Eyetracking Analysis in R

Michael Seedorff

Department of Biostatistics
University of Iowa

Jacob Oleson

Department of Biostatistics
University of Iowa

Grant Brown

Department of Biostatistics
University of Iowa

Joseph Cavanaugh

Department of Biostatistics
University of Iowa

Bob McMurray

Departments of Psychology, Communication Sciences and Disorders, and Linguistics
University of Iowa

Abstract

The package **bdots** provides techniques for analyzing eyetracking data, as well as other types of highly correlated data that consist of many consecutive tests. We explain how to set the data up in a fashion for the package to use, how to analyze the data using the built in functions, and how to check the fit to the data.

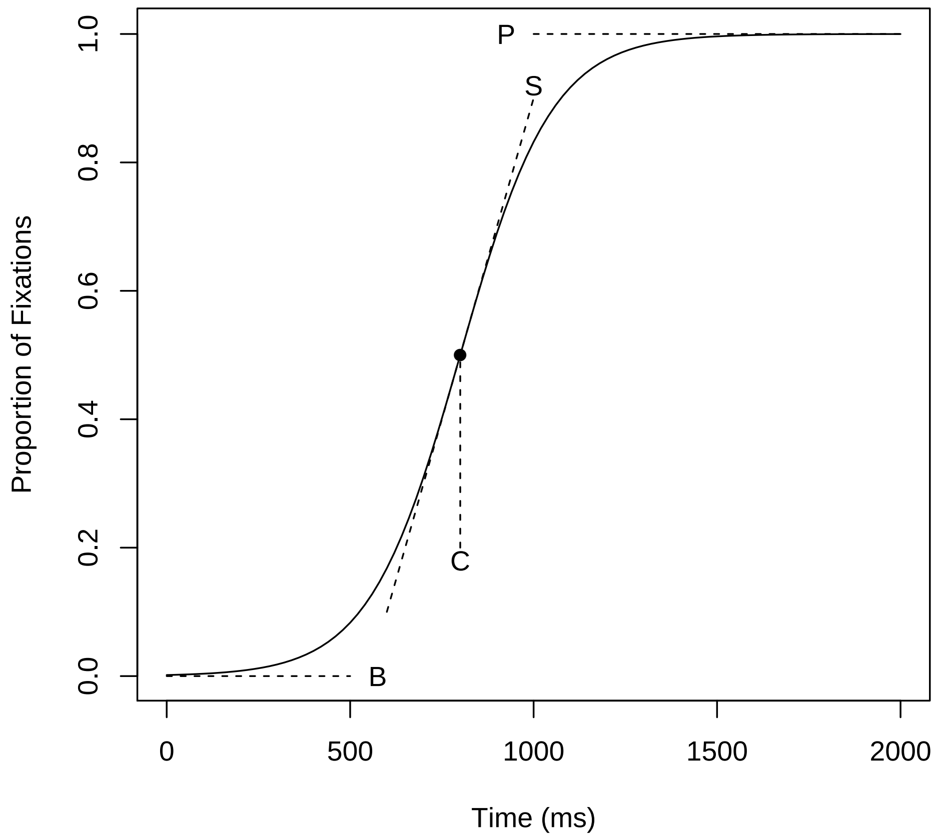
1 Introduction

If using **bdots** for analysis in a paper, please cite current version using:

```
R> citation("bdots")
```

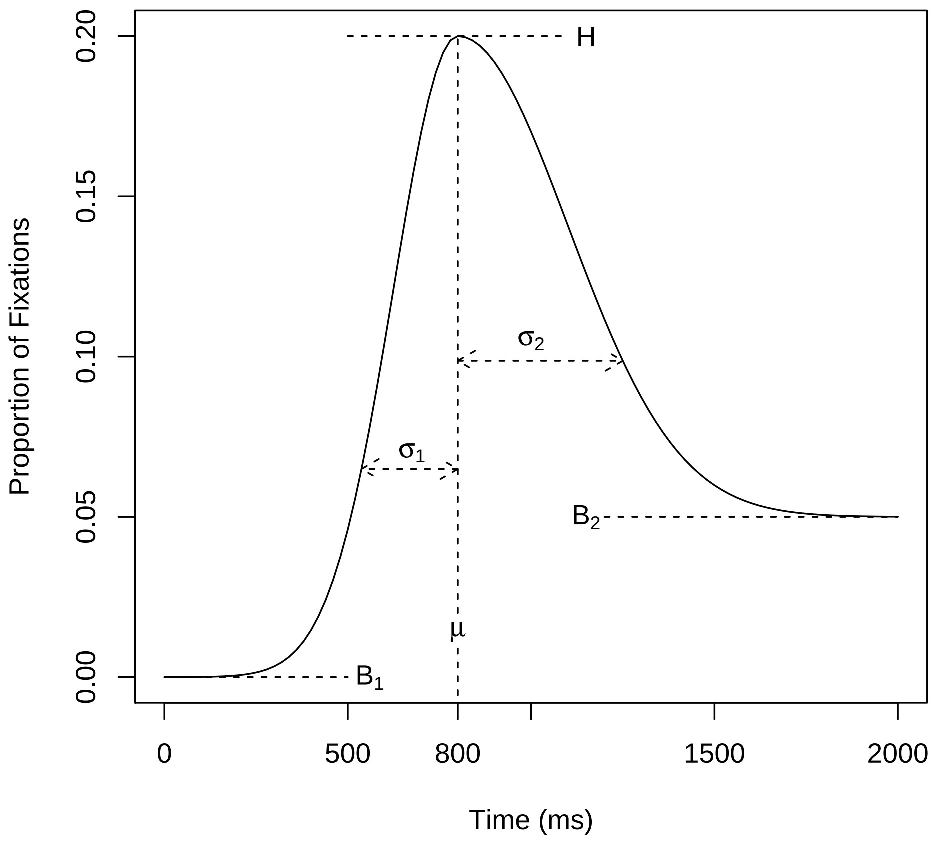
For a reference to the statistical methods used in this package, see [2]. In this vignette we provide a sample analysis of eyetracking data obtained from the study [1]. We use the following 2 functions to fit the data: a 4-parameter logistic and an asymmetric Gaussian. The 4-parameter logistic is defined as the following:

$$p_{it} = B_i + \frac{P_i - B_i}{1 + \exp\left(4 * S_i \frac{(C_i - t)}{(P_i - B_i)}\right)}$$



The asymmetric Gaussian is defined as the following:

$$p_{it} = \begin{cases} (H_i - B_{1i}) * \exp\left(\frac{(t-\mu_i)^2}{-2\sigma_{1i}^2}\right) + B_{1i} & \text{if } t \leq \mu_i \\ (H_i - B_{2i}) * \exp\left(\frac{(t-\mu_i)^2}{-2\sigma_{2i}^2}\right) + B_{2i} & \text{if } t > \mu_i \end{cases}$$



While only these 2 functions have been implemented, it would be straightforward to implement more functions in the future (e.g. polynomials).

