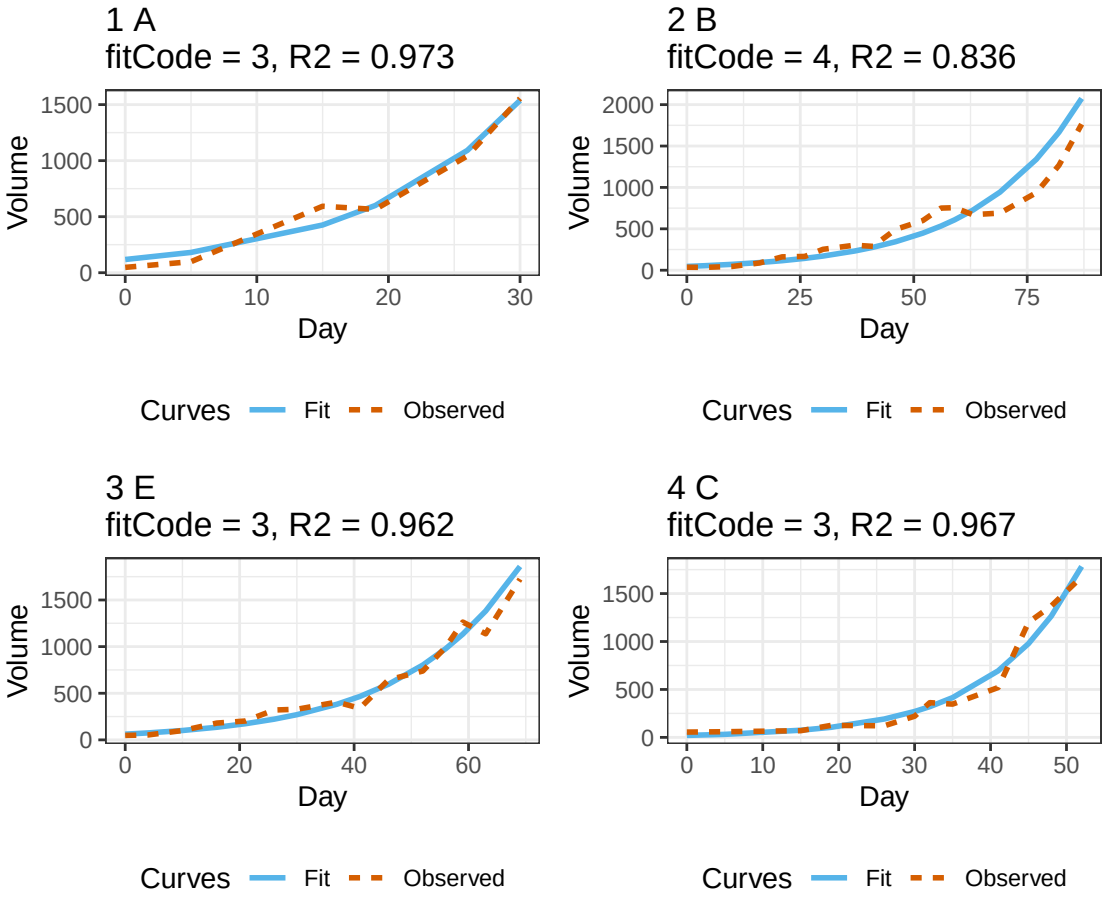


Figure 2.5: Plot of `mouse_fit`, using `ggplot2` (default) and `data.table` syntax to subset to only the first four observations

2.3.2 Bootstrapping

Once fits have been made, we are ready to begin estimating the group distributions and investigating temporal differences. This is done with the bootstrapping function, `bboot`. The number of options included in the `bboot` function have expanded to include a new formula syntax for specifying the analysis of interest as well as to include options for permutation testing. A call to `bboot` takes the following form:

```
bboot(formula, bdObj, B, alpha, permutation = TRUE, padj = "oleson", ...)
```

The `formula` argument is new to `bdots` and will be discussed in the next section. As for the remaining arguments, `bdObj` is simply the object returned from `bfit` that we wish to investigate, and `B` serves the dual role of indicating the number of bootstraps/permutations we wish to perform; `alpha` is the rate at which we wish to control the FWER. `permutation` and `padj` work in contrast to one another: when `permutation = TRUE`, the argument to `padj` is ignored. Otherwise, `padj` indicates the method to be used in adjusting the nominal `alpha` to control the FWER. By default, `padj = "oleson"`. Finally, as previously mentioned, there is no longer a need to specify if the groups are paired, and `bboot` determines this automatically based on the subject identifiers in each of the groups.

Formula

As the `bfit` function is now able to create fits for an arbitrary number of groups at once, we rely on a formula syntax in `bboot` to specify precisely which groups differences we wish to compare. Let `y` designate the outcome variable indicated in the `bfit` function and let `group` be one of the group column names to which our functions were fit. Further, let `val1` and `val2` be two values within the `group` column. The general syntax for the `bboot` function takes the following form:

$$\texttt{y} \sim \texttt{group(val1, val2)}$$

Note that this is an *expression* in R and is written without quotation marks. To give a

more concrete example, suppose we wished to compare the difference in tumor growth curves for A and B from the `Treatment` column in our mouse data (Figure 2.1). We would do so with the following syntax:

```
Volume ~ Treatment(A, B)
```

There are two special cases to consider when writing this syntax. The first is the situation that arises in the case of multiple or nested groups, the second when a difference of difference analysis is conducted. Details on both of these cases are handled in the appendix.

Summary and Analysis

Let's begin first by running `bboot` using bootstrapping to compare the difference in tumor growth between treatment groups A and E in our mouse data using permutations to test for regions of significant difference.

```
mouse_boot <- bboot(Volume ~ Treatment(A, E), bdObj = mouse_fit, permutation
= TRUE)
```

This returns an object of class `bdotsBootObj`. A summary method is included to display relevant information:

```
> summary(mouse_boot)

bdotsBoot Summary

Curve Type: expCurve
Formula: Volume ~ x0 * exp(Day * k)
Time Range: (0, 59) [21 points]

Difference of difference: FALSE
Paired t-test: FALSE
Difference: Treatment

FWER adjust method: Permutation
Alpha: 0.05
Significant Intervals:
     [,1] [,2]
[1,]   15   32
```

There are a few components of the summary that are worth identifying when reporting the results. In particular, note the time range provided, an indicator of if the test was paired, and which groups were being considered . The last section of the summary indicates the testing method used, an adjusted `alphastar` if `permutation = FALSE`, and a matrix of regions identified as being significantly different. This matrix is `NULL` is no differences were identified at the specified alpha; otherwise there is one row included for each disjointed region of significant difference.

In addition to the provided summary output, a `plot` method is available, with a list of additional options included in `help(plot.bdotsBootObj)`.

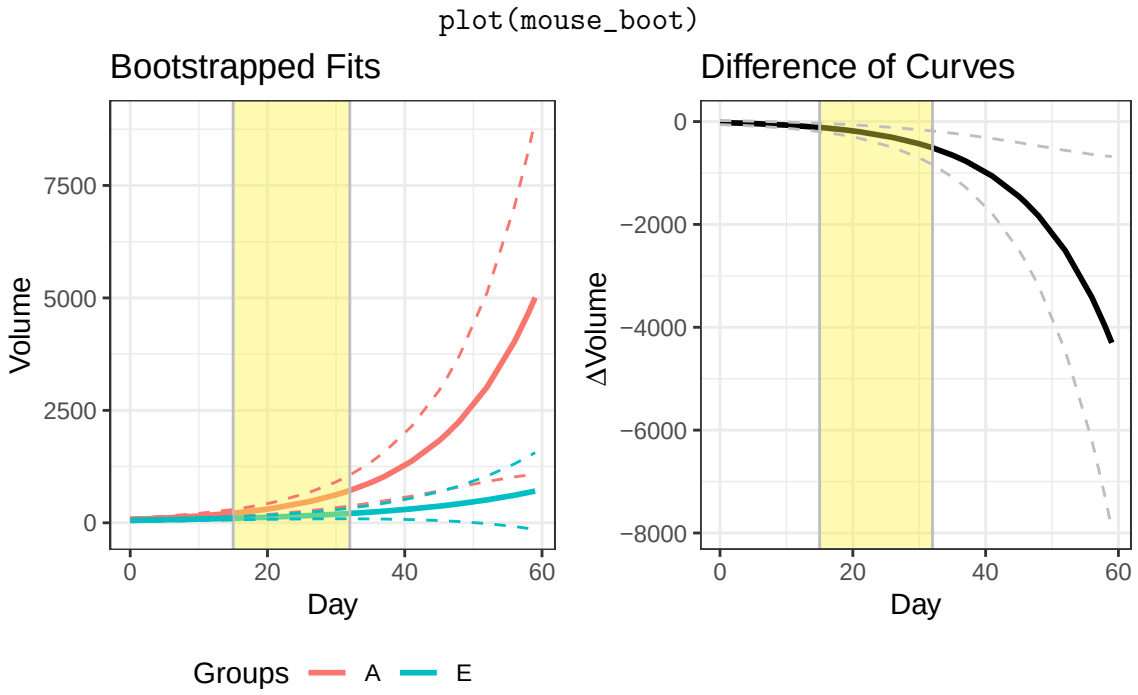


Figure 2.6: Bootstrapped distributions with regions of significant difference determined via permutation testing

2.4 Ancillary Functions

Include here are a number of different function in `bdots` facilitating analysis but are otherwise not strictly necessary

2.4.1 Refitting

There are sometimes situations in which the fitted function returned by `bfit` is a poor fit. The nonlinear curve fitting algorithm used by `nlme::gnls` in `bfit` can be sensitive to starting parameters. Sensible starting parameters are computed from the observed data as part of the curve fitting functions (i.e., within the `logistic()` function), though these can often be improved upon.

The quality of the fit can be evidenced by the `fitCode` or via a visual inspection of the fitted functions against the observations for each subject. When this occurs, there are several options available to the user, all of which are provided through the function `brefit` (previously `bdotsRefit`). `brefit` takes the following arguments:

```
brefit(bdObj, fitCode = 1L, subset = NULL, quickRefit = FALSE, paramDT = NULL)
```

The first of these arguments outside of the `bdObj` is `fitCode`, indicating the minimum fit code to be included in the refitting process. As discussed in Section 2.3.1, this can be sub-optimal. To add flexibility to which subjects are fit there is now the `subset` argument taking either a logical expression or collection of indices that would be used to subset an object of class `data.table` or a numeric vector with indices that the user wishes to refit. For example, we could elect to refit only the first 10 subjects or refit subjects with $R^2 < 0.9$:

```
refit <- brefit(fit, subset = 1:10) # refit the first 10 subjects
refit <- brefit(fit, subset = R2 < 0.9) # refit subjects with R2 < 0.9
```

When an argument is passed to subset, the `fitCode` is completely ignored.

To assist with the refitting process is the argument `quickRefit`. When set to `TRUE`, `brefit` will take the average coefficients of accepted fits within a group and use those as new starting parameters for poor fits. The new fits will be retained if they have a larger R^2 value by default. When set to `quickRefit = FALSE`, the user will be guided through a set of prompts to refit each of the curves manually.

Finally, the `paramDT` argument allows for a `data.table` with columns for subject, group identifiers, and parameters to be passed in as a new set of starting parameters. This `data.table` requires the same format as that returned by `bdots::coefWriteout`. The use of this functionality is covered in more detail in the `bdots` vignettes and is a useful way for reproducing a `bdotsObj` from a plain text file.

When `quickRefit = FALSE`, the user is put through a series of prompts along with a series of diagnostics for each of the subjects to be refit. Here, for example, is the option to refit subject 11 from the mouse data:

```
Subject: 11
R2: 0.837
AR1: FALSE
rho: 0.9
fitCode: 4

 Model Parameters:
       x0         k
53.186497  0.051749

Actions:
1) Keep original fit
2) Jitter parameters
3) Adjust starting parameters manually
4) Remove AR1 assumption
5) See original fit metrics
6) Delete subject
99) Save and exit refitter
Choose (1-6):
```

There are a number of options provided in this list. The first, of course, keeps the

original fit of the presented subject and moves on to the next subject in the list. The second option takes the values of the fitted parameter and "jitters" them, changing each of the values by a prespecified magnitude. Given the sensitivity of `nlme::gnls` to starting parameters, this is sometimes enough for the fitter to converge on a better fit for the observed data. Alternatively, the third option gives the user the ability to select the starting parameters manually. The third option gives the user the ability to attempting refitting the observed data without an AR(1) error assumption, though this is only relevant if such an assumption exists. Option (5) reprints summary information and the final option allows the user to delete the subject all together.

When any attempt to refit the observed under new conditions is presented (options (2)-(4)), a plot is rendered comparing the original fit side-by-side with the new alternative, Figure 2.7.

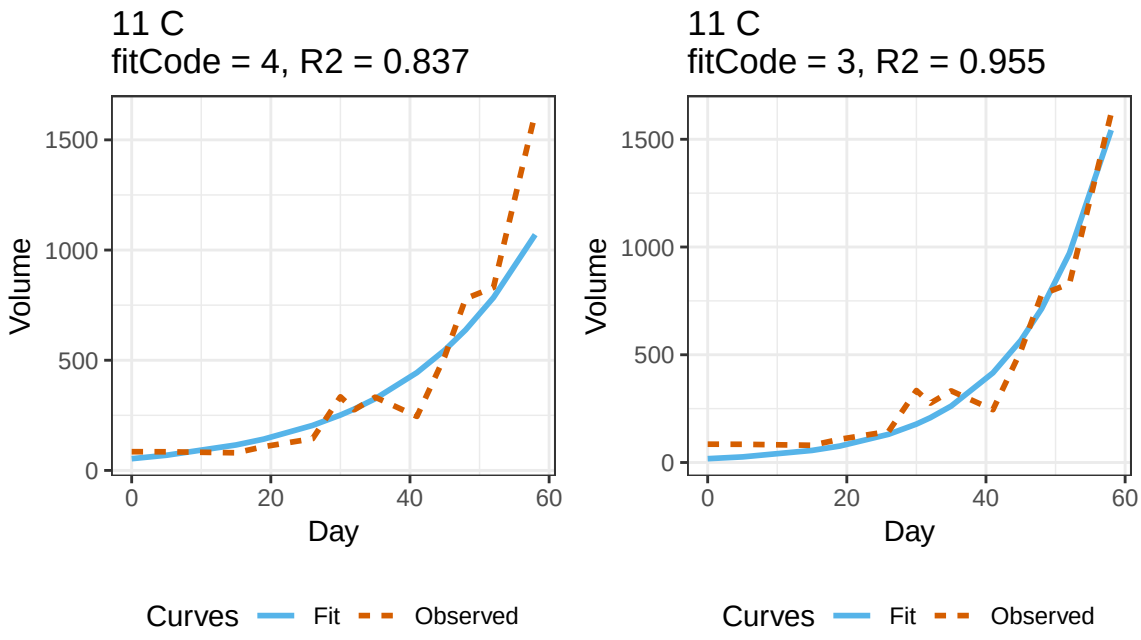


Figure 2.7: before and after refit

As the menu item suggests, users have the ability to end the manually refitting process

early and save where they had left off. To retain previously refit items and start again at a later time, pass the first refitted object back into the refitter as such:

```
refit <- brefit(fit, ...)
refit <- brefit(refit, ...) # pass in the refitted object
```

A final note should be said regarding the option to delete a subject. As `bdots` now automatically determines if subjects are paired based on subject identifiers (necessary for calculations in significance testing), it is critical that if a subject has a poor fit in one group and must be removed that he or she is also removed from all additional groups in order to retain paired status. This can be overwritten with a final prompt in the `brefit` function before they are removed. The removal of subjects can also be done with the ancillary function, `bdRemove`, useful for removing subjects without undergoing the entire refitting process. See `help(bdRemove)` for details.

2.4.2 Correlations

There are sometimes cases in which we are interested in determining the correlation of a fixed attribute with group outcome responses across time . This can be done with the `bcorr` function (previously `bdotsCorr`), which takes as an argument an object of class `bdotsObj` as well as a character vector representing a column from the original dataset used in `bfit`

```
bcorr(fit, "value", ciBands, method = "pearson")
```

Need to elaborate here with example

2.4.3 α Adjustment

There may also be situations in which users wish to make an adjustment to autorcorrelated test statistics using the modified Bonferonni adjustment provided in [11], though in a different context than what is done in `bdots`. To facilitate this, we introduce an extension

to the `p.adjust` function, `p_adjust`, identical to `p.adjust` except that it accepts method `"oleson"` and takes additional arguments `rho`, and `df`. `rho` determines the autocorrelation estimate for the oleson adjustment while `df` returns the degrees of freedom used to compute the original vector of t-statistics. If an estimate of `rho` isn't available, one can be computed on a vector of t-statistics using the `ar1Solver` function in `bdots`:

```
t       <- diffinv(rnorm(5))
rho     <- ar1Solver(t)
unadj_p <- pt(t, df = 10)
adj_p   <- p_adjust(unadj_p, method = "oleson",
                    df = 10, rho = rho, alpha = 0.05)
```

The `p_adjust` function returns both adjusted p-values, which can be compared against the specified alpha (in this case, 0.05) along with an estimate of alphastar, a nominal alpha at which one can compare the original p-values:

```
> unadjp
[1] 0.5000000 0.0849965 0.0381715 0.1601033 0.0247453 0.0013016
> adjp
[1] 0.9201915 0.1564261 0.0702501 0.2946514 0.0455408 0.0023954
attr(,"alphastar")
[1] 0.027168
```

Here, for example, we see that the last two positions of `unadjp` have values less than `alphastar`, identifying them as significant; alternatively, we see these same two indices in `adjp` significant when compared to `alpha = 0.05`

2.5 Discussion

The original implementation of `bdots` set out to address a narrow set of problems. Previous solutions beget new opportunities, however, and it is in this space that the second iteration of `bdots` has sought to expand. Since then, the interface between user and application has been significantly revamped, creating a intuitive, reproducible workflow that is able

to quickly and simply address a broader range of problems. The underlying methodology has also been improved and expanded upon, offering better control of the family-wise error rate.

While significant improvements have been made, there is room for further expansion. The most obvious of these is the need to include support for non-parametric functions, the utility of which cannot be overstated. Not only would this alleviate the need for the researcher to specify in advance a functional form for the data, it would implicitly accommodate more heterogeneity of functional forms within a group. Along with this, the current implementation is also limited in the quality-of-fit statistics used in the fitting steps to assess performance. R^2 and the presence of autocorrelation are relevant to only a subset of the types of data that can be fit, and allowing users more flexibility in specifying this metric is an active goal for future work. In all, future directions of this package will be primarily focused on user interface, non-parametric functions, and greater flexibility in defining metrics for fitted objects.

2.6 Appendix

This section currently commented out. Involves instructions for fitting difference of difference and nested groups. Also may possibly include custom curve fitting.