2 Curve Fitting

2.1 Formatting Data

The data needs the following columns:

- Time: A column labelled 'Time' that tracks the time variable (the 'x' values)
- Group: A column labelled 'Group' that designates the 2 groups that will be compared (should only have 2 unique values in the column)
- Subject: A column named 'Subject' that uses numbers to designate each of the individual curves within a group
- Data: A column containing the observed value at each time point (the 'y' values)

```
R> data(ci)
R> names(ci)[1] <- "Group"
R> head(ci)
```

```
  Group Subject Time Fixations LookType
1    NH      36    0         0   Cohort
2    NH      36    4         0   Cohort
3    NH      36    8         0   Cohort
4    NH      36   12         0   Cohort
5    NH      36   16         0   Cohort
6    NH      36   20         0   Cohort
```

2.2 Fitting Individual Curves

The first step is fitting each individual's curve to the specified function. There are 2 implemented functions: the 4-parameter logistic and the asymmetric Gaussian (referred to as a Double Gaussian in the code). For the first example, we use the logistic to compare the looks to a 'target' object in an eyetracking experiment between a group of normal hearing individuals and a group of individuals with cochlear implants (a device used to restore hearing to deaf or near-deaf individuals).

```
R> ci.1 <- subset(ci, ci$LookType == "Target")
```

We want to fit the logistic to each of the individual curves. For this purpose, the fitting method requires an initial estimate as to the autocorrelation between the errors (generally alright to leave this at the default of 0.9). Using this correlation between the errors allows for better estimates of the standard deviations around the mean estimates, which will be used later in the bootstrapping section (correlation along the errors can be turned off for curve-fitting by setting `cor = FALSE`). If a curve cannot be fit using AR1 correlation assumption, the curve is fit without the autocorrelation. If a curve can be fit with the AR1 correlation assumption but is a poor fit to the observed curve, the user should manually refit the curve using approaches discussed in section 2.4. The status of each individual's AR1 vs non-AR1 fit can be found by looking in the `coef.id#` output vector. `coef.id1` corresponds to the first curves for the first group. `coef.id2` corresponds to the first curves for the second group. `coef.id3` and `coef.id4` correspond to the second curves for the first and second groups, if `diffs = TRUE`.

The designation of more than 1 core for processing can significantly reduce time required for computation. A general recommendation is to set the number of cores used equal to the total number of cores available (only true physical cores) minus 1. For instance, we are running this on an Intel Core i7 2600 with 4 physical cores in addition to 4 'hyper-threaded cores'. We would use 3 cores for the computations (4 physical cores - 1). For compilation of package vignettes CRAN has a limit of 2 cores being used in parallel so we will limit ourselves to 2 for this vignette.

Finally, the column corresponding to the observed values needs to be designated as a number.

```
R> ci.1.out.1 <- logistic.fit(ci.1, col = 4, rho.0 = 0.9, cor = TRUE,
+           cores = 2)

########################################
############### FITS ###################
########################################
AR1,       R2>=0.95   -- 53
AR1,     0.95>R2>=0.8 -- 0
AR1,       0.8>R2     -- 0
Non-AR1,   R2>=0.95   -- 1
Non-AR1, 0.95>R2>=0.8 -- 0
Non-AR1,   0.8>R2     -- 0
No Fit                -- 0
########################################

Next Steps: Check goodness of fits (ests.plot, subs.plot),
        refit bad fits (logistic.refit),
        bootstrap and t-test (logistic.boot)

R> ci.1.out.1$cor.1

 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1

R> ci.1.out.1$cor.2

 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

For our second example we are going to look at the difference in a person's looks to 'cohort' and 'unrelated' objects. We then want to compare these between the normal hearing and cochlear implant groups. We maintain the same structure of the data, but use an additional column called 'Curve' using the numbers 1 and 2 to designate whether that curve corresponds to a 'cohort' or 'unrelated' eyetrack. Additionally, the parameter `diffs` should be set to TRUE, in this situation.

For this example we will set the cores equal to 1 to show some extra output you receive by setting this option.

```
R> ci.2 <- subset(ci, ci$LookType == "Cohort" | ci$LookType == "Unrelated")
R> ci.2$Curve <- ifelse(ci.2$LookType == "Cohort", 1, 2)
R> ci.2.out.1 <- doubleGauss.fit(ci.2, col = 4, rho.0 = 0.9, cor = TRUE,
+          cores = 1, diffs = TRUE)
```

```
[1] "Group = NH, Subject = 36, Curve = 1, R2 = 0.99, AR1 = TRUE"
[1] "Group = NH, Subject = 36, Curve = 2, R2 = 0.996, AR1 = TRUE"
[1] "Group = NH, Subject = 38, Curve = 1, R2 = 0.995, AR1 = TRUE"
[1] "Group = NH, Subject = 38, Curve = 2, R2 = 0.988, AR1 = TRUE"
[1] "Group = NH, Subject = 40, Curve = 1, R2 = 0.995, AR1 = TRUE"
[1] "Group = NH, Subject = 40, Curve = 2, R2 = 0.99, AR1 = TRUE"
[1] "Group = NH, Subject = 42, Curve = 1, R2 = 0.994, AR1 = TRUE"
[1] "Group = NH, Subject = 42, Curve = 2, R2 = 0.993, AR1 = TRUE"
[1] "Group = NH, Subject = 47, Curve = 1, R2 = 0.987, AR1 = TRUE"
[1] "Group = NH, Subject = 47, Curve = 2, R2 = 0.993, AR1 = TRUE"
[1] "Group = NH, Subject = 51, Curve = 1, R2 = 0.992, AR1 = TRUE"
[1] "Group = NH, Subject = 51, Curve = 2, R2 = 0.995, AR1 = TRUE"
[1] "Group = NH, Subject = 53, Curve = 1, R2 = 0.989, AR1 = TRUE"
[1] "Group = NH, Subject = 53, Curve = 2, R2 = 0.964, AR1 = TRUE"
[1] "Group = NH, Subject = 55, Curve = 1, R2 = 0.991, AR1 = TRUE"
[1] "Group = NH, Subject = 55, Curve = 2, R2 = 0.99, AR1 = TRUE"
[1] "Group = NH, Subject = 56, Curve = 1, R2 = 0.997, AR1 = TRUE"
[1] "Group = NH, Subject = 56, Curve = 2, R2 = 0.987, AR1 = TRUE"
[1] "Group = NH, Subject = 58, Curve = 1, R2 = 0.994, AR1 = TRUE"
[1] "Group = NH, Subject = 58, Curve = 2, R2 = 0.989, AR1 = TRUE"
[1] "Group = NH, Subject = 60, Curve = 1, R2 = 0.981, AR1 = TRUE"
[1] "Group = NH, Subject = 60, Curve = 2, R2 = 0.984, AR1 = TRUE"
[1] "Group = NH, Subject = 61, Curve = 1, R2 = 0.945, AR1 = TRUE"
[1] "Group = NH, Subject = 61, Curve = 2, R2 = 0.992, AR1 = TRUE"
[1] "Group = NH, Subject = 63, Curve = 1, R2 = 0.995, AR1 = TRUE"
[1] "Group = NH, Subject = 63, Curve = 2, R2 = 0.994, AR1 = TRUE"
[1] "Group = NH, Subject = 64, Curve = 1, R2 = 0.985, AR1 = TRUE"
[1] "Group = NH, Subject = 64, Curve = 2, R2 = 0.975, AR1 = TRUE"
[1] "Group = NH, Subject = 65, Curve = 1, R2 = 0.996, AR1 = TRUE"
[1] "Group = NH, Subject = 65, Curve = 2, R2 = 0.984, AR1 = TRUE"
[1] "Group = NH, Subject = 66, Curve = 1, R2 = 0.992, AR1 = TRUE"
[1] "Group = NH, Subject = 66, Curve = 2, R2 = 0.984, AR1 = TRUE"
[1] "Group = NH, Subject = 68, Curve = 1, R2 = 0.989, AR1 = TRUE"
[1] "Group = NH, Subject = 68, Curve = 2, R2 = 0.969, AR1 = TRUE"
```