2.6.1 Formula for nested groups and difference of differences

The formula syntax introduced in Section 2.3.2 is straightforward enough in the case in which we are interested in comparing two groups within a single category, as was the case when we compared two treatment groups, both within the `Treatment` column. As `bdots` now allows multiple groups to be fit at once, there may be situations in which we need more precision in specifying what exactly we wish to compare. Consider for example an artificial dataset that contains some outcome `y` for a collection of vehicles, consisting of eight distinct groups, nested in order of vehicle origin (foreign or domestic), vehicle class (car or truck),

and vehicle color (red or blue)

We will illustrate use of the updated `bdots` package with a worked example, using an artificial dataset to help detail some of the newer aspects of the package. The dataset will consist of outcomes for a collection of vehicles, consisting of eight distinct groups. These groups will be nested in order of vehicle origin (foreign or domesetic), vehicle class (car or truck), and vehicle color (red or blue). Further, vehicles of different color but within the same origin and class groups will be considered paired observations. A table detailing the relationship of the groups is given in Table 2.2.

Origin	Class	Color
foreign	car	red
		blue
	truck	red
		blue
domestic	car	red
		blue
	truck	red
		blue

Table 2.2: Example of nested vehicle classes

Beginning with a simple case, suppose we want to investigate the difference in outcome between all foreign and domestic vehicles. Notionally, we would write

```
y ~ Origin(foreign, domestic)
```

just as we did in the mouse data example: here, the name of the group variable `Origin`, followed by the values we are interested in comparing, `domestic` and `foreign`. Alternatively, if we wanted to limit our investigation to only foreign and domestic *trucks*, we would do this by including an extra term specifying the group and the desired value. In this case,

```
y ~ Origin(foreign, domestic) + Class(truck).
```

Similarly, to compare only foreign and domestic *red* trucks, we would add an additional term for color:

```
y ~ Origin(foreign, domestic) + Class(truck) + Color(red)
```

There are also instances in which we might be considered in the interaction between two groups. Although there is no native way to handle interactions in `bdots`, this can be done indirectly through the difference of differences [7]. To illustrate, suppose we are interested in understanding how the color of the vehicle differentially impacts outcome based on the vehicle class. In such a case, we might look at the difference in outcome between red cars and red trucks and then compare this against the difference between blue cars and blue trucks. Any difference between these two differences would give information regarding the differential impact of color between each of the two classes. This is done in `bdots` using the `diffs` syntax in the formula:

```
diffs(y, Class(car, truck)) ~ Color(red, blue)
```

Here, the *outcome* that we are considering is the difference between vehicle classes, with the groups we are interested in comparing being color. This is helpful in remembering which term goes on the left hand side of the formula.

Similar as to the case before, if we wanted to limit this difference of differences investigation to only include domestic vehicles, we can do so by including an additional term:

```
diffs(y, Class(car, truck)) ~ Color(red, blue) + Origin(domestic).
```

The formula syntax was originally contrived to make comparisons within groups or within nested groups. Conceivably, however, one could be interested in making the comparison between domestic red trucks and foreign blue cars, for example. Doing so requires a bit of a work around, which we detail in the next section.

2.7 Writing Custom Curve Functions

One of the most significant changes in the newest version of `bdots` is the ability to specify the parametric curve independently of the fitting function. Not only does this simply

a typical analysis, reducing all fitting operations to the single function `bfit`, it also provides users with a way to modify this function to meet their own needs. In this section we will detail how the curve function is used in `bdots` and how users can write their own. Finally, we will conclude with an example of how this was used in Chapter 2 to create adequate fits with the look onset method when the typical method for constructing estimated starting parameters based on the proportion of fixation method failed.

To begin, it is important to understand a little of how `bdots` works internally. In the curve fitting steps using `bfit`, the data is split by subject and group, creating a list whereby each element is the set of all observations for a single subject and a single group (i.e., in a paired setting, an individual subject would have two separate elements in this list). Ultimately, the data in each element will be used to construct a set of estimated parameters and standard errors for each subject provided by the function `lmer::gnls`. Doing so requires both (1) a formula to which we fit the data and (2) starting parameter estimates. Proving the both of these is the role of the curve function. Figure 2.8 provides an illustration of this process.

We turn our attention now to the curve function itself, using as an example a curve function for sitting a straight line to the observed data, given in Figure 2.9. In describing the purpose for each, we also include enough detail so that this may be used as a template in constructing a new one all together. We do this in an enumerated list, each item corresponding to the numbered portion in Figure 2.9.

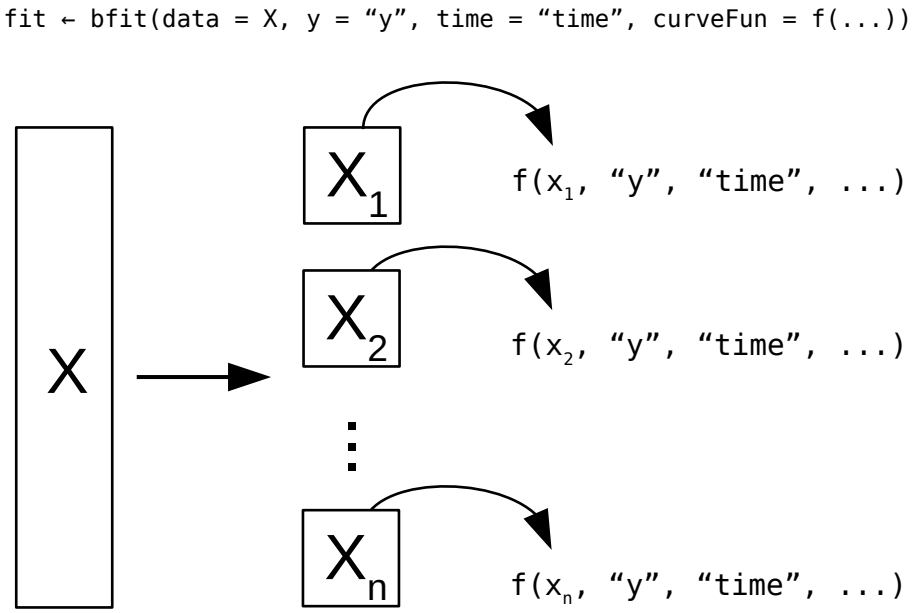


Figure 2.8: A call to the function `bfit` with data `X` and outcome and time variables `"y"` and `"time"`. `bfit` splits the dataset `X` by subject/group and passes each individual `data.frame` into the curve function `f()`, along with time and outcome character vectors as well as any other arguments passed into `....` In particular, the `...` argument allows the user to specify characteristics of the curve function that apply to all instances, as would be the case, for example, if `curveFun = polynomial(degree = 5)`. Finally, each instance of `f(...)` returns both a formula for `lmer::gnls` as well as subject-specific starting parameters.

```
① linear <- function (dat, y, time, params = NULL, ...) {

      linearPars <- function(dat, y, time) {
          time <- dat[[time]]
②         y <- dat[[y]]
          if (var(y) == 0) {
              return(NULL)
          }
          mm <- (max(y) - min(y))/max(time)
          bb <- mean(y) - mm * mean(time)
          return(c(intercept = bb, slope = mm))
      }

③     if (is.null(params)) {
          params <- linearPars(dat, y, time)
      }
④     if (is.null(params)) {
          return(NULL)
      }
      y <- str2lang(y)
      time <- str2lang(time)
⑤     ff <- bquote(.(y) ~ slope * .(time) + intercept)
      attr(ff, "parnames") <- names(params)
      return(list(formula = ff, params = params))
  }
```

Figure 2.9: An example curve function with its constituent parts

1. The first part of the curve function is the collection of arguments to be passed, also known as formals. Each curve function should have an argument `dat`, which takes a `data.frame` as described in Figure 2.8, as well as arguments `y` and `time` which will take character strings indicating which columns of `dat` represent the outcome and time variables, respectively. Following this is the prespecified argument `params = NULL`, which is used by `bdots` during the refitting process, where the estimated starting parameters for the function are retrieved from outside the curve fitting function. During the initial fitting process, however, these arguments are generally constructed from the observed data. The only exception to this would be if the user decided to specify the initial starting parameters for *all* subjects when calling `bfit`, as in the call

```
fit <- bfit(dat, "y", "time", curveFun = linear(intercept = 0, slope = 1),
```

however this should not be common. Following the `params` argument, any other arguments specific to the curve function could be included. Although there are none for `linear`, an example of when they might be used would be for `polynomial`, in which the degree of the polynomial to be fit would be included. Finally, there is the `...` argument, which is needed to accommodate the passing of any additional arguments from `bfit` that are not a part of the curve function. Generally, this is not needed by the users but should be included nonetheless.

2. Also included in a curve function is a second function to estimate starting parameters from the observed data. While not strictly necessary that it be included *within* the curve function, it is useful for keeping the curve function self contained; parameter estimating functions defined outside of the curve function will otherwise still be used if they exist in the users calling environment. For estimating starting parameters for a linear function we see here the function `linearPars`, taking as its arguments `dat`, `y`, and `time`. In this example, we check in case `var(y) == 0`, which causes issues for `lmer::gnls`, though in general it is a good idea to check for any other potential issues when estimating starting parameters (negative values for a logistic, for example). Importantly, this function returns a named vector, with the names of the parameters needing to match the parameter names in the formula given in (5).

3. As detailed in (1), with the argument `params = NULL`, the curve function should begin by estimating starting parameters. When different parameters are passed into `params`, this is skipped

4. This is a quick check on the result from (3). Had `linearPars` returned a `NULL` object, the curve function itself should return a `NULL` object so that it is not passed to the fitter

5. Finally, we have the most intricate part of the curve function, which is the construction of the formula object to be used by `lmer::gnls`. The first two lines of this use the

base R function `str2lang` which turns a character string into an R language object (specifically, an unevaluated expression), making the names of the outcome and time variable suitable for a formula. The next line using the base R function `bquote`. The function `quote` returns its argument exactly as it was passed as an unevaluated expression; `bquote` does the same but first substituting any of its elements wrapped in `.()`. As it is written here, this will return a formula object using `slope` and `intercept` as is, but while replacing `.(y)` and `.(time)` with the appropriate names based on the columns of `dat`. Finally, the names of the parameters are included as attributes to the formula object and the curve function concludes by returning a named list including both the formula object, as well as the named vector of parameters.

The object returned by the curve function is not limited to just providing starting parameters for observed data; the formula itself is converted by `bdots` into a function proper, capable of evaluating and bootstrapping values from that function in `bboot`. And so long as a user is able to recreate the steps provided, they should be able to construct any sort of nonlinear function to be fit to their data, even if it is not included in `bdots`.

While there is obvious utility in being able to specify *new* curves for `bdots` to fit, we describe a case here in which the flexibility of the curve function was used to recreate the `doubleGauss()` function for use with our simulated data. In short, Chapter 2 details a proposed method for fitting data in the Visual World Paradigm relying *not* on a densely sampled function in time, but rather as a collection of unordered binary observations. When the `doubleGauss` function was originally introduced to `bdots`, the empirically observed data was a relatively close match for its parametric form:

$$f(t|\theta) = \begin{cases} \exp\left(\frac{(t-\mu)^2}{-2\sigma_1^2}\right)(p - b_1) + b_1 & \text{if } t \leq \mu \\ \exp\left(\frac{(t-\mu)^2}{-2\sigma_2^2}\right)(p - b_2) + b_2 & \text{if } t > \mu \end{cases} \tag{2.8}$$

An example of how the observed data matched the proposed functional form is taken from an example given in Figure 2.10.

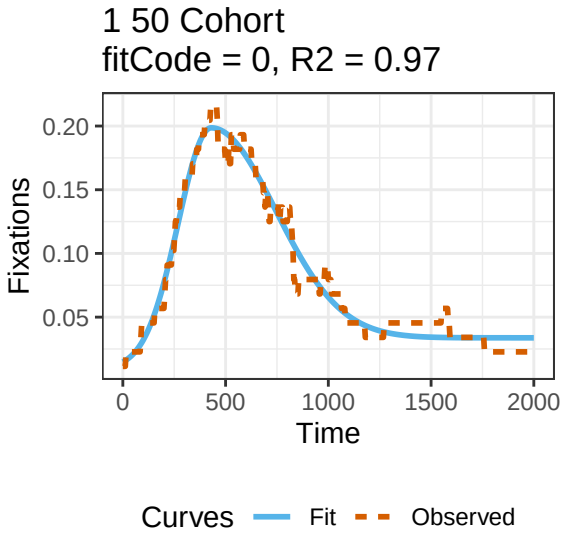


Figure 2.10: Example data for a four parameter logistic

Accordingly, an appropriate function for estimating the starting parameters took the form given in Figure 2.11.

```
dGaussPars <- function (dat, y, time) {
    time <- dat[[time]]
    y <- dat[[y]]
    if (var(y) == 0) {
        return(NULL)
    }
    mu <- time[which.max(y)]
    ht <- max(y)
    base1 <- min(y[time < mu])
    base2 <- min(y[time > mu])
    y1 <- y - base1
    y1 <- rev(y1[time <= mu])
    time1 <- rev(time[time <= mu])
    totalY1 <- sum(y1)
    sigma1 <- mu - time1[which.min(abs((pnorm(1) - pnorm(-1)) *
        totalY1 - cumsum(y1)))]
    y2 <- y - base2
    y2 <- y2[time >= mu]
    time2 <- time[time >= mu]
    totalY2 <- sum(y2)
    sigma2 <- time2[which.min(abs((pnorm(1) - pnorm(-1)) * totalY2 -
        cumsum(y2)))] - mu
    return(c(mu = mu, ht = ht, sig1 = sigma1, sig2 = sigma2,
        base1 = base1, base2 = base2))
}
```

Figure 2.11: Estimate starting parameters for empirical asymmetric Gaussian

This is appropriate, for example, when considering that the `mu` parameter is estimated by finding *when* the observed data is at its peak, while the `ht` parameter is found to be the peak itself. In the case of the look onset method, however, data consists of non-ordered binary observations $\{0, 1\}$ in time, making estimates for even these two parameters completely unreliable. An immediate consequence of this was that in a simulation of 1,000 subjects fit with asymmetric Gaussian data was conducted, less than half were able to return adequate fits from `bdots`, proving problematic for any kind of systematic analysis in assessing the proposed method.

The solution, of course, was to provide a new curve function utilizing a different interval function for estimating starting parameters. The particular interest in presenting this here

is how broad of a solution this may be to any number of problems: rather than attempting to construct parameters directly from the data (which may be misbehaved), we provide a reasonable distribution of starting parameters, drawing any number of samples from it, and retaining those which best fit the observed data:

```
dgaussPars_dist <- function(dat, y, time, startSamp = 8) {
        time <- dat[[time]]
        y <- dat[[y]]

        if (var(y) == 0) {
            return(NULL)
        }

        spars <- data.table(param = c("mu", "ht", "sig1", "sig2", "base1", "base2"),
                            mean = c(630, 0.18, 130, 250, 0.05, 0.05),
                            sd = c(77, 0.05, 30, 120, 0.015, 0.015),
                            min = c(300, 0.05, 50, 50, 0, 0),
                            max = c(1300, 0.35, 250, 400, 0.15, 0.15))

        fn <- function(p, t) {
            lhs <- (t < p[1]) * ((p[2] - p[5]) *
                       exp((t - p[1])^2/(-2 * p[3]^2)) + p[5])
            rhs <- (t >= p[1]) * ((p[2] - p[6]) *
                       exp((t - p[1])^2/(-2 * p[4]^2)) + p[6])
            lhs + rhs
        }
        npars <- vector("list", length = startSamp)
        for (i in seq_len(startSamp)) {
            maxFix <- Inf
            while (maxFix > 1) {
                npars[[i]] <- Inf
                while (any(spars[, npars[[i]] <= min | npars[[i]] >= max])) {
                  npars[[i]] <- spars[, rnorm(length(npars[[i]])) * sd + mean]
                }
                maxFix <- max(fn(npars[[i]], time))
            }
        }
        r2 <- vector("numeric", length = startSamp)
        for (i in seq_len(startSamp)) {
            yhat <- fn(npars[[i]], time)
            r2[i] <- mean((y - yhat)^2)
        }
        finalPars <- npars[[which.min(r2)]]
        names(finalPars) <- c("mu", "ht", "sig1", "sig2", "base1", "base2")
        return(finalPars)
    }
```

Figure 2.12: Using random distribution to estimate starting parameters

By utilizing an existing fitting function while substituting the method by which the starting parameters are estimated, we were able to go from recovering less than half of the simulated starting parameters successfully to well over 85%.

CHAPTER 3
LOOK ONSET METHOD

3.1 Introduction

Spoken words create analog signals that are processed by the brain in real time. That is, as spoken word unfolds, a collection of candidate words are considered until the target word is recognized. The degree to which a particular candidate word is being considered is known as activation. An important part of this process involves not only correctly identifying the word but also eliminating competitors. For example, we might consider a discrete unfolding of the word "elephant" as "el-e-phant". At the onset of "el", a listener may activate a cohort of potential resolutions such as "elephant", "electricity", or "elder", all of which may be considered competitors. With the subsequent "el-e", words consistent with the received signal, such as "elephant" and "electricity" remain active competitors, while incompatible words, such as "elder", are eliminated. Such is a rough description of this process, continuing until the ambiguity is resolved and a single word remains.

Our interest is in measuring the degree of activation of a target, relative to competitors. Activation, however, is not measured directly, and we instead rely on what can be observed with physiological behavior. And though there are a number of relevant indices (Spivey mouse trials), we concern ourselves here with eye tracking data collected in the context of the Visual World Paradigm (VWP) [14], an experimental model in which a participant's eye movements are tracked as the respond to spoken language. In a typical VWP experiment, participants are placed in the presence of visual objects (typically presented on a computer screen) and asked to select one in response to spoken language. The location of fixations are measured in real time, with the proportion of fixations towards any potential targeted aggregated across trials

[...]

Recently, researchers have begun to reexamine some of the underlying assumptions

associated with the VWP, calling into question the validity or interpretation of current methods. We present here a brief history of word recognition in the context of the VWP, along with an examination of contemporary concerns. We address some of these concerns directly, presenting an alternate method for relating eye-tracking data to lexical activation.

This section needs work but it mostly covers the gist of what I am trying to convey, namely we are about to go from history → current state of the world → proposal and comparison → results.