```
[1] "Group = NH, Subject = 71, Curve = 1, R2 = 0.991, AR1 = TRUE"
[1] "Group = NH, Subject = 71, Curve = 2, R2 = 0.979, AR1 = TRUE"
[1] "Group = NH, Subject = 73, Curve = 1, R2 = 0.983, AR1 = TRUE"
[1] "Group = NH, Subject = 73, Curve = 2, R2 = 0.988, AR1 = TRUE"
[1] "Group = NH, Subject = 85, Curve = 1, R2 = 0.979, AR1 = TRUE"
[1] "Group = NH, Subject = 85, Curve = 2, R2 = 0.95, AR1 = TRUE"
[1] "Group = NH, Subject = 95, Curve = 1, R2 = 0.985, AR1 = TRUE"
[1] "Group = NH, Subject = 95, Curve = 2, R2 = 0.991, AR1 = TRUE"
[1] "Group = NH, Subject = 98, Curve = 1, R2 = 0.956, AR1 = TRUE"
[1] "Group = NH, Subject = 98, Curve = 2, R2 = 0.991, AR1 = TRUE"
[1] "Group = NH, Subject = 100, Curve = 1, R2 = 0.988, AR1 = TRUE"
[1] "Group = NH, Subject = 100, Curve = 2, R2 = 0.985, AR1 = TRUE"
[1] "Group = NH, Subject = 102, Curve = 1, R2 = 0.984, AR1 = TRUE"
[1] "Group = NH, Subject = 102, Curve = 2, R2 = 0.994, AR1 = TRUE"
[1] "Group = NH, Subject = 105, Curve = 1, R2 = 0.987, AR1 = TRUE"
[1] "Group = NH, Subject = 105, Curve = 2, R2 = 0.973, AR1 = TRUE"
[1] "Group = NH, Subject = 106, Curve = 1, R2 = 0.994, AR1 = TRUE"
[1] "Group = NH, Subject = 106, Curve = 2, R2 = 0.99, AR1 = TRUE"
[1] "Group = CI, Subject = 2, Curve = 1, R2 = 0.974, AR1 = TRUE"
[1] "Group = CI, Subject = 2, Curve = 1, R2 = 0.967, AR1 = TRUE"
[1] "Group = CI, Subject = 3, Curve = 1, R2 = 0.996, AR1 = TRUE"
[1] "Group = CI, Subject = 3, Curve = 1, R2 = 0.991, AR1 = TRUE"
[1] "Group = CI, Subject = 4, Curve = 1, R2 = 0.962, AR1 = TRUE"
[1] "Group = CI, Subject = 4, Curve = 1, R2 = 0.989, AR1 = TRUE"
[1] "Group = CI, Subject = 6, Curve = 1, R2 = 0.953, AR1 = TRUE"
[1] "Group = CI, Subject = 6, Curve = 1, R2 = 0.988, AR1 = TRUE"
[1] "Group = CI, Subject = 10, Curve = 1, R2 = 0.974, AR1 = TRUE"
[1] "Group = CI, Subject = 10, Curve = 1, R2 = 0.952, AR1 = TRUE"
[1] "Group = CI, Subject = 16, Curve = 1, R2 = 0.98, AR1 = TRUE"
[1] "Group = CI, Subject = 16, Curve = 1, R2 = 0.992, AR1 = TRUE"
[1] "Group = CI, Subject = 17, Curve = 1, R2 = 0.981, AR1 = TRUE"
[1] "Group = CI, Subject = 17, Curve = 1, R2 = 0.991, AR1 = TRUE"
[1] "Group = CI, Subject = 19, Curve = 1, R2 = 0.982, AR1 = TRUE"
[1] "Group = CI, Subject = 19, Curve = 1, R2 = 0.935, AR1 = TRUE"
[1] "Group = CI, Subject = 22, Curve = 1, R2 = 0.987, AR1 = TRUE"
[1] "Group = CI, Subject = 22, Curve = 1, R2 = 0.986, AR1 = TRUE"
[1] "Group = CI, Subject = 23, Curve = 1, R2 = 0.983, AR1 = TRUE"
[1] "Group = CI, Subject = 23, Curve = 1, R2 = 0.983, AR1 = TRUE"
[1] "Group = CI, Subject = 24, Curve = 1, R2 = 0.965, AR1 = TRUE"
[1] "Group = CI, Subject = 24, Curve = 1, R2 = 0.973, AR1 = TRUE"
[1] "Group = CI, Subject = 29, Curve = 1, R2 = 0.992, AR1 = TRUE"
[1] "Group = CI, Subject = 29, Curve = 1, R2 = 0.986, AR1 = TRUE"
[1] "Group = CI, Subject = 30, Curve = 1, R2 = 0.963, AR1 = TRUE"
[1] "Group = CI, Subject = 30, Curve = 1, R2 = 0.844, AR1 = TRUE"
```

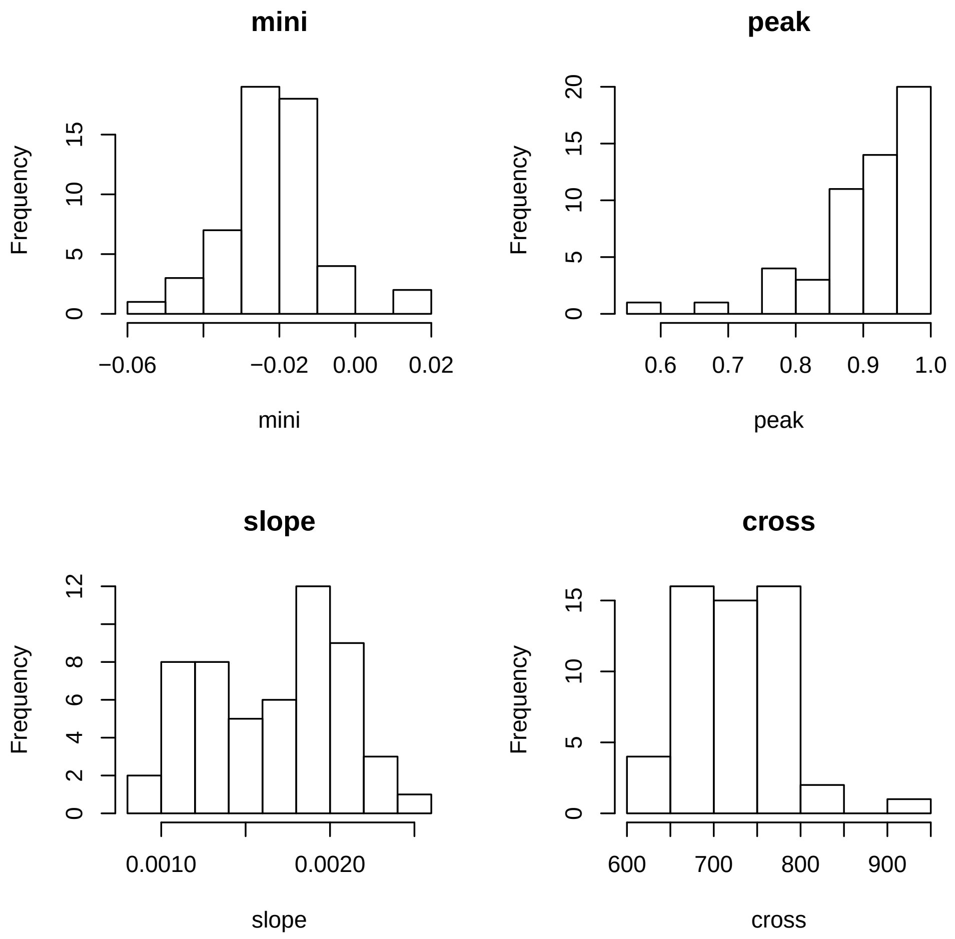
```
[1] "Group = CI, Subject = 41, Curve = 1, R2 = 0.969, AR1 = TRUE"
[1] "Group = CI, Subject = 41, Curve = 1, R2 = 0.975, AR1 = TRUE"
[1] "Group = CI, Subject = 48, Curve = 1, R2 = 0.974, AR1 = TRUE"
[1] "Group = CI, Subject = 48, Curve = 1, R2 = 0.978, AR1 = TRUE"
[1] "Group = CI, Subject = 72, Curve = 1, R2 = 0.954, AR1 = TRUE"
[1] "Group = CI, Subject = 72, Curve = 1, R2 = 0.976, AR1 = TRUE"
[1] "Group = CI, Subject = 74, Curve = 1, R2 = 0.987, AR1 = TRUE"
[1] "Group = CI, Subject = 74, Curve = 1, R2 = 0.977, AR1 = TRUE"
[1] "Group = CI, Subject = 75, Curve = 1, R2 = 0.97, AR1 = TRUE"
[1] "Group = CI, Subject = 75, Curve = 1, R2 = 0.988, AR1 = TRUE"
[1] "Group = CI, Subject = 76, Curve = 1, R2 = 0.988, AR1 = TRUE"
[1] "Group = CI, Subject = 76, Curve = 1, R2 = 0.988, AR1 = TRUE"
[1] "Group = CI, Subject = 79, Curve = 1, R2 = 0.982, AR1 = TRUE"
[1] "Group = CI, Subject = 79, Curve = 1, R2 = 0.989, AR1 = TRUE"
[1] "Group = CI, Subject = 81, Curve = 1, R2 = 0.99, AR1 = TRUE"
[1] "Group = CI, Subject = 81, Curve = 1, R2 = 0.978, AR1 = TRUE"
[1] "Group = CI, Subject = 82, Curve = 1, R2 = 0.987, AR1 = TRUE"
[1] "Group = CI, Subject = 82, Curve = 1, R2 = 0.975, AR1 = TRUE"
[1] "Group = CI, Subject = 83, Curve = 1, R2 = 0.98, AR1 = TRUE"
[1] "Group = CI, Subject = 83, Curve = 1, R2 = 0.884, AR1 = TRUE"
[1] "Group = CI, Subject = 84, Curve = 1, R2 = 0.989, AR1 = TRUE"
[1] "Group = CI, Subject = 84, Curve = 1, R2 = 0.977, AR1 = TRUE"
[1] "Group = CI, Subject = 86, Curve = 1, R2 = 0.986, AR1 = TRUE"
[1] "Group = CI, Subject = 86, Curve = 1, R2 = 0.952, AR1 = TRUE"
[1] "Group = CI, Subject = 90, Curve = 1, R2 = 0.99, AR1 = TRUE"
[1] "Group = CI, Subject = 90, Curve = 1, R2 = 0.962, AR1 = TRUE"
[1] "Group = CI, Subject = 94, Curve = 1, R2 = 0.983, AR1 = TRUE"
[1] "Group = CI, Subject = 94, Curve = 1, R2 = 0.931, AR1 = TRUE"
[1] "Group = CI, Subject = 99, Curve = 1, R2 = 0.987, AR1 = TRUE"
[1] "Group = CI, Subject = 99, Curve = 1, R2 = 0.986, AR1 = TRUE"
########################################
############### FITS ###################
########################################
AR1,       R2>=0.95   -- 103
AR1,     0.95>R2>=0.8 -- 5
AR1,       0.8>R2     -- 0
Non-AR1,   R2>=0.95   -- 0
Non-AR1, 0.95>R2>=0.8 -- 0
Non-AR1,   0.8>R2     -- 0
No Fit                -- 0
########################################

Next Steps: Check goodness of fits (ests.plot, subs.plot),
        refit bad fits (doubleGauss.refit),
bootstrap and t-test (doubleGauss.boot)
```

2.3 Checking Curve Fits

After fitting all the curves, the quality of these fits should be checked. There are a few ways to check this. The two most common are plotting the histograms for for each of the estimated parameter values (`ests.plot`) and plotting the observed curve and the estimated curve for each subject (`subs.plot`).

```
R> subs.plot(ci.1.out.1, "topleft")
```

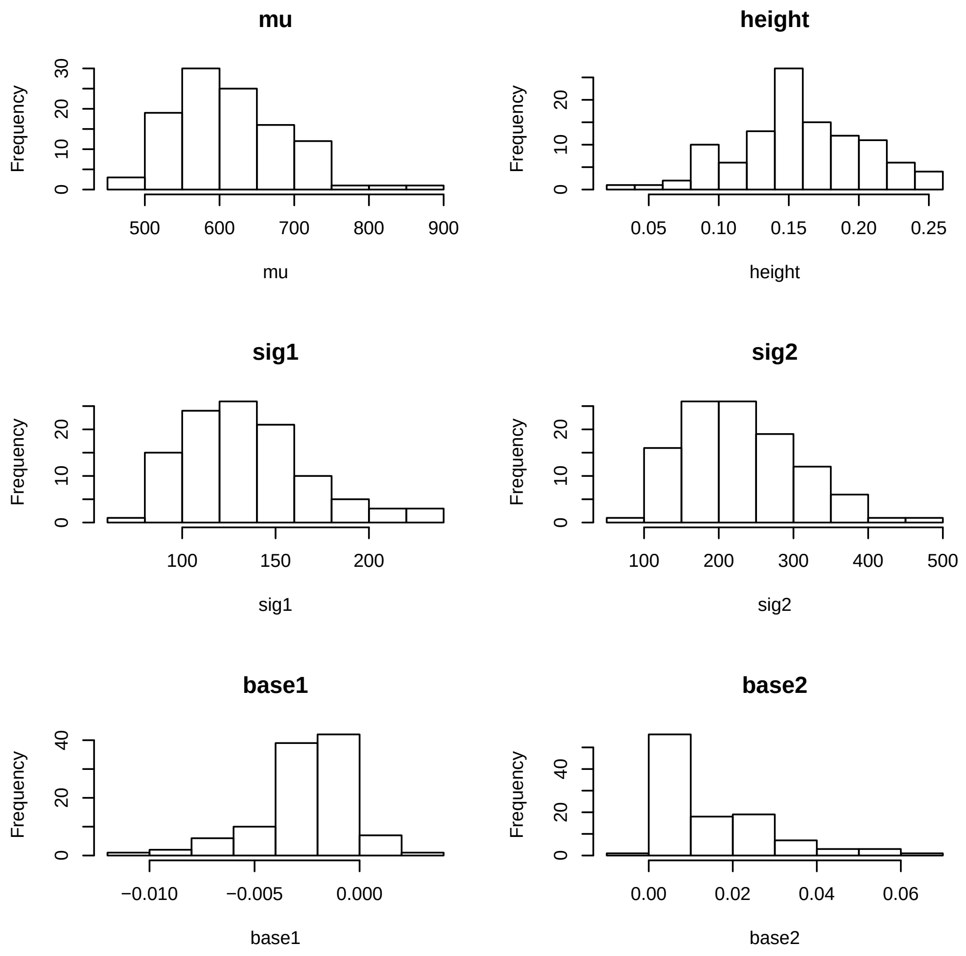
```
R> ests.plot(ci.1.out.1)
```



Looking at the second example, there are some curves that need attending to. There are too many plots to show in this vignette, but you can view these on your local machine by running the example code.

```
R> subs.plot(ci.2.out.1)
```

```
R> ests.plot(ci.2.out.1)
```



2.4 Refitting Bad Curves

As seen in example 2 (ci.2.*), sometimes the observed curves just aren't fit very well. In these examples, there are 2 approaches to take. The first: designate better starting parameters for the curve fit. When initial parameter estimates are too far from a reasonable curve fit, the curve fitter can't find a good fit. Coming up with new initial parameter estimates is typically done by looking at an individual curve and making a visual estimation as to the parameters (a bit more tougher to do with the double gauss). While the first approach typically failes to fix a bad fit, the second approach will be successful more often: relax the autocorrelation assumption of the errors. However, it will increase standard error of the estimates and reduces overall power. First we'll try refitting with given parameter estimates. For the first curve (subject 13, group 2, curve 2) we will use the following initial parameter estimates: $\mu = 650$, $H = 0.15$, $\sigma_1^2 = 150$, $\sigma_2^2 = 100$, $B_1 = 0$, and $B_2 = 0.03$. For the second curve (subject 23, group 2, curve 2) we will use the following initial parameter estimates: $\mu = 700$, $H = 0.10$, $\sigma_1^2 = 150$, $\sigma_2^2 = 100$, $B_1 = 0$, and $B_2 = 0.01$.