Visual World Paradigm

The Visual World Paradigm (VWP) was first introduced in 1995, making the initial link between the mental processes associated with language comprehension and eye movements [14]. A typical experiment in the VWP involves situating a subject in front of a "visual world", commonly a computer screen today, and asking them to identify and select an object corresponding to a spoken word. The initiation of eye movements and subsequent fixations are recorded as this process unfolds, with the location of the participants' eyes serving as a proxy for which words or images are being considered. This association was first demonstrated by comparing how the mean time to initiate an eye movement to the correct object was mediated by the presence of phonological competitors ("candy" and "candle", sharing auditory signal at word onset) and situations containing syntactic ambiguity ("Put the apple on the towel in the box" and "Put the apple *that's* on the towel in the box" in ambiguous scenarios with one or more apples). It is by comparing the trajectory of these mechanics across trials in the presence of auditory or semantic competitors that researchers have used the VWP in their investigation of spoken word recognition.

Proportion of fixation

It was against simulated TRACE data that Allopenna (1998) found a tractable way of analyzing eye tracking data. By coding the period of a fixation as a 0 or 1 for each referent and taking the average of fixations towards a referent at each time point, Allopenna was able to create a "fixation proportion" curve that largely reflected the shape and competitive

dynamics of word activation suggested by TRACE, both for the target object, as well as competitors. This also served to establish a simple linking hypothesis, specifically, "We made the general assumption that the probability of initiating an eye movement to fixate on a target object o at time t is a direct function of the probability that o is the target given the speech input and where the probability of fixating o is determined by the activation level of its lexical entry relative to the activation of other potential targets." Further of note is what this linking hypothesis does not include, namely:

1. No assumption that scanning patterns in and of themselves reveal underlying cognitive processes

2. No assumption that the fixation location at time t necessarily reveals where attention is directed (only probabilistically related to attention)

Other assumptions included here include that language processing proceeds independent of vision, and that visual objects are not automatically activated. Or, more succinctly, it assumes that fixation proportions over time provide an essentially direct index of lexical activation, whereby the probability of fixating an object increases as the likelihood that it has been referred to increases.

While other linking hypotheses have been presented (Magnuson 2019) [3], that there is *some* link between the function of fixation proportions and activation has guided the last 25 years of VWP research.

Parametric Methods and Individual Curves

While there have most certainly been advancements to the use of the VWP for speech perception and recognition (and expanded into related domains, such as sentence processing and characterizing language disorders (according to Bob)), we limit ourselves here to one in particular. In 2010, McMurray et al expanded the domain of the VWP by introducing emphasis on individual differences in participant activation curves. Two aspects of this paper are relevant here. First, although they were not the first to introduce non-linear functions to

be fit to observed data, they did introduce a number of important parametric functions in use today, namely the four (or five) parameter logistic and the double-gauss (asymmetrical gauss), the primary benefit being that the parameters of these functions are interpretable, that is, they "describe readily observable properties." Second, which I suppose was also introduced by Mirman (2008) [10] to some degree (though I have not read it yet, just pulling from Bob) is specifying individual subject curves across participants. This has been critical in that:

1. The parameters of the functions describe interpretable properties
2. This made the idea of distributions of parameters for a particular group a relevant construct

Though not stated directly (given it predates bdots by 8 years), this also served as the impetus for investigating group differences in word activation through the use of bootstrapped differences in time series [11] and the subsequent development of the `bdots` software in R for analyzing such differences. (A history of exploring differences in group curves can be found in [12]).

This brings us to the current day, where the state of things is such that VWP data is widely used to measure word recognition by collecting data on individual subjects and fitting to them non-linear parametric curves with interpretable parameters. Context in hand, we are now able to introduce some of the main characters of our story, specifically how data in the VWP is understood and used.

3.2 Analysis with VWP Data

The following section goes into more detail on the specifics

3.2.1 Anatomy of Eye Mechanics

In the context of eye tracking data and word recognition, there are a few mechanics with which we are concerned. The first of these is activation. Even with the immediacy and (fullness? some word they use to describe dense time series here being better than

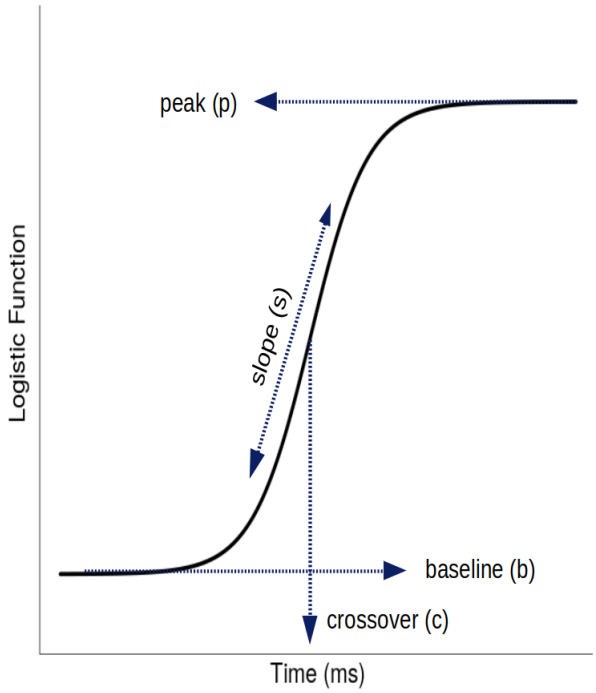


Figure 3.1: An illustration of the four-parameter logistic and its associated parameters, introduced as a parametric function for fixations to target objects in McMurray 2010. Can describe the parameters in detail, but should also have the formula itself somewhere to be referenced. (Equation 4.10)

yes/no response), what we observe with any eye movement is not a direct readout of the underlying activation. Rather, there is a period of latency between the decision to launch an eye movement and the physiological response, a period known as oculomotor delay. And finally, there are they physical mechanics of the eye movements themselves, the saccade and the fixation which, together, make up a "look". We will briefly address each of these in the reverse of the order in which they were introduced.

Saccades and fixations:

Rather than acting in a continuous sweeping motion as our perceived vision might suggest, our eyes themselves move about in a series of short, ballistic movements, followed

by brief periods of stagnation. These periods of movement and stagnation, respectively, are the saccades and fixations.

Saccades are short, ballistic movements lasting between 20ms-60ms, during which time we are effectively blind. Once in motion, saccades are unable to change trajectory from their intended destination. Following this movement is a period known as a fixation, itself made up of a necessary refraction period (during which time the eye is incapable of movement) followed by a period of voluntary fixation which may include planning time for deciding the destination of the next eye movement; the duration of fixations are typically (some length). It will be convenient to follow previous convention and consider a saccade followed by its adjacent fixation as a single concept called a "look" [9]. We take particular care here to note that the beginning of a look, or "look onset", starts the instance that a previous look ends or, said another way, the instant an eye movement is launched. A visual description of these is provided in Figure 3.2.

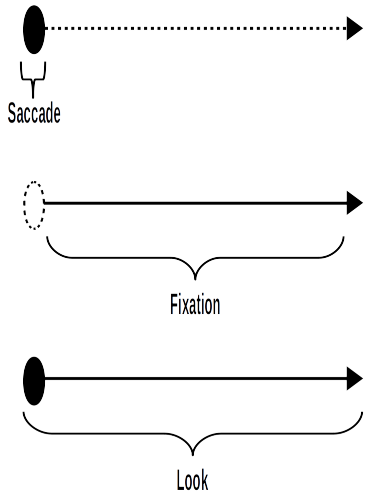


Figure 3.2: redo this image to match anataomy of look image, also for size

Oculomotor delay:

While the physiological responses are what we can measure, they are not themselves what we are interested in. Rather, we are interested in determining word activation, itself

governing the cognitive mechanism facilitating movements in the eye. Between the decision to launch an eye movement (a cognitive mechanism governed by the activation, next section) and the movement itself is a period known as oculmotor delay. It is typically estimated to take around 200ms to plan and launch an eye movement [15], and this is usually accounted for by subtracting 200ms from any observed behavior. As oculomotor delay is only "roughly" estimated to be around 200ms, we suggest that accounting for randomness will be critical in correctly recovering the the cognitive mechanism of interest or at very least in identifying possible sources of bias or error. How this phenomenon relates to saccades and fixations is demonstrated in Figure 3.3.

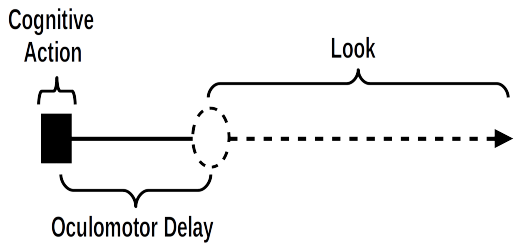


Figure 3.3: redo this image

3.2.2 Activation

The concept of activation, as it relates to the discussion here, arises from the metaphor in which word perception is made up of a network made up of hierarchical levels (letter, phoneme, word, etc.,) acting as an interactive process unfolding in time [5]. Under this *interactive activation model*, greater activation is associated with a greater excitatory action for a network node (specifically here, a word) resulting from consistency with the received auditory signal. The interactive activation model allows for both excitatory and inhibiting activations, resulting in the "competing" activation curves being modeled in the VWP.

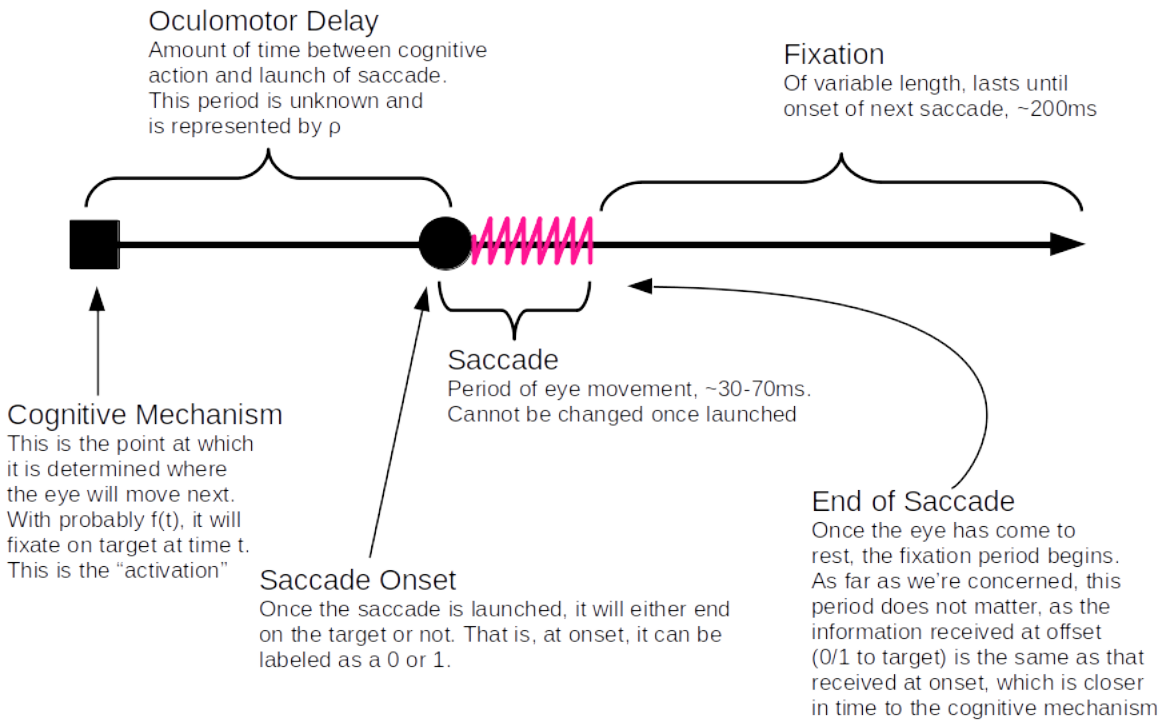


Figure 3.4: I want to do this figure again but differently. have saccade be two bars matching anatomy of look, include refractory period of fixation, noting that that and saccade are identical, followed by period of time of voluntary fixation (theoretically relevant) followed by next CM

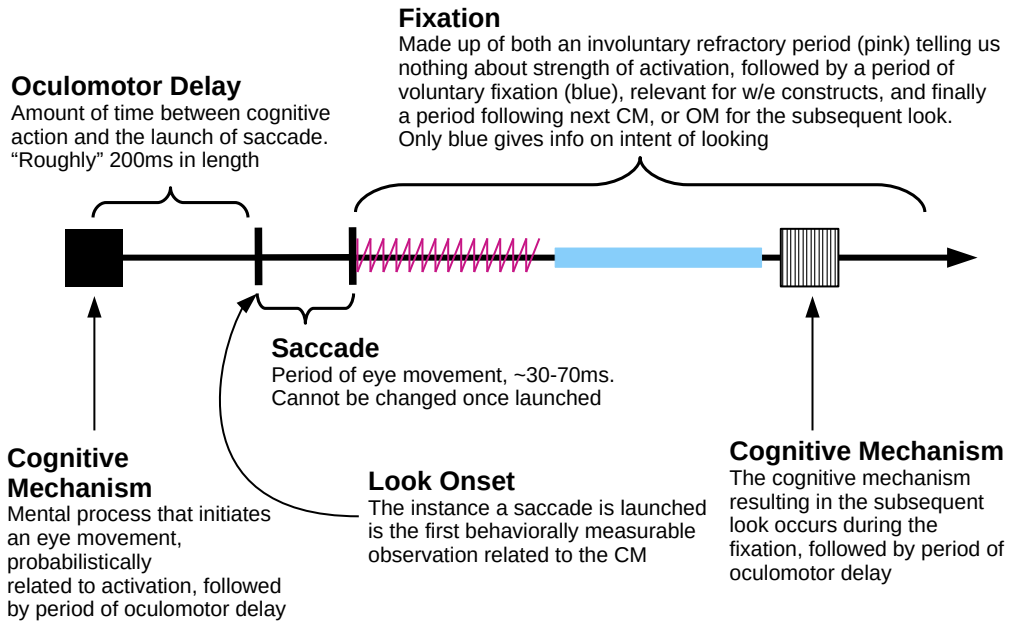


Figure 3.5: Alternative to Figure 3.4 with diagram of saccade/fixation more consistent with Figure 3.8. There may be a bit more than is needed here, especially with the pink and blue (and maybe even subsequent CM). Although I don't discuss their use in detail, the pink/blue/second CM highlight the fact that *even when* we argue that using proportion of fixation method indirectly captures strength of activation via length of duration, the assumption of linearity is incorrect. For the sake of argument, assuming that the refractory period is exactly 100ms and the length of oculomotor delay is 200ms, a fixation of length 500ms may seem to be 25% larger than a fixation lasting 400ms, but in terms of the *intentional* fixation period, it is twice as large. This serves both to demonstrate another issue with the proportion method while also paving the way for incorporating fixation length in the future. Still, this may not all be necessary here, or presented in a cleaner way

[maybe also address happens continuously, hence the continuous mapping model of lexical activation]

Here tie in idea of activation, though need to be more concise about what we mean.

While a number of experimental methods are used as real-time indices of lexical access ([13], others), we concern ourselves here with the use of eyetracking as it relates to activation as first suggested by [1]. Whereas the initial treatment of eyetracking data made no attempt identify or model subject-specific trends, more recent work has made strides in making subject analysis more tractable. Specifically, we adopt the idea that each participant's results can be fit to non-linear functions who's parameters describe clinically relevant properties [8]. We will denote this activation function f with parameters θ as a function in time, giving $f(t|\theta)$

For example, the four parameter logistic function in Figure 3.1 is often used to model fixations to the Target object in the VWP with functional form

$$f(t|\theta) = \frac{p-b}{1+\exp\left(\frac{4s}{\mathrm{p}-b}(x-t)\right)} + b. \tag{3.1}$$

Similarly, a six parameter asymmetric Gaussian function, often used for fixations to competitors, is given as

$$f(t|\theta) = \begin{cases} \exp\left(\frac{(t-\mu)^2}{-2\sigma_1^2}\right)(p-b_1)+b_1 & \text{if } t \leq \mu \\ \exp\left(\frac{(t-\mu)^2}{-2\sigma_2^2}\right)(p-b_2)+b_2 & \text{if } t > \mu \end{cases} \tag{3.2}$$

(I didn't make a nice graph/label for this).

While both functions are commonly used in the VWP for modeling eye fixations, for simplicity we will limit the primary focus of our discussion to fixations to the Target with the four parameter logistic, though ultimately our argument is agnostic to the modeling function used, parametric or otherwise. Discussion related to the asymmetric Gaussian is treated in

the appendix.

3.2.3 VWP data

We now consider how the aforementioned mechanics relate to the visual world paradigm. In a typical instantiation of the VWP, a participant is asked to complete a series of trials, during each of which they are presented with a number of competing images on screen (typically four). A verbal cue is given, and the participants are asked to select the image corresponding to the spoken word. All the while, participants are wearing (generally) a head-mounted eye tracking system recording where on screen they were fixated.

An individual trial of the VWP may be short, lasting anywhere from 1000ms to 2500ms before the correct image is selected. Prior to selecting the correct image, the participant's eyes scan the environment, considering images as potential candidates to the spoken word. As this process unfolds, a snapshot of the eye is taken at a series of discrete steps (typically every 4ms) indicating where on the screen the participant is fixated. A single trial of the VWP typically contains no more than four to eight total "looks" before the correct image is clicked, resulting in a paucity of data in any given trial.

[relevant quote]["We find that eye movements to objects in the workspace are closely time-locked to referring expression s in the unfolding speech stream, providing a sensitive and nondistruptive measure of spoken language comprehension during continuous speech" [1]]

To be clear, eye trackers themselves only record x and y coordinates of the eye at any given time, and it is only after the fact that "psychophysical" attributes are mapped onto the data (saccades, fixations, blinks, etc.,). We adopt the strategy of prior work in discussing eye tracking data in terms of their physiological mapping, as this will be crucial in constructing a physiologically relevant understanding of the problem at hand [9].

["Default interpretation is that greater fixation proportions indicate greater activation in the underlying processing system" [3]]

To create a visual summary of this process aggregated over all of the trials, a la Allopena, a "proportion of fixations" curve is created, aggregating at each discrete time point the average of indicators of whether or not a participant is fixated on a particular image. A resulting curve is created for each of the competing categories (target, cohort, rhyme, unrelated), creating an empirical estimate of the activation curve, $f(t|\theta)$. See Figure 3.6. For any subject $i = 1, \ldots, n$, across times $t = 0, \ldots, T$ and trials $j = 1, \ldots, J$, a construction of this curves can be expressed as:

$$y_{it} = \frac{1}{J} \sum z_{ijt} \tag{3.3}$$

where z_{ijt} is an indicator $\{0, 1\}$ towards a particular object in trial j at time t and such that we have an empirical estimate of the activation curve,

$$f(t|\theta_i) \equiv y_{it}. \tag{3.4}$$

For our discussions here, we will call this the proportion of fixation method (though we not that in actuality it is the proportion of *trials* in which a fixation occurs at each time point).