```
R> ci.2.out.1 <- doubleGauss.refit(ci.2.out.1,
+          subj = c(30, 83),
+          group = c("CI", "CI"),
+          curves = c(2, 2),
+          params = list(c(650, 0.15, 150, 100, 0, 0.03),
+                   c(700, 0.10, 150, 100, 0, 0.01)))

Subject =  30 , Group =  CI , Curve =  2 , Old R2 =  0.844 , New R2 =  0.845
         Old AR1 =  TRUE , New AR1 =  TRUE
Subject =  83 , Group =  CI , Curve =  2 , Old R2 =  0.884 , New R2 =  0.883
         Old AR1 =  TRUE , New AR1 =  TRUE
```

This didn't help the estimates at all. We'll now use the second approach, relaxing the error correlation assumption (`cor = FALSE`). We'll first try this using the newer matrix input (at this point requires the input of parameters) and then with the more traditional refitting approach.

```
R> refit.matrix <- matrix(NA, nrow = 2, ncol = 9)
R> refit.matrix[1,] <- c(30, "CI", 2, 650, 0.15, 150, 100, 0, 0.03)
R> refit.matrix[2,] <- c(83, "CI", 2, 700, 0.10, 150, 100, 0, 0.01)
R> refit.matrix

     [,1] [,2] [,3] [,4]  [,5]   [,6]  [,7]  [,8] [,9]
[1,] "30" "CI" "2"  "650" "0.15" "150" "100" "0"  "0.03"
[2,] "83" "CI" "2"  "700" "0.1"  "150" "100" "0"  "0.01"

R> ci.2.out.1 <- doubleGauss.refit(ci.2.out.1, cor = FALSE,
+          info.matrix = refit.matrix)

Subject =  30 , Group =  CI , Curve =  2 , Old R2 =  0.845 , New R2 =  0.954
         Old AR1 =  TRUE , New AR1 =  FALSE
Subject =  83 , Group =  CI , Curve =  2 , Old R2 =  0.883 , New R2 =  0.954
         Old AR1 =  TRUE , New AR1 =  FALSE

R> ci.2.out.1 <- doubleGauss.refit(ci.2.out.1 ,
+          subj = c(30, 83),
+          group = c("CI", "CI"),
+          curves = c(2, 2),
+          cor = c(FALSE, FALSE))

Subject =  30 , Group =  CI , Curve =  2 , Old R2 =  0.954 , New R2 =  0.954
         Old AR1 =  FALSE , New AR1 =  FALSE
Subject =  83 , Group =  CI , Curve =  2 , Old R2 =  0.954 , New R2 =  0.954
         Old AR1 =  FALSE , New AR1 =  FALSE
```

A summary of the quality of fits (the same summary setup as at the end of the fitting methods) can be output after doing refittings using the `printFits` method.

```
R> printFits(ci.2.out.1)

########################################
############### FITS ##################
########################################
AR1,       R2>=0.95   -- 103
AR1,     0.95>R2>=0.8 -- 3
AR1,       0.8>R2     -- 0
Non-AR1,   R2>=0.95   -- 2
Non-AR1, 0.95>R2>=0.8 -- 0
Non-AR1,   0.8>R2     -- 0
No Fit                -- 0
########################################
```

3 Bootstrapping and Analysis

3.1 Bootstrapping

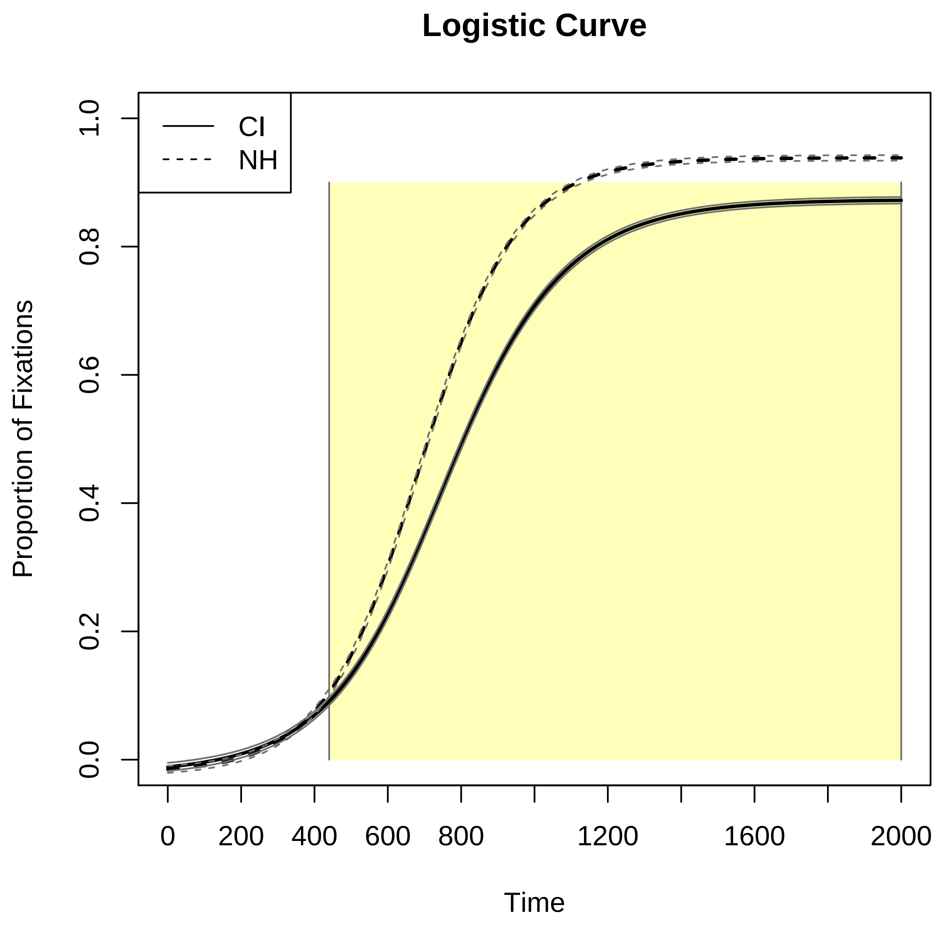
The final step is to bootstrap the group mean curve. At each bootstrap curve, every individual's curve is fit using their estimated parameters and standard deviation. These are then averaged across each group to get the bootstrap average. The following are the argument inputs:

- seed: the random number seed for reproducible results. By default, this is set based on the time at running the method
- alpha: the family wise overall P(TIE)
- paired: whether the subjects in both groupings are the same and paired t-tests should be used
- N.iter: the number of bootstrap iterations
- cores: the number of cores to use. Typically should be set to the number of available true physical cores minus 1
- p.adj: 'oleson' is our Bonferroni adjustment based on the correlation of the test statistics. FDR is another option that can be used by specifying 'fdr'
- test.spots: the time values to test at using the adjusted p-value. Our sample data is observed at every 4ms (from 0 to 2000). If we wanted to test at every 8ms we would set this to `seq(0, 2000, by = 8)`
- time.test: the individual time values to test at using an unadjusted p-value. These are typically only a couple individual spots of particular interest
- test.params: whether to test for differences in mean parameter estimates between the groups. This is a t-test using the bootstrapped means. If `paired = FALSE`, performs a 2-sample t-test with the equal variance assumption. If `paired = TRUE`, performs a paired t-test. This should only be set to `TRUE` when `diffs = FALSE`, or the results won't make much sense (tests for differences between group 1 curve 1 and group 2 curve 1).

Output includes the adjusted alpha value, the significant regions on the time course, and a graph with the bootstrapped group averages and areas of significance highlighted.