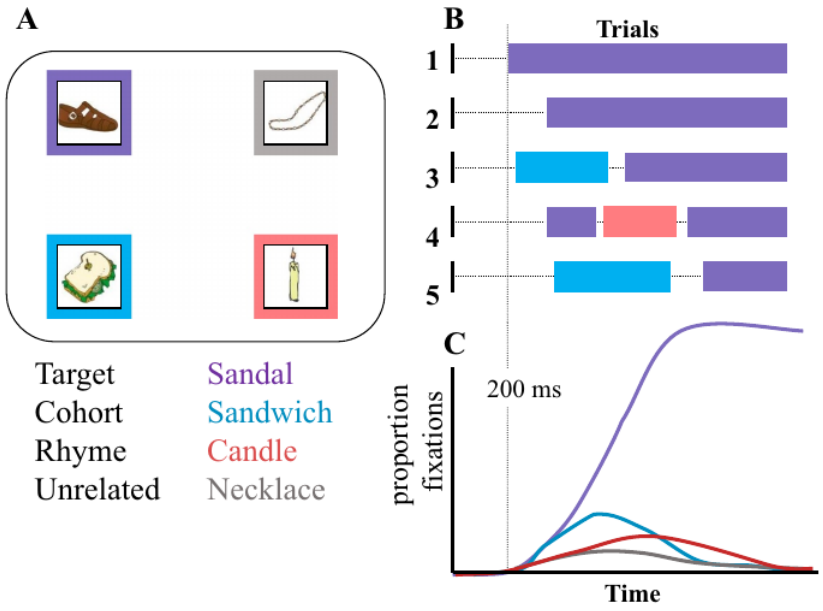


Figure 3.6: Stole this from Bob (who apparently stole it from richard aslin), plan on making my own

As each individual trial is only made up of a few ballistic movements, the aggregation across trials allows for these otherwise discrete measurements to more closely represent a continuous curve. Curve fitting methods, such as those employed by `bdots`, are then used to construct estimates of function parameters fitted to this curve. Figure 3.6 provides an illustration.

3.3 Bias/Look Onset (better name for section)

Having given due consideration to the state of things are they are, we find ourselves in a time of moral reflection, reexamining the underlying relationship between lexical activation (the mechanism of interest) and the physiological behavior we are able to observe (here, specifically eye-tracking). This is referred to in the literature as the linking hypothesis. And while there are a number of competing hypothesis, they each share a collection of implicit assumptions relating what is observed to what is being studied.

The simplest version of a linking hypothesis in the context of the VWP is the "general

assumption that the probability of initiating an eye movement to fixate on a target object o at time t is a direct function of the probability that o is the target given the speech input and where the probability of fixating o is determined by the activation level of its lexical entry relative to the activations of other potential targets (i.e., the other visible objects)" [1]. It is from this assumption that we justify the relation in Equation 3.4. To a degree, this assumption is shared by most linking hypothesis in that the probabilistic nature of the proportions of fixations is assumed to be related in time to the strength of the underlying activation. Primary differences in linking hypotheses tend to revolve around the particulars of the mechanics involved, including the duration of fixations, eye scanning behavior, the impacts of priming, or the relation between visual processing acting in conjunction with lexical activation. We make no statement as to the merits of each, though see [3] for a review.

We consider a particular meta contribution to this debate presented by McMurray in which he probed the relationship between the observed dynamics in the fixations and the underlying dynamics of activation under a variety of assumptions [6]. In short, he showed that curves reconstructed using the standard proportion of fixations analysis in the VWP were poor estimates of the underlying system, with the magnitude of bias increasing with the complexity of the mechanisms involved. Though this made few claims as to what the underlying mechanics may be, it did demonstrate the inherent difficulty in relating observable behavior to the underlying cognitive process.

An important contribution made there, however, and one that we adopt here is an explicit definition of the underlying activation function. Given the relation in Equation 3.4, it is reasonable to assume that the underlying activation of any of the objects with the VWP could be modeled with a nonlinear function $f(t|\theta)$. The goal of a VWP analysis, then, is the recovery of this underlying activation function.

From this assumption we propose an alternative model of the relation between the

underlying activation and the observed behavior, with a careful delineation of the psychophysical components of a look in conjunction with its generating behavior. In particular, we consider the cognitive mechanism associated with initiating an eye movement, which is probabilistically associated with lexical activation, the delay between this and the onset of its associated look, and finally how the different components of the look are related to fundamentally different mechanisms. From this and what we ultimately argue is that observed bias in the recovery of the activation curve under the proportion of fixations method can be partitioned into two distinct components:

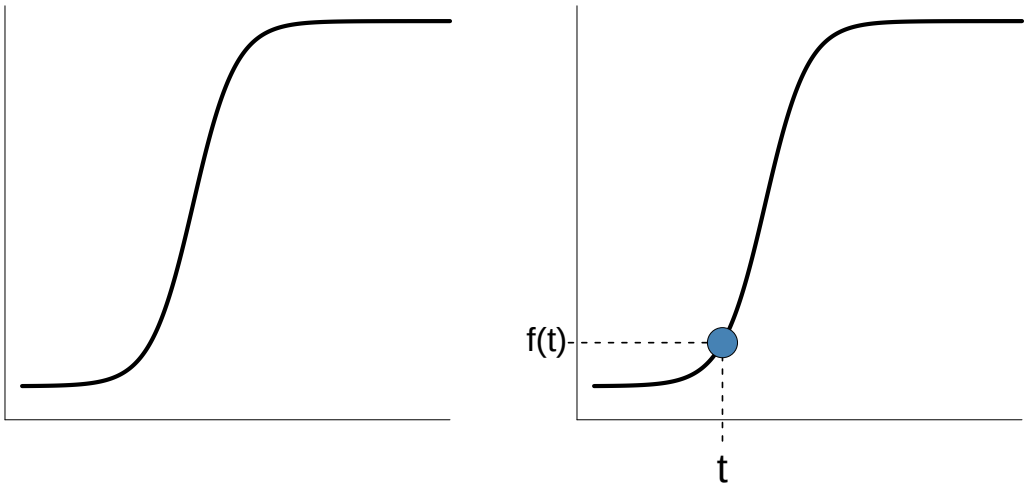
[i would like to maybe go into more detail here or have a picture]

1. The first source of bias, which is the primary emphasis of my proposal, is what I call the "added observation" bias. This involves the fact that in a standard analysis of VWP data, the entire duration of a fixation is indicated with a $\{0, 1\}$ at any time, t, without having observed any behavior associated with the initiation of an eye movement at that time. In other words, by using the entire fixation, we are both obscuring data relevant to the mechanism of interest (onset) while also conflating it with data generated by a fundamentally different mechanism [unpack this a bit more here].

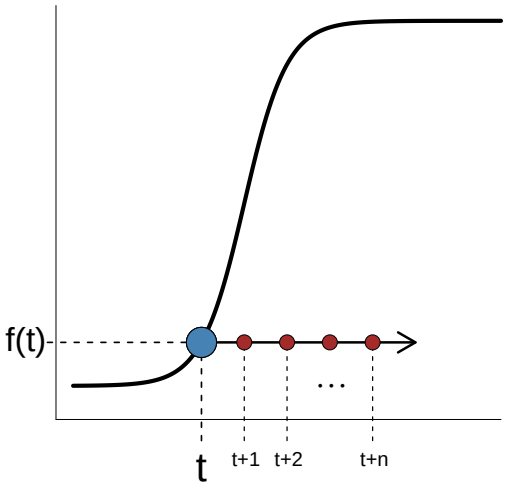
2. The second source of bias is "delayed observation bias". This bias arises from the fact that an eye movement launched at some time t was planned at some time prior. This is primarily a consequence of the oculomotor delay

The first source of bias, the "added observation" bias, arises singularly from the fact that the destination of a look, which is observed at look onset, has a fundamentally *different* generating mechanism than what determines the duration of a look, never minding such mechanics as the duration of a saccade or the refractory period of a fixation. Nonetheless, a standard analysis of VWP data does not differentiate between the initial onset and the period of subsequent fixation: both are recorded as either 0 or 1 according to it's location. A look onset at time t is probabilistically determined by by its lexical activation $f(t|\theta)$ whereas the

period of fixation is governed by a separate mechanism altogether. Treating the subsequent fixation as indistinguishable has the effect of not only "adding" observations to the data, but adding observations that necessarily biased. The result is a distorted estimation of the underlying activation. A depiction of this phenomenon is given in Figure 3.7.



(a) (b)



(c)

Figure 3.7: **(a.)** Example of a nonlinear activation curve $f(t|\theta)$ **(b.)** At some time, t, a saccade is launched with its destination probabilistically determined by $f(t|\theta)$ **(c.)** For a look persisting over n time points, $t+1, \ldots, t+n$, we are recording "observed" data, adding to the proportion of fixations at each time but without having gathered any additional observed data at $f(t+1|\theta), \ldots, f(t+n|\theta)$, thus inflating (or in the case of a monotonically increasing function like the logistic, deflating) the true probability.

The second source of bias is the "delayed observation" bias. It is well established in the literature that the time it takes to plan and launch a saccade is around 200ms [15],

which is typically accounted for by subtracting 200ms from the observed data. There are two aspects of this that are worth considering further. First, if the mean duration of this oculomotor delay is not 200ms, bias will be observed as the difference between the true time and the 200ms adjustment. And although not bias in the technical sense, there has been no accounting for what effect randomness in this delay has on the error in the recovery of the underlying activation. It will be worthwhile in investigating this as the potential magnitude will determine if this delay is worth considering in any more detail in future research.

—

While we present no immediate solution to the effects of randomness in the delayed observation bias, we argue that the added observation bias can be rectified by using *only* the times observed with look onset in the recovery of the underlying dynamics. We call this the the "look onset" method, which we explain in more detail.

We argue further that the added observation bias can be rectified by using *only* the times observed with the look onset in the recovery of the underlying dynamics. We call this the "look onset" method in contrast to the "proportion of fixation" method. And while we do not present an immediate solution to the problem of the delayed observation bias, we do demonstrate the effects on estimation error in the presence of oculomotor delay even when the correct mean is accounted for.

Look Onset Method:

The look onset method differ in the proportion of fixation method only in determining which observed data should be considered relevant in the estimation of lexical activation. A particularly compelling argument to made in favor of the look onset method, a corollary of the added observation bias, is that it has a readily defensible mathematical description. A saccade launched at time t (marking the onset of a look) is assumed to be probabilistically determined by its lexical activation (relative to competitors) at time t, giving us

$$s_t \sim Bin[f(t|\theta)] \tag{3.5}$$

(it may be that l_t for look onset is better notation, but my concern is that it doesn't capture the "onset" nature that we are concerned with and may instead suggest the entire saccade + fixation).

The utility of this is evident when tasked with stating the distribution of y_t in Equation 3.3 as it relates to $f(t|\theta)$. And where given the overlap of fixations within a particular trial, it is unclear what relation y_t may have to y_{t+1}.

[maybe statement along the lines of we make no assumptions as to the processing of visual stimuli, with probabilities of fixations independent of previous fixations? consistent with other "bare bones" linking hypotheses]

Two further comments are made about this method here. First, in anticipation of the observation that the look onset method discards relevant information regarding the strength of activation by not implicitly including the length of a fixation (source), we acknowledge this and reserve further comment for the discussion. Second, given the difference in structure of the observed data, we confirm that the current iteration of `bdots` is capable of fitting nonlinear curves to data both under the proportion of fixation and look onset methods, removing any technical difficulties in the adoption of this method.

3.4 Simulations

Simulations were conducted to replicate the mechanics of a look combined with oculomotor delay as detailed in Figure 3.8. This section only address Target fixations with a four parameter logistic as given in Equation 4.10; simulations according to looks to competitors is treated in the appendix. We will begin by describing the process of simulating a single subject.

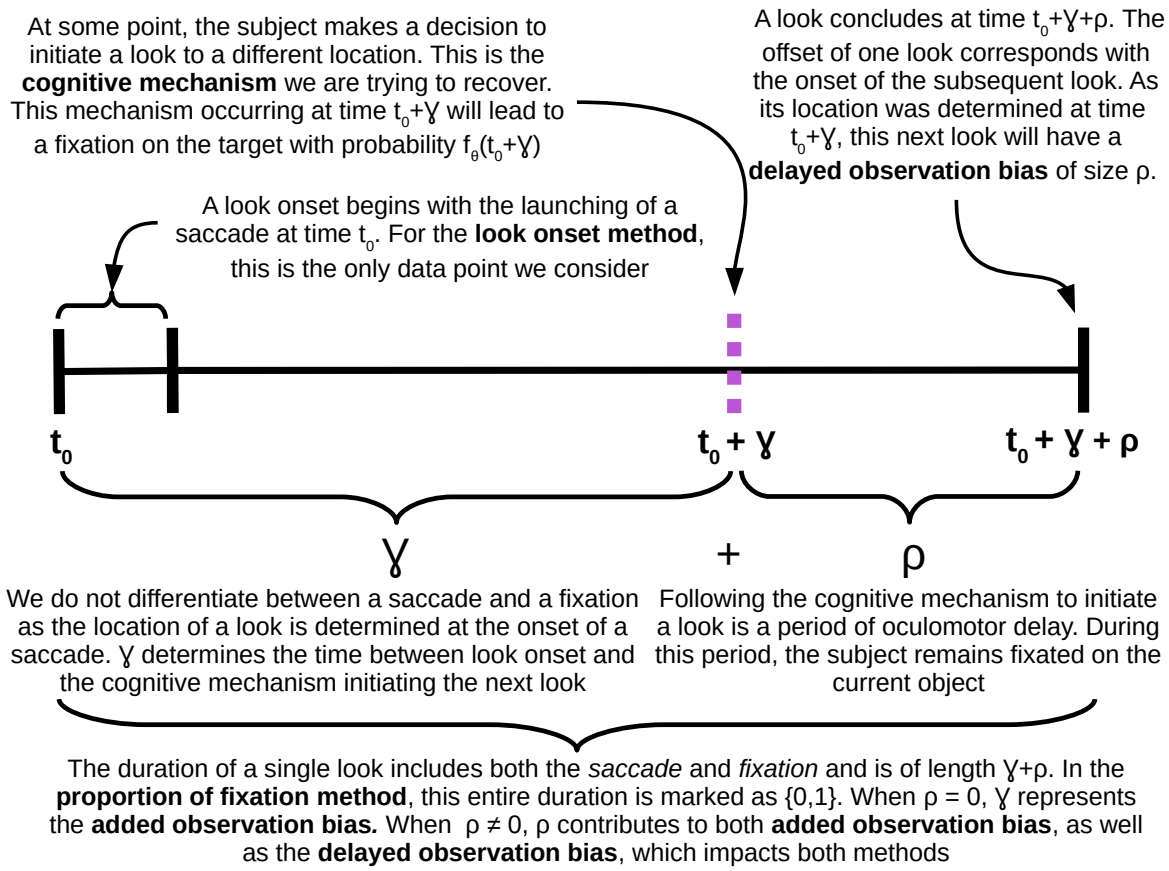


Figure 3.8: Anatomy of a look – a key thing to discuss somewhere is the OM delay, refractory period, and planning time. The latter two go in γ. Worth noting also that while we do need to be able to control for ρ, *information* regarding strength of consideration will be in γ less the refractory period

First, each subject randomly draws a set of parameters θ_i from an empirically determined distribution based on normal hearing participants in the VWP [2] to construct a subject specific generating curve, $f(\cdot|\theta_i)$, where f here is assumed to be the logistic given in Equation 4.10. It is according to this function that the decision to initiate a look at time t will subsequently direct itself to the Target with probability $f(t|\theta_i)$. We then go about simulating trials according to the following method: At some time t_0, a subject initiates a look. This look persists for at least a duration of γ, drawn from a gamma distribution with mean and standard deviation independent of time and previous fixations. At time $t_0+\gamma$, the

subject determines the location of its next look, with the next look being directed towards the target with probability $f(t+\gamma|\theta_i)$. The decision to initiate a look is followed by a period of oculomotor delay, ρ, during which time the subject remains fixated in the current location. Finally, at time $t_0 + \gamma + \rho$, the subject ends the look initiated at t_0 and immediately begins its next look to the location determined at time $t_0 + \gamma$. For the look onset method, the only data recorded are the times of a look onset and their location: in this case, at times t_0 and $t_0 + \gamma + \rho$. By contrast, the proportion of fixation method records the object of fixation at 4ms intervals for the entire period of length $\gamma + \rho$. A single trial begins at $t = 0$ and continues constructing looks as described until the total duration of looks exceeds 2000ms. Each subject undergoes 300 trials, and 1,000 subjects are included in each simulation.

Three total simulations were performed to investigate the biases identified in the previous section, each differing only in the random distribution of the oculomotor delay parameter, ρ. In the first simulation, we set $\rho = 0$ to remove any oculomotor delay. In this scenario, a look initiated at time t by subject i will be directed towards the target with probability $f(t|\theta_i)$. Doing so removes any potential bias from delayed observation and allows us to identify the effects of the added observation bias in isolation. In the remaining simulations we probe the effects of randomness in oculomotor delay, investigating what effect uncertainty may have in our recovery of the generating function. We do this assigning ρ to follow either a normal or Weibull distribution, each with a mean value of 200ms. As is standard in a VWP analysis, we subtracted 200ms from each observed point prior to fitting the data. A consequence of this is that in these simulations, the bias itself is accurately accounted for by subtracting the correct mean, with the resulting error in the curve fitting process the result of the inherent variability. This does not detract from the argument being made, however, and any true bias in the mean of the oculomotor delay would asymptotically result in a horizontal shift of the observed data according to the direction and magnitude of the bias.

The simulations are performed in R, with the simulation code available on the author's Github page (link?). Simulated data was fit to the four parameter logistic function using

`bdots v2.0.0`.

As all of the data could not be individually inspected prior to being included in the analysis, subjects were excluded from consideration if fitted parameters from either the look onset method or the proportion of fixation method resulted in a peak less than the slope, or if the crossover or slope were negative. In the settings in which there was no delay, normally distributed delay, or Weibull distributed delay, 981, 973, and 981 subjects were retained, respectively.

[In all of the histograms I removed the top and bottom 1% of observations, true outliers which obscured the ranges of the histograms, relevant particularly in comparing proportion method par bias with and without delay]

3.4.1 No Delay

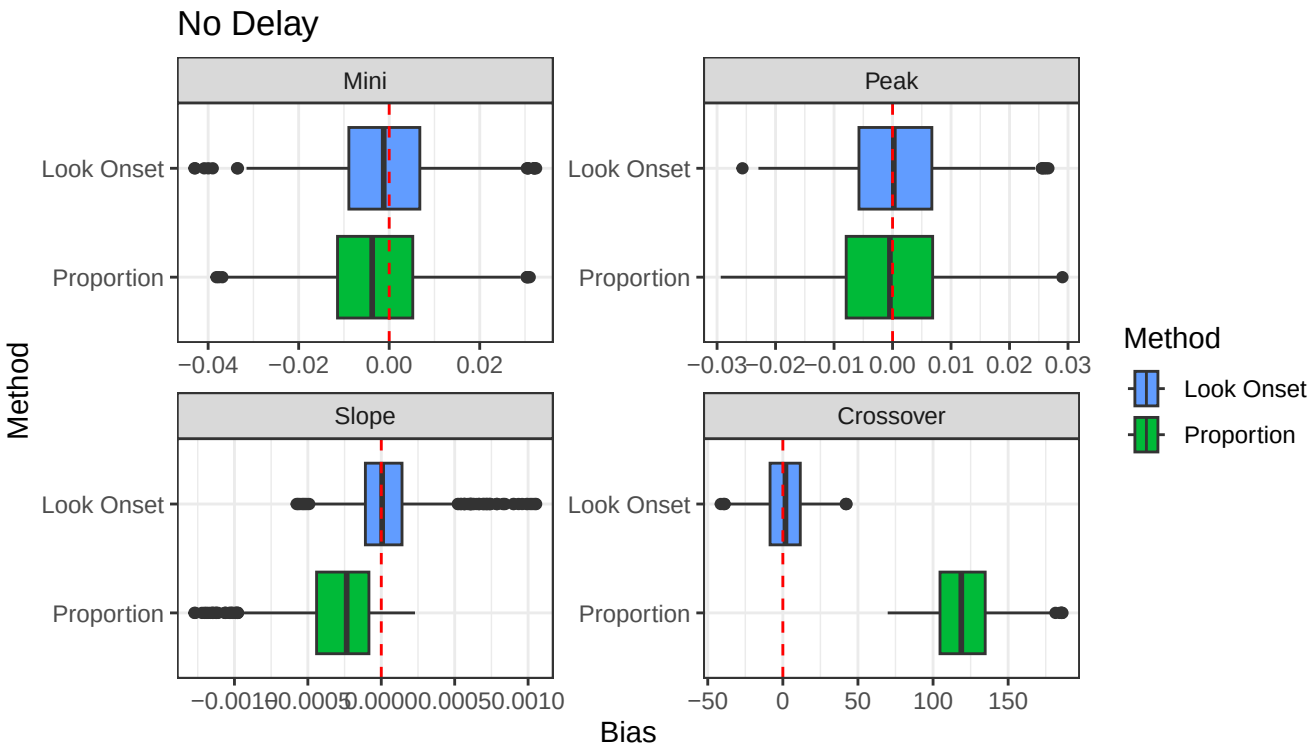


Figure 3.9: Parameter bias with no oculomotor delay

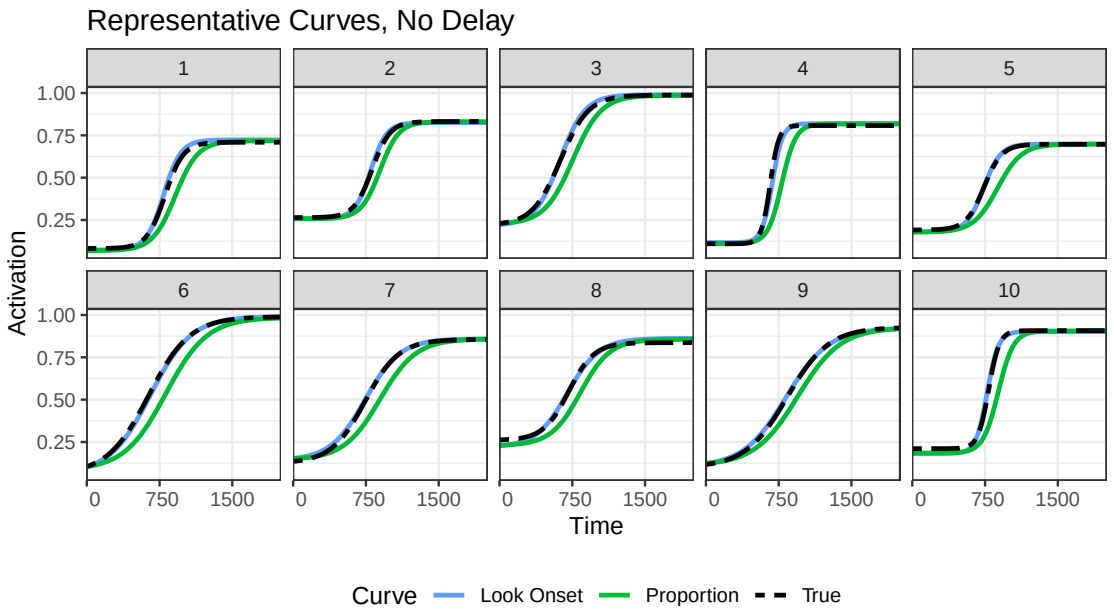


Figure 3.10: Representative curves with no oculomotor delay

3.4.2 Normal Delay

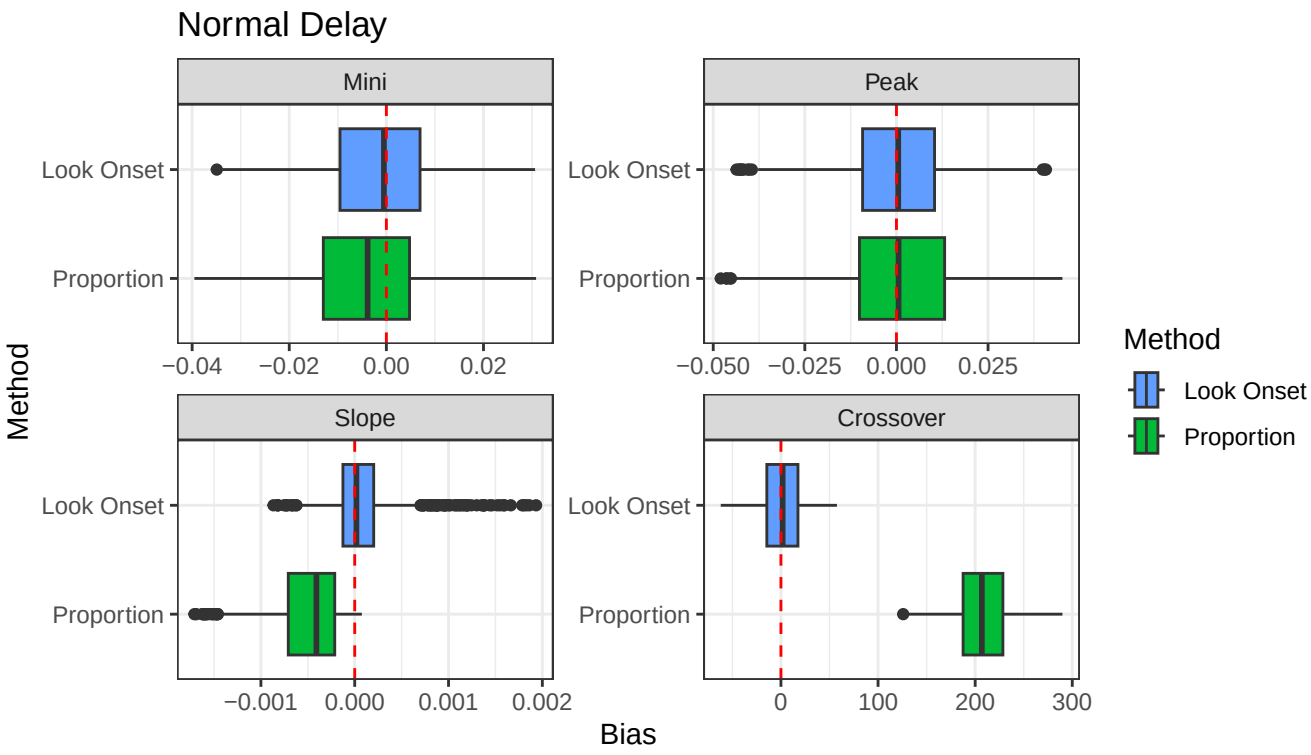


Figure 3.11: Parameter bias with normal oculomotor delay

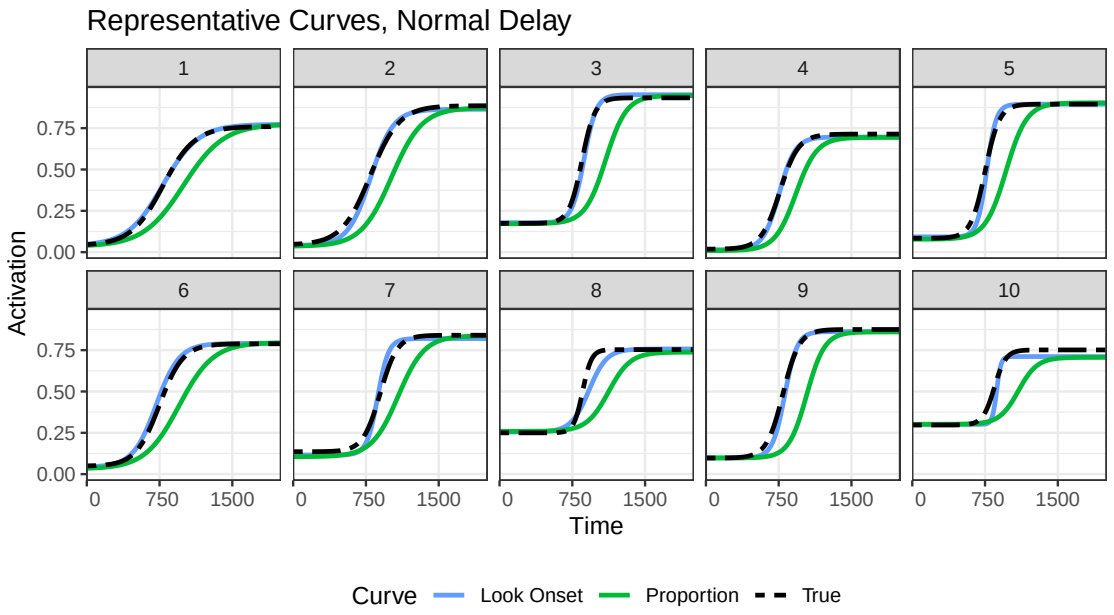


Figure 3.12: Representative curves with normal oculomotor delay

3.4.3 Weibull Delay

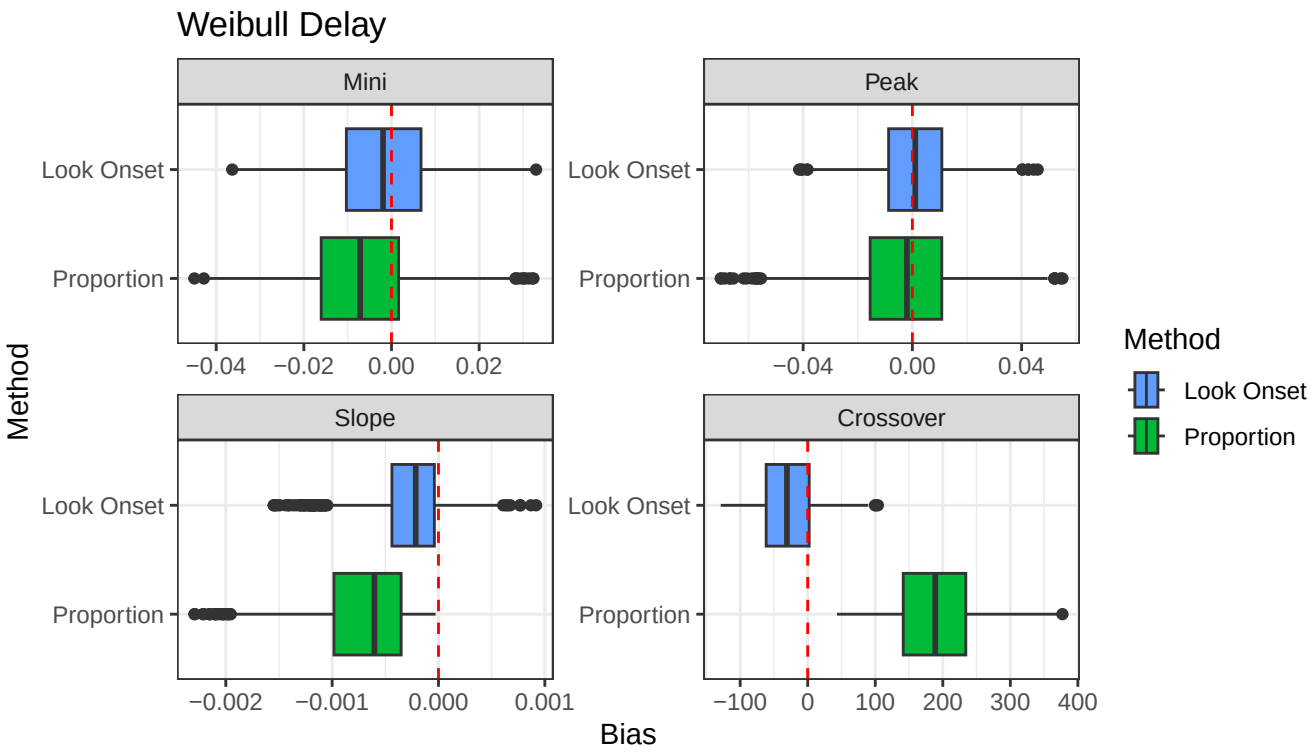


Figure 3.13: Parameter bias with Weibull oculomotor delay

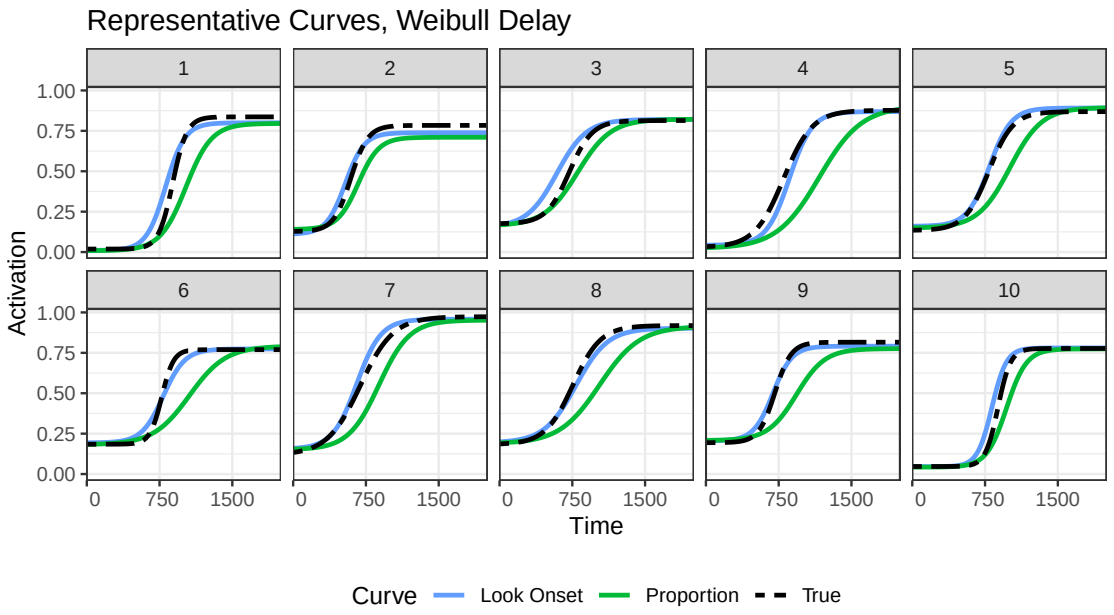


Figure 3.14: Representative curves with Weibull oculomotor delay

3.4.4 Results

[I will go back and comment on results in each of the scenarios with broader discussion of implications in each, though normal and Weibull will be similar]

In addition to the visual summaries, we present in Table 3.1 a summary of the mean integrated squared error (MISE) between the generating and recovered curves using each of the methods

We begin by noting the magnitude of difference between the look onset method and the proportion of fixation method in the case of $\rho = 0$, or No Delay, demonstrating the amount of bias introduced in the proportion method. This alone demonstrates how critical of an issue the added observation bias is in the recovery of the underlying activation.

To assess the effects of randomness in the oculomotor delay, it seems prudent to limit the comparisons within each method. Considering first the look onset method, we see that as the degree of variability increases, so does the difficulty in correctly recovering the underlying curve. It is important to note that these magnitudes are meant to be relative rather

than absolute: the particular values observed are a function of the relationship between the generating γ distribution and that of ρ. Nonetheless, this does suggest a need to further investigate ways to control for the added uncertainty. To quickly comment on the apparently "flipped" MISE for the proportion of fixation method as it relates to the normal and Weibull distributed oculomotor delay, it would seem as if the skew of the Weibull distribution acted in such a way as to actually offset some of the observed added observation bias and seems more an artifact of the simulation conditions rather than an inherent statement relating OM bias to the proportion of fixation method in general.

Method	Delay	1st Qu.	Median	3rd Qu.
Look Onset	No Delay	0.17	0.32	0.56
Look Onset	Normal Delay	0.37	0.71	1.24
Look Onset	Weibull Delay	1.05	2.16	4.23
Proportion	No Delay	8.21	11.33	16.01
Proportion	Normal Delay	22.90	30.65	39.37
Proportion	Weibull Delay	15.27	24.75	38.14

Table 3.1: Summary of mean integrated squared error across simulations

The proportion of fixation method seems no longer tenable in the recovery the underlying lexical activation, given both the magnitude of differences presented in Table 3.1 as well as the theoretical arguments made and illustrated in Figure 3.7. And given that data collected via the look onset method can be adequately fit in the newest version of the `bdots` software, there appears to be little to argue against its adoption.

[this next part maybe belongs elsewhere, mainly exists to demonstrate that despite appearances, OM delay still worth further investigation]

Outside of a demonstration of its existence and potential consequence, little more has been said about addressing the delayed observation bias. Further, the consequences of the

delayed observation (under the assumption that the mean value is correctly accounted for) seem almost trivial in comparison to the differences between it and the added observation bias. That being said, we believe there are still critical reasons for considering its significance.

As mentioned earlier, the particular values observed in these simulations are both a function of the relationship between the distribution generating γ and that of ρ. However, they are also a function of the generating function itself. In particular, we draw attention to the degree of total variation f over the interval $[a, b]$, defined as

$$V(f) = \sup_{\mathcal{P}} \sum_{i=0}^{n_p-1} |f(t_{i+1} - f(t_i)| \,, \tag{3.6}$$

where $\mathcal{P} = \{P = \{t_0, \ldots, t_{n_p}\}\}$ is the set of all possible partitions of $[a, b]$. Despite appearances, this is a relatively straightforward metric in the case of monotone functions such as the logistic, where the total variation is simply $|f(b) - f(a)|$. To illustrate the relevance of this, consider a hypothetical situation in which the underlying activation we are wishing to recover is a constant function, $f(t) = c$, where the probability of fixating on a target is independent of time. In such a situation, a delayed observation would be of no issue; despite changes in time t, the probability c remains unchanged. In contrast, consider a second hypothetical situation in which activation is defined exponentially, $f(t) = \exp(t)$. In this case, the impact of delayed observation depends drastically on time, when the delay in observation in the range of small values of t have a drastically smaller impact than delayed observations when t is large ($\exp(1) - \exp(0) = 1.7183$ while $\exp(11) - \exp(10) = 37848$, despite both cases having $\Delta t = 1$).