```
R> ci.1.out.2 <- logistic.boot(ci.1.out.1, seed = 123, cores = 2)

$alpha
    alpha alpha.adj   rho.est
   0.0500    0.0018    0.9974
```

```
$significant
     [,1] [,2]
[1,]  440 2000
```



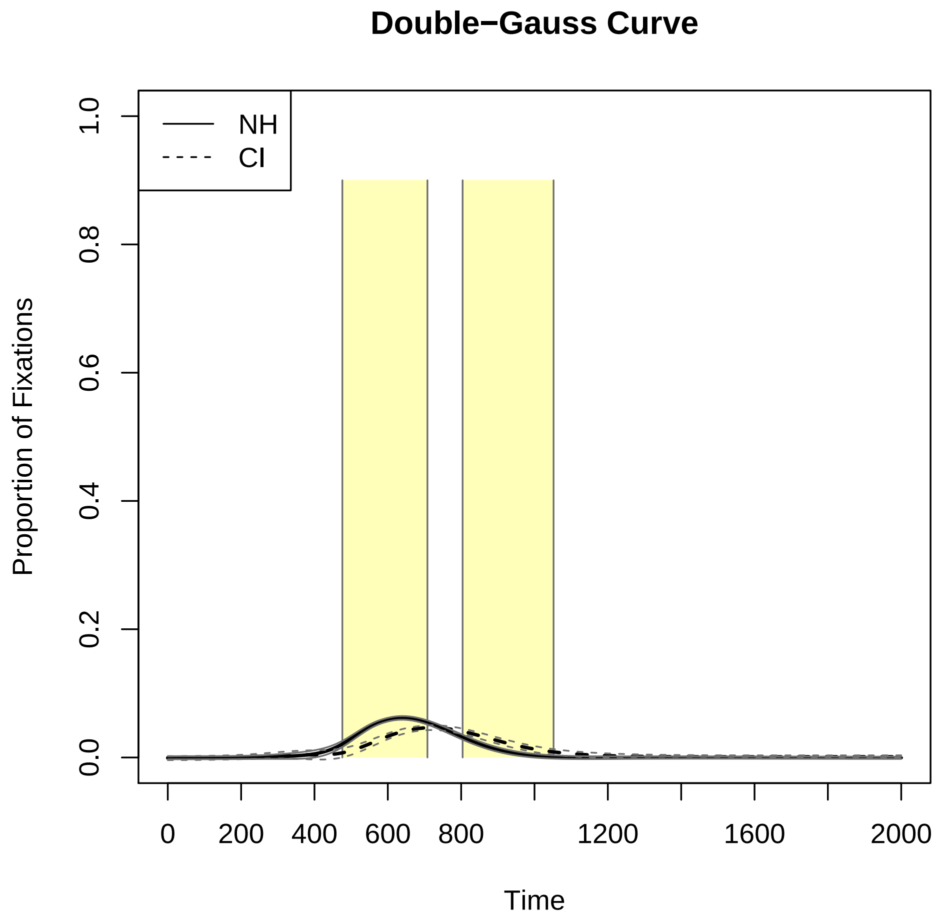
We will run this again (suppressing the output plot, which will look the same), but testing for difference in parameter estimates between the groups.

```
R> logistic.boot(ci.1.out.1, seed = 123, cores = 2, test.params = TRUE)

######################
## Parameter Tests  ##
## 2 Sample t-test  ##
######################
Mini -- Diff: -7e-04, t: -0.155, SE: 0.004, df: 52, p: 0.8775
Peak -- Diff: -0.065, t: -18.797, SE: 0.003, df: 52, p: 0
Slope -- Diff: -4e-04, t: -25.95, SE: 0, df: 52, p: 0
Cross -- Diff: 67.5391, t: 24.254, SE: 2.785, df: 52, p: 0