In short, these hypothetical situations detail how the magnitude of total variation can have differential effects on the delay in observation. Now consider again the logistic function in Figure 3.1 and imagine its domain partitioned into three equally sized portions. Both the first and third, near the asymptotes, have relatively low total variation, resulting in a relatively benign effect from oculomotor delay. In contrast, the middle third contains

nearly all of the variation of the function, indicating the delayed observation here will have a disproportionate impact on the successful recovery of the function. Given the clinical relevance of the both the slope and crossover parameters, as well as acknowledging the impact that these have on the overall shape of the function, we demonstrate a need in a accounting for this delay precisely where it will impact function recovery the most. This, of course, is not unique to the logistic, with the effects of the delayed observation bias compounded in the asymmetric Gaussian functions (appendix) which has a more complicated variation structure and, accordingly, more difficulty in recovering the generating curves.

3.5 Discussion

This section needs to be tightened quite a bit.

Through our investigation, we have presented a physiologically grounded model relating eye tracking data to underlying lexical access by placing emphasis singularly on the first instance of a look, discarding information on the rest of the look entirely.

Of course, we know that longer fixations are relevant to the strength of consideration. One of the primary benefits of the proportion method is that it indirectly captures the duration of fixations, with longer times being associated with stronger activation. This is important when differentiating fixations associated with searching patterns (i.e., what images exist on screen?) against those associated with consideration (is this the image I've just heard?). However, implicit in the proportion of fixations method is a crucially overlooked assumption of a linear relationship between the fixation length and the activation. That is, insofar as the construction of the fixation curve is considered, a fixation persisting at 20ms after look onset (and well within the refraction period in which no new information regarding the cognitive mechanism or voluntary fixation could be obtained (see figure i haven't made yet)) is considered identical to a fixation persisting at 500ms after onset: both are undifferentiated in being recorded as either a 0 or 1. More likely it seems this would be more of an exponential relationship, with longer fixations offering increasingly more evidence

of lexical activation. By separating the moment of onset from the look itself, we free ourselves to construct far more nuanced models. Put more succinctly, it is not that the look onset method considers the length of fixation irrelevant; rather, it makes no statement about it at all. How these mechanisms could be combined should be the focus of future research.

Another utility that comes to mind when considering the look onset method is how much sparser of a dataset is created. Consider, for example, fitting a mixed model to the four parameter logistic modeling individual trial data for each subject all together. Such models with proportion of fixation data typically infeasible with cumbersome autocorrelative structures found within a trial (from Bob's comment, need to find source). The present method, being a much sparser dataset, may be far more computationally tractable.

The arguments presented here has hoped to satisfy two goals, agnostic to the linking hypothesis or functions ultimately decided upon. Foremost is the recognition that the mechanisms governing a look onset and the looks themselves follow separate mechanisms, and treating them as such requires fewer assumptions while still retaining the utility in fitting the same nonlinear curves to the observed data.

Second to this, we have put a name to two important sources of potential bias in the recovering of activation curves, generalizable beyond the particulars of the presented simulations. The first is the added observation bias, speaking again to the utility in separating the relevant mechanisms making up a look. And second is the delayed observation bias, bringing into focus the importance of reevaluating how we address bias and variability in the oculomotor delay.

In short, what we have hoped to accomplish here is not to drastically change the original assumptions presented in Allopenna (1996, but rather to qualify them in statistically sound ways. And really, that is pretty much it. Concluding sentence.

As a not really conclusion, I am sometimes left to wonder to what degree the proportion of fixation method was a "local minimum" in the pursuit of utilizing eye-tracking data. The proportion of fixations created an ostensible curve, prompting McMurray to establish

theoretically grounded non-linear functions to model them. These, in turn, where shown to be suitable functions with which to model saccade data over a period of trials. Had saccades lent themselves so naturally to visualization as the proportion of fixations, perhaps that is where we may have started.

3.6 Appendices

Omitting relation of this method to TRACE as it seems mostly unnecessary to the present argument

Double Gauss Simulations

Collection of simulations performed with the double gauss, currently without further comment. Number of fits retained is 868, 833, and 866 for the no delay, normal distribution, and weibull, respectively.

3.6.1 No Delay

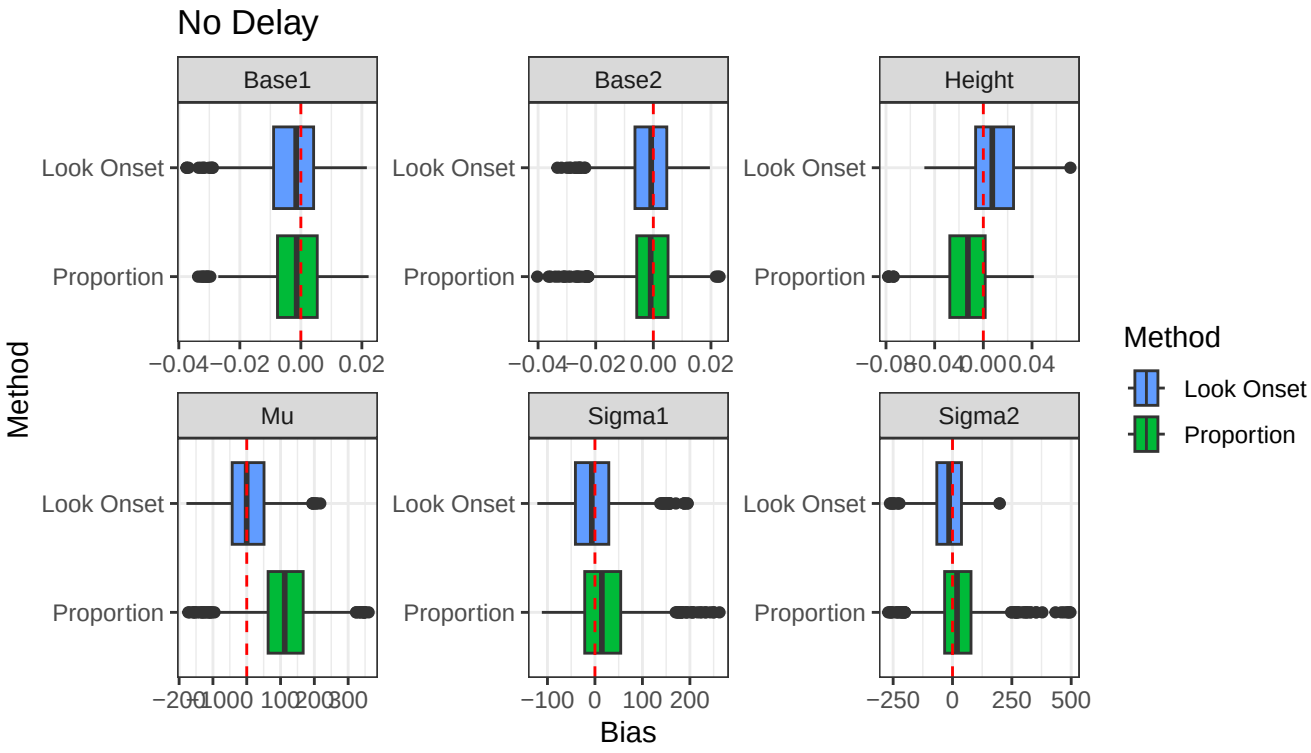


Figure 3.15: Parameter bias with no oculmotor delay

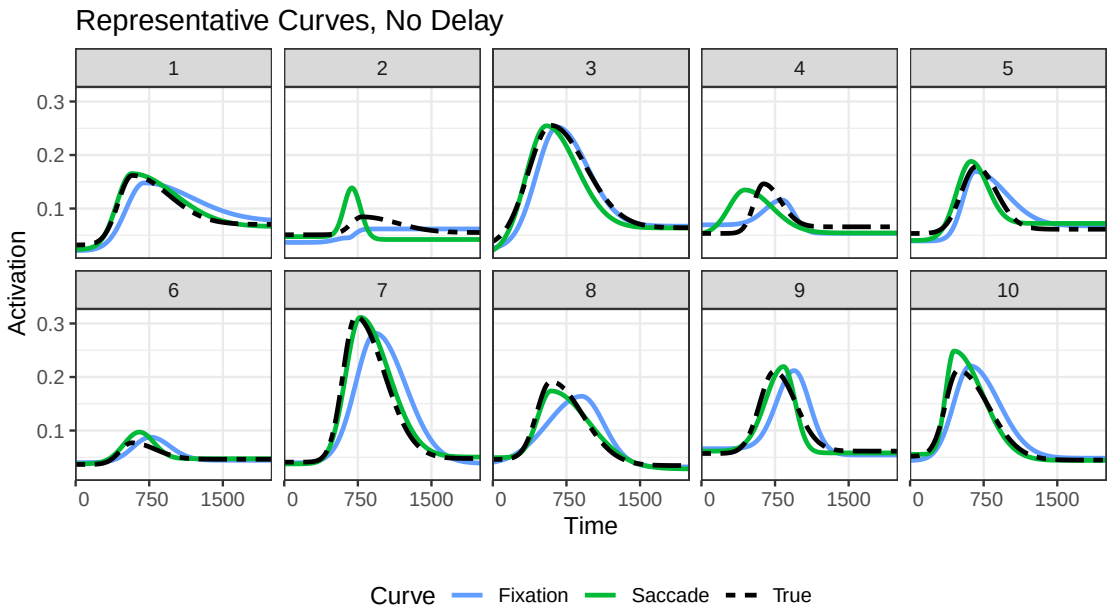


Figure 3.16: Representative curves for no oculmotor delay

3.6.2 Normal Delay

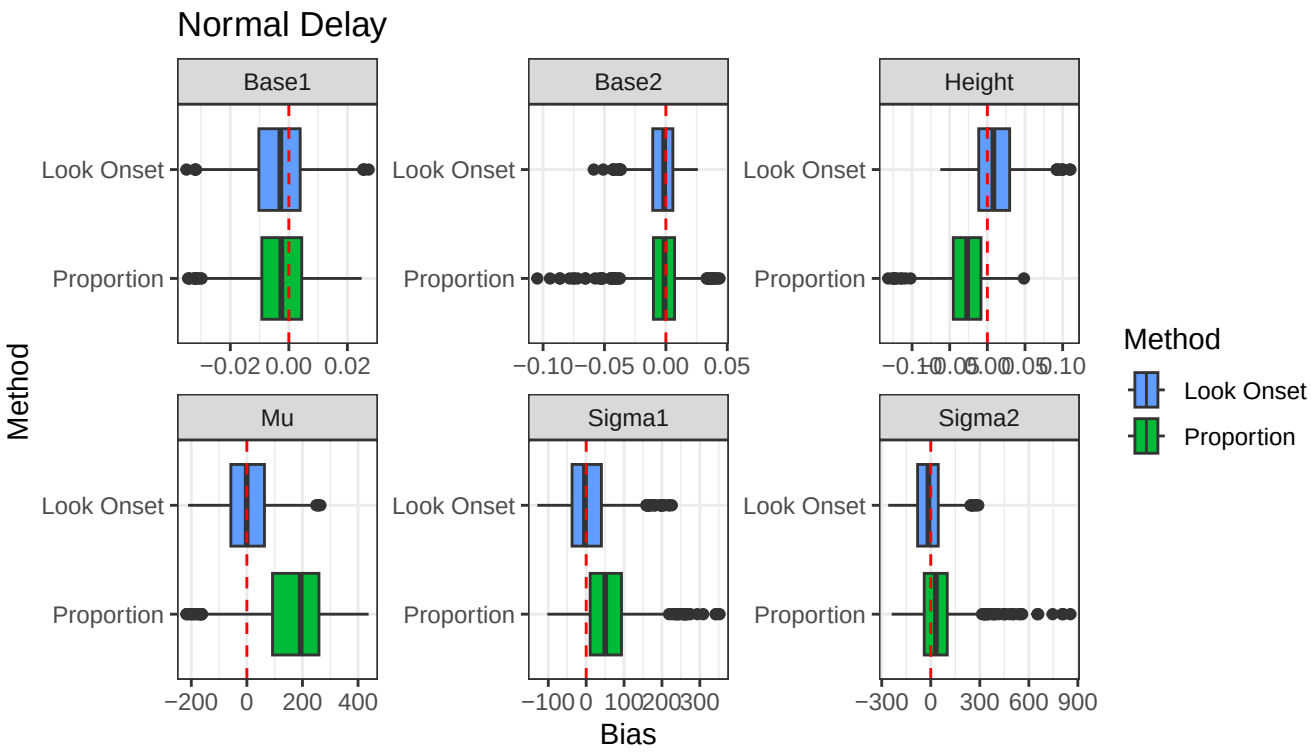


Figure 3.17: Parameter bias for normal OM delay

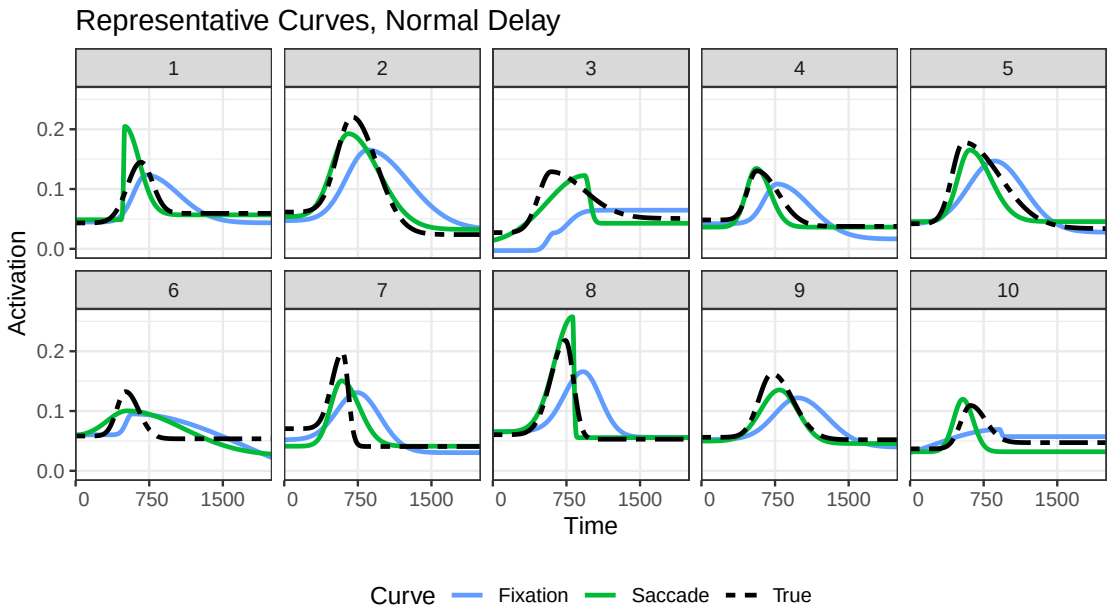


Figure 3.18: Representative curves for normal oculomotor delay

3.6.3 Weibull Delay

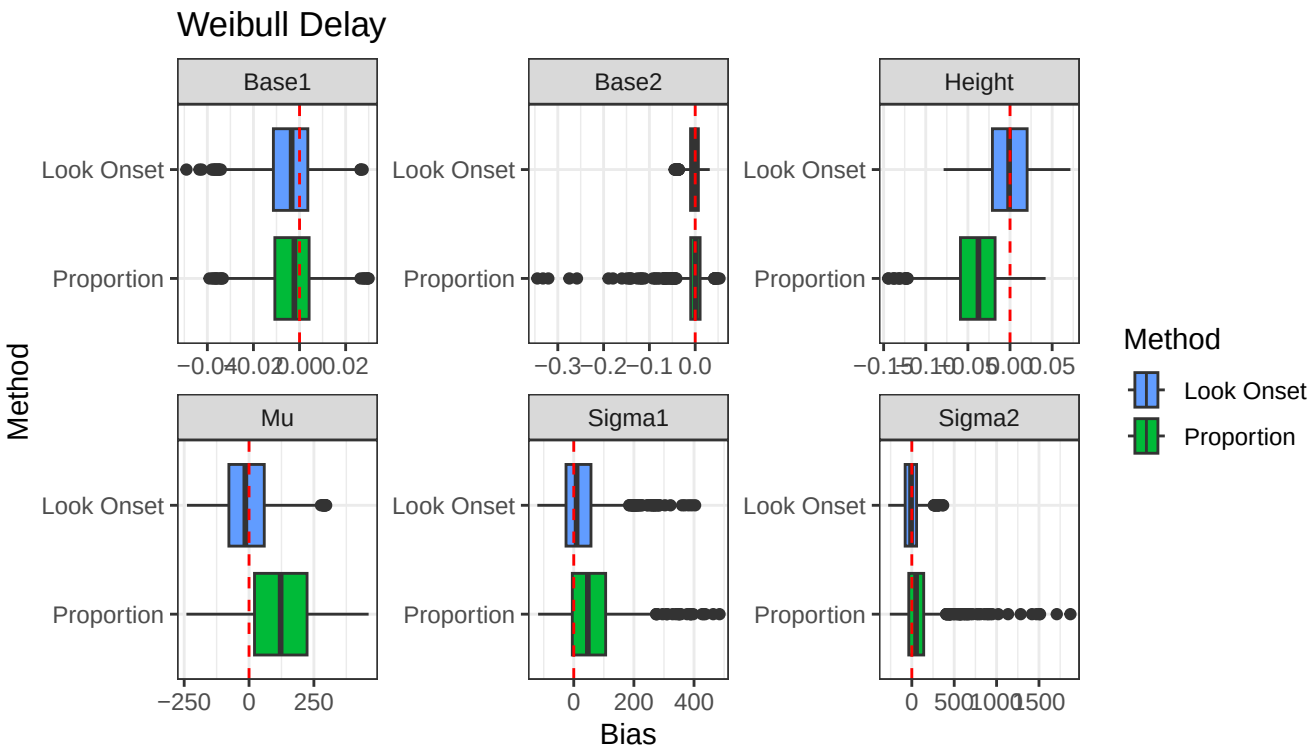


Figure 3.19: Parameter bias for weibull OM delay

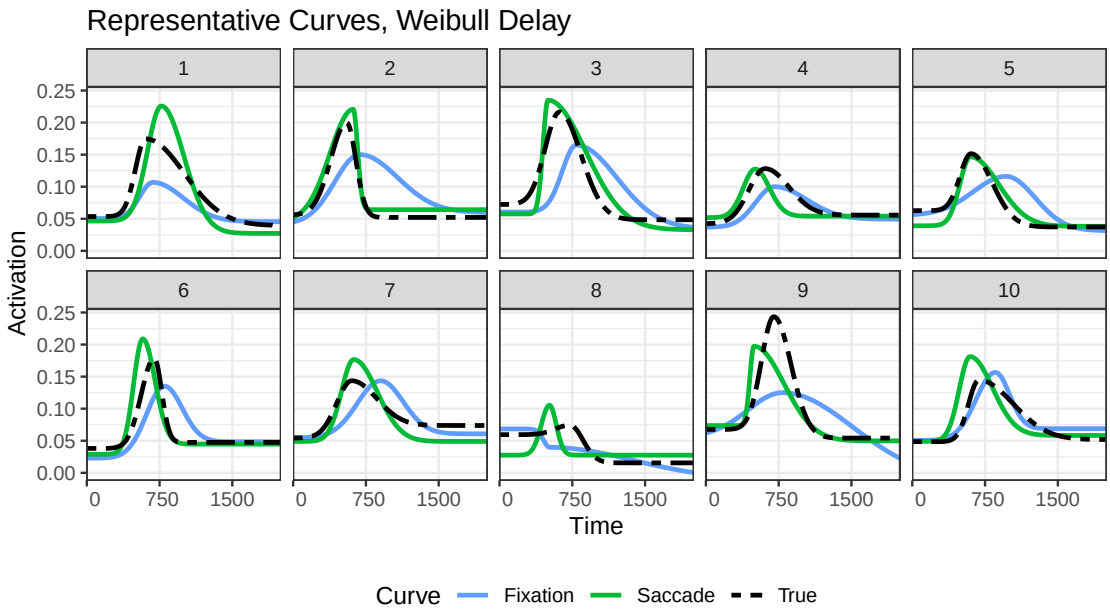


Figure 3.20: Representative curves for weibull oculmotor delay

3.6.4 Results

Curve	Delay	1st Qu.	Median	3rd Qu.
Look Onset	No Delay	0.22	0.36	0.63
Look Onset	Normal Delay	0.38	0.70	1.15
Look Onset	Weibull Delay	0.52	0.84	1.39
Proportion	No Delay	0.75	1.29	2.08
Proportion	Normal Delay	1.38	2.44	3.96
Proportion	Weibull Delay	1.00	1.98	3.43

Table 3.2: Figures not as striking as logistic, but keep in mind that the scale of this is significantly smaller, peaking at around 0.15 and being close to zero elsewhere. R^2 is another metric available

R^2 instead of MISE for logistic/doublegauss

3.6.4.1 for logistic

Curve	Delay	1st Qu.	Median	3rd Qu.
Look Onset	No Delay	1.00	1.00	1.00
Look Onset	Normal Delay	0.99	1.00	1.00
Look Onset	Weibull Delay	0.98	0.99	0.99
Proportion	No Delay	0.92	0.94	0.95
Proportion	Normal Delay	0.80	0.83	0.86
Proportion	Weibull Delay	0.80	0.86	0.91

Table 3.3: R^2 for logistic, not as clear a hierarchy

3.6.4.2 for doublegauss

Curve	Delay	1st Qu.	Median	3rd Qu.
Look Onset	No Delay	0.80	0.91	0.95
Look Onset	Normal Delay	0.63	0.82	0.91
Look Onset	Weibull Delay	0.57	0.77	0.87
Proportion	No Delay	0.48	0.65	0.75
Proportion	Normal Delay	0.10	0.33	0.52
Proportion	Weibull Delay	0.20	0.46	0.64

Table 3.4: R^2 for double gauss

CHAPTER 4
BDOTS METHODOLOGY

4.1 Introduction

A standard problem in psycholinguistics, and the cognitive sciences in general, is that of statistically analyzing a process unfolding in time. Particularly in the case of comparing a process between experimental groups in time, while there are many appropriate techniques for demonstrating that differences *exist*, such as comparing the area under the curve (AUC) of the two processes or testing for differences in the underlying parameters (if the process is being modeled in a parametric manner). However, these offer little insight with respect to identifying *when* these processes are significantly different, which is often the more relevant scientific question. Testing for time differences is complicated by the fact that there are effectively an infinite number of time points that could be compared, with the test results at nearby time points highly correlated. Both of these factors complicate efforts to correct for multiple comparisons.

Various approaches have been proposed, such as using cluster based permutation testing [4]. Here, test-statistics are computed at each time, with adjacent significant tests being combined into clusters. This controls for the family wise error rate (FWER) by reducing adjacent test statistics into a single cluster, thereby reducing the number of tests. More recently, [11] used bootstrapped fits of subject-level curves and estimated the correlation structure of the test statistics in order to modify the Bonferroni correction of the significance level to control FWER. This approach, which they named bootstrapped differences of time series, was introduced in the R package `bdots` [12].

A closer look at the specific conditions under which the original iteration of `bdots` was presented raises concerns, however, involving quite restrictive assumptions that are unlikely to be met in many, if not most, situations. This includes data typically collected in the Visual World Paradigm (VWP), context in which the underlying methodology was first

proposed. Specifically, it assumed a homogeneous mean structure within each of the groups being compared, with no between-subject variability to be accounted for. Empirical data collected in a variety of (VWP) contexts can be used to show that such an assumption is unlikely to be true, with the resulting type I error rate being unacceptably high.

Here, we present two alternatives that accommodate flexibility in the assumptions made by the original bdots. First, we propose a modified bootstrapping procedure that adequately accounts for observed between subject variability while retaining the FWER adjustment method presented for autocorrelated errors. In addition, we offer a permutation test between the groups, borrowing from the insight of the original bdots in that it also captures within-subject variability as demonstrated in the standard errors in the model fits. We begin by describing the two proposed alternatives to the original bdots bootstrap. We then outline the details of the simulations in demonstrating the type I error rate across a number of experimental conditions, and finally we conclude with a demonstration of power in the competing methods.

4.2 Methods

4.2.1 Homogeneous bootstrap

Most generally the original bdots algorithm, which we will call the *homogeneous bootstrap*, proposed that we have empirically observed data in time resulting from a parametric function f with an associated error:

$$y_{it} = f(t|\theta) + \epsilon_{it} \tag{4.1}$$

where

$$\epsilon_{it} = \phi\epsilon_{i,t-1} + w_{it}, \quad w_{it} \sim N(0, \sigma). \tag{4.2}$$

Under this paradigm, the errors could be iid normal (with $\phi = 0$) or have an AR(1) structure, with $0 < \phi < 1$. It further assumes homogeneity of the mean structure, meaning that if subjects i and j are from the same group, we have . In other words, there is no variability in the mean structure between subjects, only between groups.

1. For each subject, fit a nonlinear regression model to obtain $\hat{\theta}_i$. [11] recommended specifying an AR(1) autocorrelation structure for model errors. Assuming large sample normality, the sampling distribution of each estimator can be approximated by a normal distribution with mean corresponding to the point estimate and standard deviation corresponding to the standard error.

2. Using the approximate sampling distributions in (1.), randomly draw one bootstrap estimate for each of the model parameters on every subject .

3. Once a bootstrap estimate has been collected for each parameter and for every subject, for each parameter, find the mean of the bootstrap estimates across individuals .

4. Use the mean estimates to determine the predicted population level curve, which provides the average population response at each time point.

5. Perform steps (2)-(4) 1000 times to obtain estimates of the population curves. Use these to create estimates of the mean response and standard deviation at each of the time points. .

Population means and standard deviations at each time point for each of the groups were used to construct a series of (correlated) test statistics, where the family wise error rate was controlled by using the modified Bonferonni correction introduced in [11] to test for significance.

4.2.2 Heterogeneous Bootstrap

Typically, subjects within a group demonstrate considerable variability in their mean parameter estimates. In this case, we should avoid the presumption that $\theta_i = \theta_j$, since

accounting for between-subject variability within a group will be critical for obtaining a reasonable distribution of the population curves.

Instead of (**??**)

The distribution of parameters for subjects $i = 1, \ldots, n_g$ in group $g = 1, \ldots, G$ follows the distribution

$$\theta_i \sim N(\mu_g, V_g). \tag{4.3}$$

The uncertainty present in our estimation of θ_i can be accounted for by treating the standard errors derived when fitting the observed subject data to Equation 4.2.2 as estimates of their standard deviations. This gives us a distribution for the subject's estimated parameter,

$$\hat{\theta}_i \sim N(\theta_i, s_i^2). \tag{4.4}$$

When obtaining reasonable estimates of the population curve, it is necessary to estimate both the observed within-subject variability found in each of the $\{s_i^2\}$ terms, *as well as* the between-subject variability present in V_g. For example, let θ_{ib}^* represent a bootstrapped sample for subject i in bootstrap $b = 1, \ldots, B$, where

$$\theta_{ib}^* \sim N(\hat{\theta}_i, s_i^2). \tag{4.5}$$

If we were to sample *without replacement*, we would obtain a homogeneous mean value from the bth bootstrap for group g, $\theta_{bg}^{(hom)}$, where

$$\theta_{bg}^{(hom)} = \frac{1}{n_g} \sum \theta_{ib}^*, \quad \theta_{bg}^{(hom)} \sim N\left(\mu_g, \frac{1}{n_g^2} \sum s_i^2\right). \tag{4.6}$$

Such an estimate captures the totality of the within-subject variability with each draw

but fails to account for the variability in the group overall. For this reason, we sample the subjects *with* replacement, creating the heterogeneous bootstrap mean $\theta_{bg}^{(het)}$, where again each θ_{ib}^{*} follows the distribution in Equation 4.5, but the heterogeneous bootstrapped group mean now follows

$$\theta_{bg}^{(het)} \sim N\left(\mu_g, \frac{1}{n_g}V_g + \frac{1}{n_g^2}\sum s_i^2\right). \tag{4.7}$$

The estimated mean value remains unchanged, but the variability is now fully accounted for. We therefore present a modified version of the bootstrap which we call the *heterogeneous bootstrap*, making the following changes to the original:

1. In step (1.), the specification of AR(1) structure is *optional* and can be modified with arguments to functions in `bdots`. Our simulations show that while failing to include it slightly inflates the type I error in the v2 bootstrap when the data truly is autocorrelated, specifying an AR(1) structure can lead to overly conservative estimates when it is not.

2. In step (2.), we sample subjects *with replacement* and then for each drawn subject, randomly draw one bootstrap estimate for each of their model parameters based on the mean and standard errors derived from the `gnls` estimate.

Just as with the homogeneous bootstrap, these bootstrap estimates are used to create test statistics T_t at each time point, written

$$T_t^{(b)} = \frac{(\overline{p}_{1t} - \overline{p}_{2t})}{\sqrt{s_{1t}^2 + s_{2t}^2}}, \tag{4.8}$$

where $\overline{p}_{gt}$ and s_{gt}^2 are mean and standard deviation estimates at each time point for groups 1 and 2, respectively. Finally, just as in Oleson 2017, one can use the autocorrelation of the $T_t^{(b)}$ statistics to create a modified α for controlling the FWER.

4.2.3 Permutation Testing

In addition to the heterogeneous bootstrap, we also introduce a permutation method for hypothesis testing. The permutation method proposed is analogous to a traditional permutation method, but with an added step mirroring that of the previous in capturing the within-subject variability. For a specified FWER of α, the proposed permutation algorithm is as follows:

1. For each subject, fit the nonlinear function with *optional* AR(1) autocorrelation structure for model errors. Assuming large sample normality, the sampling distribution of each estimator can be approximated by a normal distribution with mean corresponding to the point estimate and standard deviation corresponding to the standard error

2. Using the mean parameter estimates derived in (1.), find each subject's corresponding fixation curve. Within each group, use these to derive the mean and standard deviations of the population level curves at each time point, denoted $\overline{p}_{jt}$ and s^2_{jt} for $j = 1, 2$. Use these values to compute a test statistic T_t at each time point,

$$T_t^{(p)} = \frac{|\overline{p}_{1t} - \overline{p}_{2t}|}{\sqrt{s^2_{1t} + s^2_{2t}}}. \tag{4.9}$$

   This will be our observed test statistic.

3. Repeat (2) P additional times, each time shuffling the group membership between subjects. This time, when constructing each subject's corresponding fixation curve, draw a new set of parameter estimates using the distribution found in (1). Recalculate the test statistics $T_t^{(p)}$, retaining the maximum value from each permutation. This collection of P statistics will serve as our null distribution which we denote $\widetilde{T}$. Let $\widetilde{T}_\alpha$ be the 1 - α quantile of $\widetilde{T}$

4. Compare each of the observed $T_t^{(p)}$ with $\widetilde{T}_\alpha$. Areas where $T_t^{(p)} > \widetilde{T}_\alpha$ are designated significant.

Paired permutation testing is implemented with a minor adjustment to step (3). Instead of permuting all of the labels between groups, randomly assign each subject to either retain their current group membership or to change groups. This ensures that each permuted group contains one observation from each subject.

4.3 Type I Error Simulations

We now go about comparing the type I error rate of the three methods just described. In doing so, we will consider several conditions under which the observed subject data may have been generated or fit. This includes two conditions for the mean structure, two conditions for the error structure, paired and unpaired data, and data fit with and without an AR(1) assumption. Considering each permutation of this arrangement results in sixteen different settings. Each simulation will then be examined for type I error using each of the three methods described.

4.3.1 Data Generation

Data was generated according to Equation 4.2.2, with the parametric function $f(t|\theta)$ belonging to the family of four-parameter logistic curves defined:

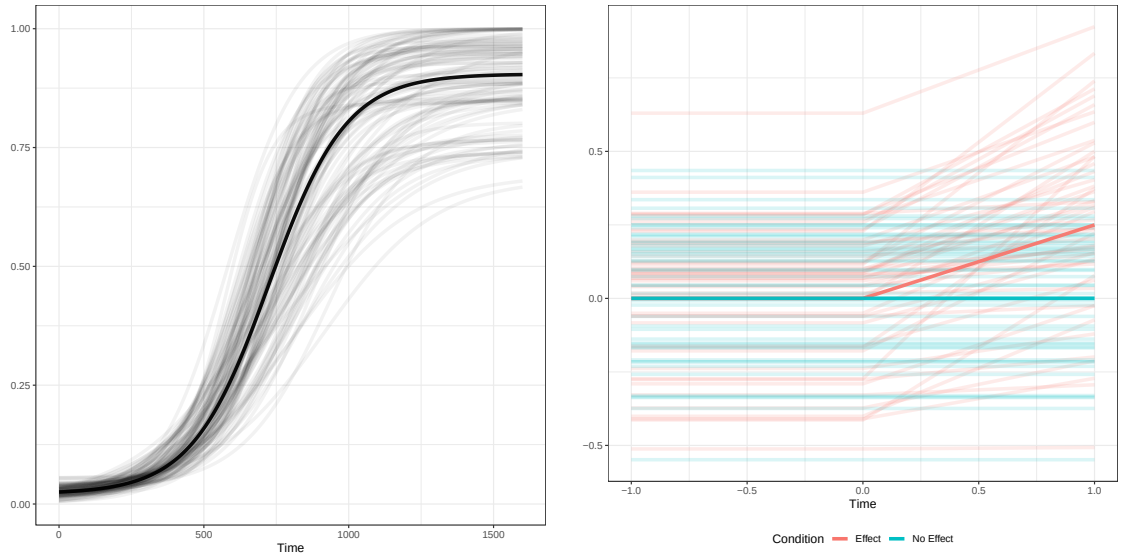
$$f(t|\theta) = \frac{p-b}{1+\exp\left(\frac{4s}{\mathrm{p}-b}(x-t)\right)} + b \tag{4.10}$$

where $\theta = (p, b, s, x)$, the peak, baseline, slope, and crossover parameters, respectively.

We further assume that each group drew subject-specific parameters from a normal distribution, with subject $i = 1, \ldots, N$ in group $g = 1, \ldots, G$ following the distribution in Equation 4.3.

Mean Structure

In all of the simulations presented, the distribution used in Equation **??** was empirically determined from data on normal hearing subjects in the VWP (Farris-Trimble et al., 2014



(a) (b)

Figure 4.1: Distributions of various group with the mean curve in bold (a.) 50 samples from the generating distribution of the four-parameter logistic in Equation 4.10 used for testing FWER. (b.) 50 samples from the generating distributions of each group in Equation 4.12. The legend makes size weird, might just explain what they are here. Also need to see if I can change size of the mean lines

[2]). Parameters used were those fit to fixations on the Target, following the functional form of Equation 4.10. Under the assumption of between-subject homogeneity, we set $\theta_i = \theta_j$ for all subjects i, j, assuring that each of the subjects' observations is derived from the same mean structure, differing only in their observed error. We will call this the homogeneous means assumption.

Error Structure

The error structure is of the form

$$e_{it} = \phi e_{i,t-1} + w_{it}, \quad w_{it} \sim N(0, \sigma) \tag{4.11}$$

where the w_{it} are iid with $\sigma = 0.025$. ϕ corresponds to an autocorrelation parameter and is set to $\phi = 0.8$ when the generated data is to be autocorrelated and set to $\phi = 0$ when we

assume the errors are all independent and identically distributed.

Paired Data

Finally we considered paired data, though this is only a sensible condition under the assumption of heterogeneous means. In order to construct paired data, we simply used the same parameter estimates for each subject between groups, with the observed data between subjects differing only in the observed error.

Each set of conditions generates two groups, with $n = 25$ subjects in each group, with timepoints $t = 0, 4, 8, \ldots, 1600$ in each trial and with 100 simulated trials for each subject. Columns in the tables indicate homogeneity of means assumption, whether or not an AR(1) error structure was used in constructing the data, and if autocorrelation was specified in the fitting function. The last conditions helps assess the impact of correctly identifying the type of error when conducting an analysis in `bdots`. Each simulation was conducted 100 times to determine the rate of type I error (time intensive and will launch 1000 for final dissertation).

4.3.2 Results

We consider the efficacy the methods under each of the simulation settings with an analysis of the family wise error rate (FWER) and the median per-comparison error rate. The first of these details the proportion of simulations under each condition that marked *at least* one time point as being significantly different between the two groups. This is critical is understanding each method's ability to correct adjust for the multiple testing problem associated with testing each of the observed time points. These are presented in Table 4.1 and Table 4.2 for unpaired and paired data, respectively.

Complimenting the FWER estimate is an estimate of the median per-comparison rate. For each time point across each of the simulations, we computed the proportion of times in which that time was determined significant. The median of these values across all time points is what is considered. This metric gives a sense of magnitude to the binary FWER; for example, a situation in which there was a high FWER and low median per-comparison

rate would indicate that the type I error within a particular time series would be sporadic and impact limited regions. Large median per-comparison rates indicate that large swaths of a time series frequently sustain type I errors. The median per-comparison rates for unpaired and paired simulations are presented in Table 4.3 and Table 4.4.