$alpha
```

```
    alpha alpha.adj   rho.est
   0.0500    0.0018    0.9975

$significant
     [,1] [,2]
[1,]  440 2000
```

For the next example we will also run a non-adjusted test at the time point 900.

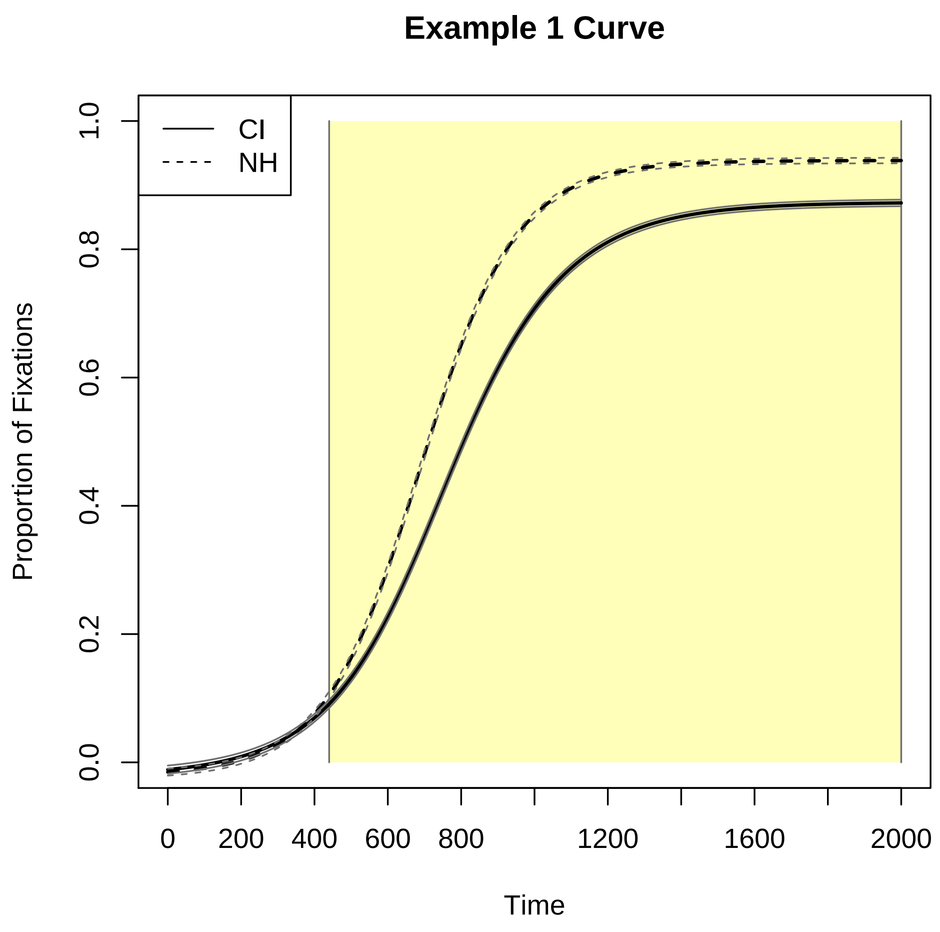
```
R> ci.2.out.2 <- doubleGauss.boot(ci.2.out.1, seed = 123, cores = 2, time.test = 900)

######################
## Individual Tests ##
######################
Test # = 1 --- Time = 900
Mean Diff = -0.0139 --- SE = 0.003
t = -4.69 --- DF = 52 --- p < 0.0001
Pooled SD = 0.0109 --- Cohen's d = -1.3

$alpha
    alpha alpha.adj   rho.est
   0.0500    0.0031    0.9993

$significant
     [,1] [,2]
[1,]  476  708
[2,]  804 1052
```


Double−Gauss Curve
NH
CI
Proportion of Fixations
0.0
0.2
0.4
0.6
0.8
1.0
0
200
400
600
800
1200
1600
2000
Time

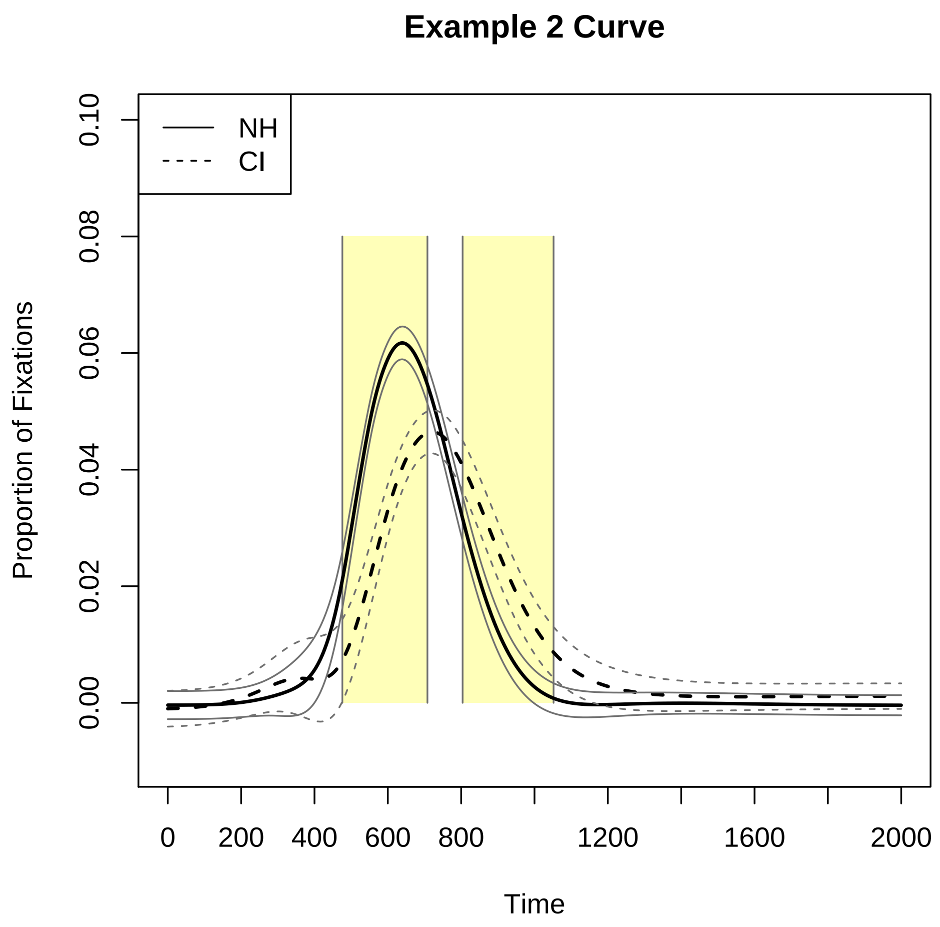
3.2 Replotting

There are no options for modifying the plotting properties in the bootstrap step, as we wanted those options to be focused on the computational issues. Typically we will want to adjust the size of the y-axis and, by extension, the size of the highlighting box in terms of the y-axis values.

```
R> replot(ci.1.out.2, bucket.lim = c(0, 1), main = "Example 1 Curve")
```



```
R> replot(ci.2.out.2, ylim = c(-0.01, 0.1), bucket.lim = c(0, 0.08),
+           main = "Example 2 Curve")
```



References

[1] 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, 2014.

[2] 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*, 2015.