4.3.2.1 FWER

There are a few things of immediate note when considering the results of Table 4.1. First, we see from the first two settings of the unpaired simulations that the type I error rates for the homogenous bootstrap are consistent with those presented in [11], confirming the importance of specifying the existence of autocorrelation in the `bdots` fitting function when autocorrelated error is present. By contrast, this is far less of a concern when using the heterogeneous bootstrap or permutation testing, both of which maintain a FWER near the nominal alpha, regardless of whether or not the error structure was correctly identified. This continues to be true under the homogenous mean assumption when the true error structure is not autocorrelated. Interestingly, the performance of the homogeneous bootstrap falters here despite theoretical consistency with the simulation settings.

The most striking results of this, however, appear when the data generation assumes a heterogeneous mean structure. While both the heterogeneous bootstrap and the permutation test maintain a FWER near the nominal alpha, the homogeneous bootstrap fails entirely, with a FWER > 0.9 in all cases.

Heterogeneity Assumption	Autocorrelated Error	AR(1) Specified	Homogeneous Bootstrap	Heterogeneous Bootstrap	Permutation
No	Yes	Yes	0.06	0.01	0.08
No	Yes	No	0.87	0.08	0.00
No	No	Yes	0.08	0.00	0.06
No	No	No	0.15	0.02	0.01
Yes	Yes	Yes	0.92	0.03	0.05
Yes	Yes	No	0.96	0.02	0.08
Yes	No	Yes	0.99	0.05	0.03
Yes	No	No	1.00	0.05	0.06

Table 4.1: FWER for empirical parameters (unpaired)

Paired data is given in Table 4.2. Matching the conclusions drawn from Table 4.1, we only note here that both the permutation test and heterogeneous bootstraps maintain a valid FWER under the assumption of paired data.

Heterogeneity Assumption	Autocorrelated Error	AR(1) Specified	Homogeneous Bootstrap	Heterogeneous Bootstrap	Permutation
Yes	Yes	Yes	0.49	0.02	0.01
Yes	Yes	No	0.94	0.03	0.02
Yes	No	Yes	0.72	0.02	0.00
Yes	No	No	0.74	0.04	0.00

Table 4.2: FWER for empirical parameters (paired)

4.3.2.2 Median per comparison error rate

We next consider the median comparison rate, which offers some insight into the FWER. In particular, consider the situation in which in Table 4.3, in the fourth row we see a median per-comparison error rate of 0.00 for the homogeneous bootstrap, despite Table 4.1 indicating a FWER of 0.15. This is a consequence of the majority of the type I errors occuring in a relatively limited region. In contrast, the median per-comparison error rate of the homogeneous bootstrap under the assumption of heterogeneity suggests that the type I errors are widespread and not limited to any particular area.

It is also worth commenting on the permutation test median per-comparison error rate in Table 4.3; combined with a FWER near the nominal 0.05, these values suggest that errors are likely distributed across the entire range rather than limited to a small area (which would result in a MPCR of 0).

Heterogeneity Assumption	Autocorrelated Error	AR(1) Specified	Homogeneous Bootstrap	Heterogeneous Bootstrap	Permutation
No	Yes	Yes	0.01	0.00	0.04
No	Yes	No	0.31	0.00	0.04
No	No	Yes	0.00	0.00	0.02
No	No	No	0.00	0.00	0.03
Yes	Yes	Yes	0.51	0.01	0.01
Yes	Yes	No	0.76	0.01	0.00
Yes	No	Yes	0.86	0.01	0.00
Yes	No	No	0.81	0.01	0.00

Table 4.3: median per comparison error rate (unpaired)

Heterogeneity Assumption	Autocorrelated Error	AR(1) Specified	Homogeneous Bootstrap	Heterogeneous Bootstrap	Permutation
Yes	Yes	Yes	0.13	0.00	0.00
Yes	Yes	No	0.52	0.02	0.00
Yes	No	Yes	0.38	0.01	0.00
Yes	No	No	0.44	0.01	0.00

Table 4.4: median per comparison error rate (paired)

4.3.3 Discussion

Table 4.1 and Table 4.2 demonstrate that under a variety of settings, both the heterogeneous bootstrap and permutation offer a FWER near the nominal alpha, making their performance similar to that of the homogeneous bootstrap in the best of cases. This includes less restrictive assumptions in which they continue to perform well while the homogeneous bootstrap maintains an unacceptably high FWER.

Transition sentence to power simulations

4.4 Power Simulations

To determine power, two experimental groups were simulated with mean structures of the following form:

$$y = \begin{cases} b & x < 0 \\ mx + b & x \geq 0 \end{cases} \tag{4.12}$$

The first simulated group was "No Effect", with intercept and slope parameters normally distributed and standard deviation $\sigma = 0.05$. The second group, the "Effect" group, was similarly distributed, but with the slope parameter having mean value of $\mu = 0.25$. The error structure was identical to that in the FWER simulations, with both an AR(1) error

structure and independent noise included. 100 simulations were conducted for each scenario.

We limited consideration to three possible scenarios: first, we assumed the conditions presented in Oleson 2017, assuming homogeneity between subject parameters and an AR(1) error structure, with the model fitting performed assuming autocorrelated errors. For the remaining scenarios, we assumed heterogeneity in the distribution of subject parameters, simulated with and without an AR(1) error structure. In both of these last two scenarios, we elected to *not* fit the model assuming autocorrelated errors. This was for two reasons: first, simulations exploring the type I error rate suggested that models fit with the autocorrelation assumption tended to be conservative. Second, and given the results of the first, this makes setting the assumption of autocorrelation to FALSE in `bdots` seem like a sensible default, and as such, it would be of interest to see how the model performs in cases in which their is autocorrelated error that is not accounted for.

For each subject, parameters for their mean structure given in Equation 4.12 were drawn according to their group membership and fit using `bdots` on the interval (-1,1). Time windows in which the groups differed were identified using each the homogenous bootstrap, heterogeneous bootstrap, and permutation testing. By including the interval (-1,0) in which the null hypothesis was true, we are able to mitigate the effects of over-zealous methods in determining power, and we present the results in the following way: any tests in which a difference was detected in (-1,0) was marked as having a type I error, and the proportion of simulations in which this occurred for each method is reported as the FWER in the column labeled α. The next column, β, is the type II error rate, indicating the proportion of trials in which no differences were identified over the entire region. The last Greek-letter column is $1 - \beta - \alpha$, a modified power statistic indicating the proportion of tests in which a difference was correctly identified. The remaining columns relate to this modified power columns, giving a partial summary of the earliest onset of detection. As a true difference occurs on the interval $t > 0$, smaller values indicate greater power in detecting differences. Finally, a (currently base-R) plot giving the power at each time point is given in Figure 4.2. This plot

represents the true power, though note that it does not take into account the rate at which these regions were identified in conjunction with a type I error rate.

4.4.1 Results

The results of the power simulation are presented in Table 4.5. We begin by considering the case in which we assumed a homogeneous mean structure with autocorrelated errors (the first three rows), matching the conditions in which the homogeneous bootstrap was first presented. Notably, we find that the permutation method demonstrates the greater power, with the median onset time just under that of the homogeneous bootstrap. This is at the expense of a larger FWER, though still below the nominal level. Alternatively, the heterogeneous bootstrap maintains a similar FWER as the homogeneous bootstrap at the cost of power.

The remaining settings tell a similar story with the exception of the homogeneous bootstrap which continues to demonstrate unacceptable FWER under the heterogeneous means assumption. We also seem some effect of not correctly specifying an AR(1) error structure when comparing the last two settings, as the failure to specify resulted in a slightly higher FWER and lower power, although not significantly (it may be worth it to include setting with AR(1) error correctly specified to see how it compares).

Method	Heterogeneity	AR(1)	α	β	1 - α - β	1st Qu.	Median	3rd Qu.
Hom. Boot	No	Yes	0.00	0.00	1.00	0.025	0.030	0.035
Het. Boot	No	Yes	0.00	0.00	1.00	0.035	0.040	0.045
Perm	No	Yes	0.03	0.00	0.97	0.015	0.025	0.025
Hom. Boot	Yes	No	0.96	0.00	0.04	0.005	0.008	0.010
Het. Boot	Yes	No	0.00	0.10	0.90	0.403	0.513	0.690
Perm	Yes	No	0.03	0.05	0.92	0.378	0.515	0.681
Hom. Boot	Yes	Yes	0.97	0.00	0.03	0.008	0.010	0.010
Het. Boot	Yes	Yes	0.01	0.10	0.89	0.420	0.525	0.690
Perm	Yes	Yes	0.08	0.03	0.89	0.360	0.540	0.705

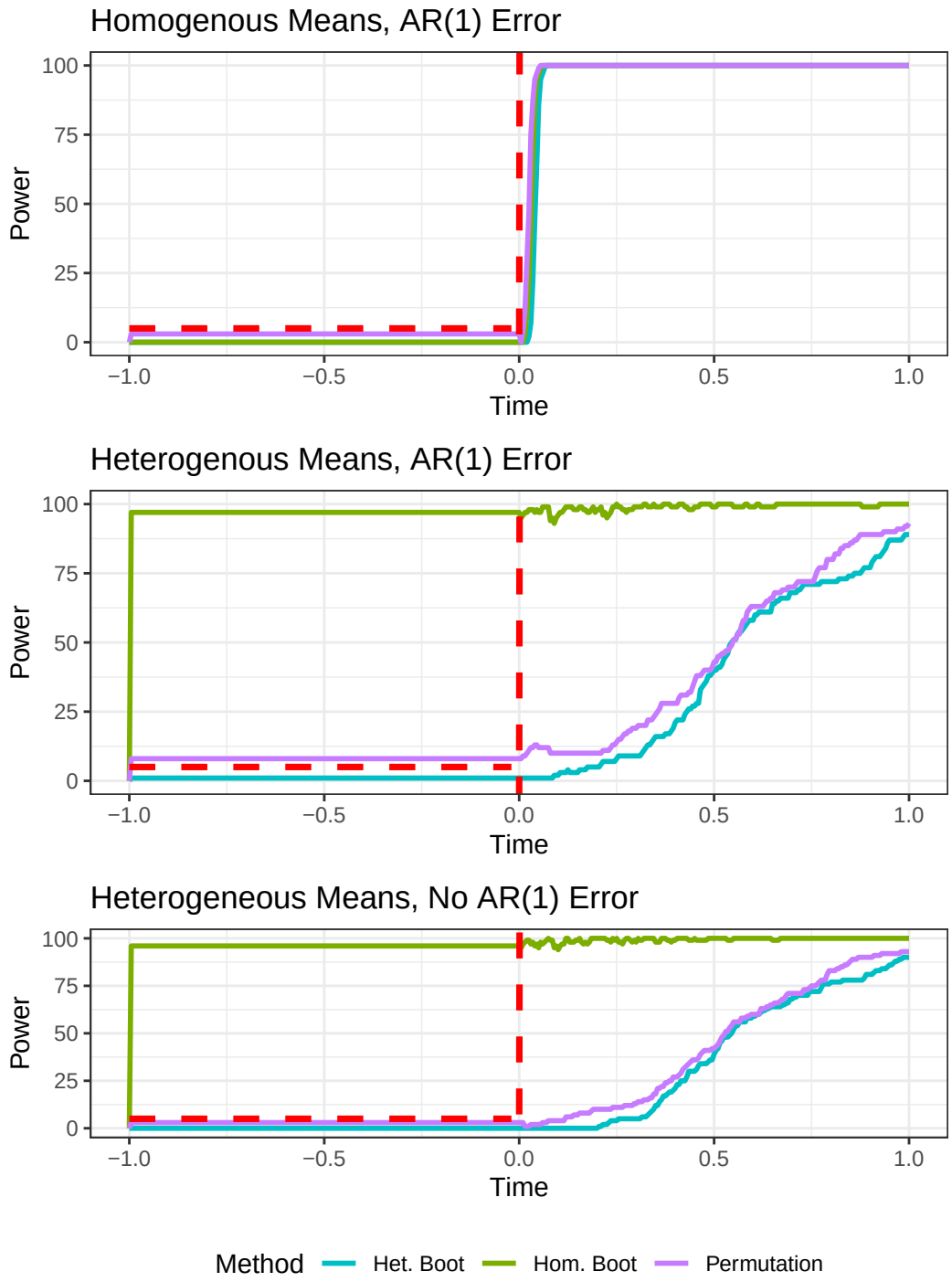
Table 4.5: Power for methods (possibly include AR(1) error and AR(1) specification to compare to last setting)

The results in Table 4.6 are a summary of all of the methods found by taking the mean of each of the results presented. This is intended to interrogate the performance of each of these methods when underlying assumptions are unknown or unspecified. We find a robust performance for each of the new methods presented, maintaining a reasonable relationship between FWER and power. The metrics associated with homogeneous bootstrap are perhaps a bit misleading here as they appear to demonstrate exceptional power, though at the cost of unacceptable type I error.

Method	α	β	$1-\alpha-\beta$	1st Qu.	Median	3rd Qu.
Hom. Bootstrap	0.643	0.000	0.357	0.013	0.016	0.018
Het. Bootstrap	0.003	0.067	0.930	0.286	0.359	0.475
Permtuation	0.047	0.027	0.927	0.251	0.360	0.470

Table 4.6: Summary of methods for Type II error

4.4.1.1 Summary of methods



.

Figure 4.2: Observed power of each of the methods at each time in (-1,1) (not sure if better as subplots?)

4.5 Discussion and concluding remarks

We set out both to interrogate the validity of the homogenous bootstrap assumptions and to propose two alternative methods that would be more robust under a greater variety of assumptions. In doing so, we demonstrated conclusively the utility of the heterogeneous bootstrap and permutation tests while also highlighting a major shortcoming of the original. It's worth noting, however, that the FWER adjustment proposed in [11] is still valid, if not slightly conservative, and with power similar to that of the permutation method.

In light of the results presented, one issue of concern is addressing the fact that a version of `bdots` with the homogeneous mean assumption was presented in 2018 and remained accessible on CRAN until the end of 2022. This has implications for the number of papers in which `bdots` may have demonstrated significance between groups when the underlying assumptions of homogeneous mean structure did not hold, as is likely the case in all instances related to the VWP. Concurrent with this issue is the issue of identifying current users of `bdots` of this change, as results found only a month ago will be profoundly different than what is seen today. At present, I am not sure the best way to address either of these. In either case, however, it will be prudent to remove this option from the `bdots` package all together, as there appears to be no obvious advantage to the homogeneous bootstrap over the others in terms of either controlling the FWER or obtaining power, even when the homogeneous mean structure assumption is met.

There are several limitations of the current paper that are worthy of further investigation. First, First, limited consideration was given to the effect of sample density on the observed type I error rate or power. As the fitting function in `bdots` simply returns a set of parameters, one could conceivably perform any of the methods presented on any arbitrary collection of points, whether or not any data were observed there. This extends itself to the condition in which subjects were sampled at heterogeneous time points, as may be the case in many clinical settings. What impact this may have or how to best handle these cases

remains open for exploration. It is also worth investigating in greater detail what impact the re-drawing of subject specific parameters from their respective distributions has on both the FWER and power, as in several of the simulations the observed FWER was much lower than the nominal level. Particularly in the case of the permutation method which is *not* seeking to estimate the group distributions, it may be worthwhile to see if a favorable trade can be made to increase the resulting power.

We conclude by noting that `bdots` is now equipped with two methods to effectively control the FWER when assessing the differences in time series under a greater set of underlying assumptions, including those involving the presence of highly correlated test statistics. Further, both methods presented are robust to misspecification of the error structure while maintaining an acceptable FWER and adequate power.

Bibliography

[1] Paul D Allopenna, James S Magnuson, and Michael K Tanenhaus. Tracking the time course of spoken word recognition using eye movements: Evidence for continuous mapping models. *Journal of memory and language*, 38(4):419–439, 1998.

[2] Ashley Farris-Trimble, Bob McMurray, Nicole Cigrand, and J. Bruce Tomblin. The process of spoken word recognition in the face of signal degradation. *Journal of Experimental Psychology: Human Perception and Performance*, 40(1):308–327, feb 2014.

[3] James S. Magnuson. Fixations in the visual world paradigm: where, when, why? *Journal of Cultural Cognitive Science*, 3(2):113–139, sep 2019.

[4] Eric Maris and Robert Oostenveld. Nonparametric statistical testing of EEG- and MEG-data. *Journal of Neuroscience Methods*, 164(1):177–190, aug 2007.

[5] James L. McClelland and David E. Rumelhart. An interactive activation model of context effects in letter perception: I. an account of basic findings. *Psychological Review*, 88(5):375–407, sep 1981.

[6] Bob McMurray. I'm not sure that curve means what you think it means: Toward a [more] realistic understanding of the role of eye-movement generation in the visual world paradigm. *Psychonomic Bulletin & Review*, pages 1–45, 2022.

[7] Bob McMurray, Jamie Klein-Packard, and J. Bruce Tomblin. A real-time mechanism underlying lexical deficits in developmental language disorder: Between-word inhibition. *Cognition*, 191:104000, oct 2019.

[8] Bob McMurray, Vicki M Samelson, Sung Hee Lee, and J Bruce Tomblin. Individual differences in online spoken word recognition: Implications for sli. *Cognitive psychology*, 60(1):1–39, 2010.

[9] Bob McMurray, Michael K. Tanenhaus, and Richard N. Aslin. Gradient effects of within-category phonetic variation on lexical access. *Cognition*, 86(2):B33–B42, dec 2002.

[10] Daniel Mirman, James A. Dixon, and James S. Magnuson. Statistical and computational models of the visual world paradigm: Growth curves and individual differences. *Journal of Memory and Language*, 59(4):475–494, nov 2008.

[11] Jacob J Oleson, Joseph E Cavanaugh, Bob McMurray, and Grant Brown. Detecting time-specific differences between temporal nonlinear curves: Analyzing data from the visual world paradigm. *Statistical methods in medical research*, 26(6):2708–2725, 2017.

[12] Michael Seedorff, Jacob Oleson, and Bob McMurray. Detecting when timeseries differ: Using the bootstrapped differences of timeseries (bdots) to analyze visual world paradigm data (and more). *Journal of memory and language*, 102:55–67, 2018.

[13] Michael J. Spivey, Marc Grosjean, and Günther Knoblich. Continuous attraction toward phonological competitors. *Proceedings of the National Academy of Sciences*, 102(29):10393–10398, jun 2005.

[14] Michael K Tanenhaus, Michael J Spivey-Knowlton, Kathleen M Eberhard, and Julie C Sedivy. Integration of visual and linguistic information in spoken language comprehension. *Science*, 268(5217):1632–1634, 1995.

[15] P. Viviani. Eye movements in visual search: cognitive, perceptual and motor control aspects. *Eye movements and their role in visual and cognitive processes*, pages 353–393, 1990.