Memory 8/15 Progress

Eric Bridgeford

August 15, 2016

Overview of Dataset

Experimental Setup

The dataset analyzed was collected by Dr. Kara Blacker et al. between January and June, 2015. A total of 32 subjects were analyzed, for 5 runs per subject. Each subject has 1 anatomical scan provided, which is useful for alignment during anatomical preprocessing. During the experiments, the subjects experienced two types of trials (location and relation) and two loads (1 or 3). The trial type and load order are pseudo randomized for each subject (each trial is equally likely to be followed by any other trial type), and subjects receive the exact same trial order for each of the 6 runs.

The trial period is structured as follows: - fixation period (400 ms): subjects fixate on center cross - trial cue (500 ms): cue indicates location or relation trial - sample array (500 ms): colored circles shown - delay (8000 ms): working memory maintance period, when only a fixation cross is shown - test array (1500 ms): subjects respond based on a comparison of the sample array in memory and the presented test array - feedback (100 ms): indicates correct, incorrect, or too slow of a response

Each run contains 48 trials total, and each timestep is 2 seconds in duration, so each run represents 283 timesteps. The cue period is considered to be the timestep in which the cue begins, and the delay period is considered to be the next 4 timesteps. Only correct trials are considered for the purposes of this investigation.

Data Analysis Framework

Previously, we used a non-consistent assumption for our expected hemodynamic delays. We assumed that the cue period would last for the period in which an onset occurs and the step immediately following, and that the delay period would last for the next 4 steps. Here, we consider a single response function for the hemodynamic delay in which we assume that our maximal signal for a particular condition will occur in either the 4 steps following a condition (the simple model) or a weighted combination of the signal values of the next 6 timesteps (where the 2nd and 3rd weight most heavily). Choice of function obviously does not have a significant impact on the results, as can be seen below (as we do not explicitly have onsets for rest periods, we do not compute a rest matrix here, but just rely on the overall matrix instead).

Rectangular Case

For each condition, consider that we have a spike train of condition onset periods, where the spike train receives a value of 1 where every condition begins. We then convolve this with our proportion of evoked signal function, which gives us the contribution of the MR signal at a particular time point to the condition being observed. In the simple case, we might assume a rectangular window with a lag of 1 timestep, lasting for a period of 4 timesteps, as follows:

```
require('ggplot2')
```

```
## Loading required package: ggplot2
```

```
require('reshape2')
```

```
## Loading required package: reshape2
```

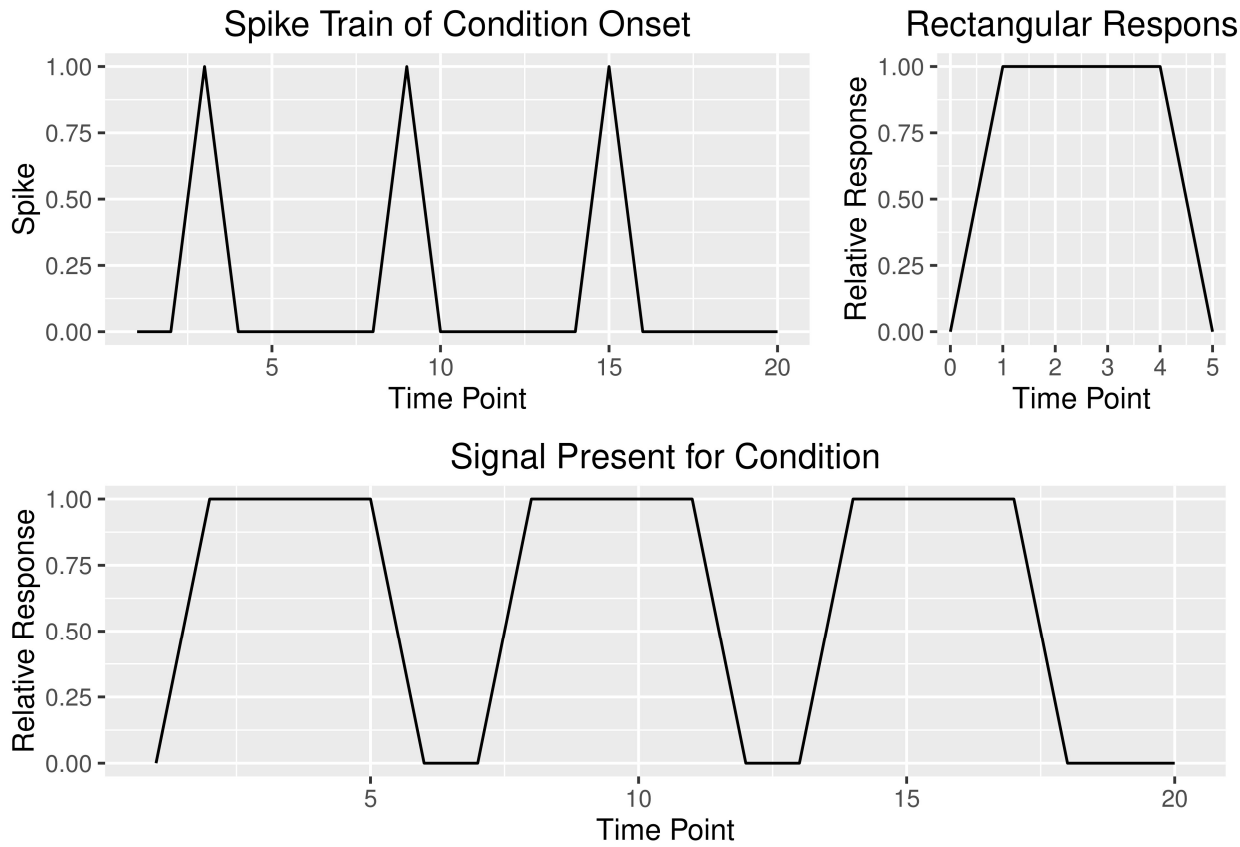
```r
require('Rmisc')
```

```
## Loading required package: Rmisc
```

```
## Loading required package: lattice
```

```
## Loading required package: plyr
```

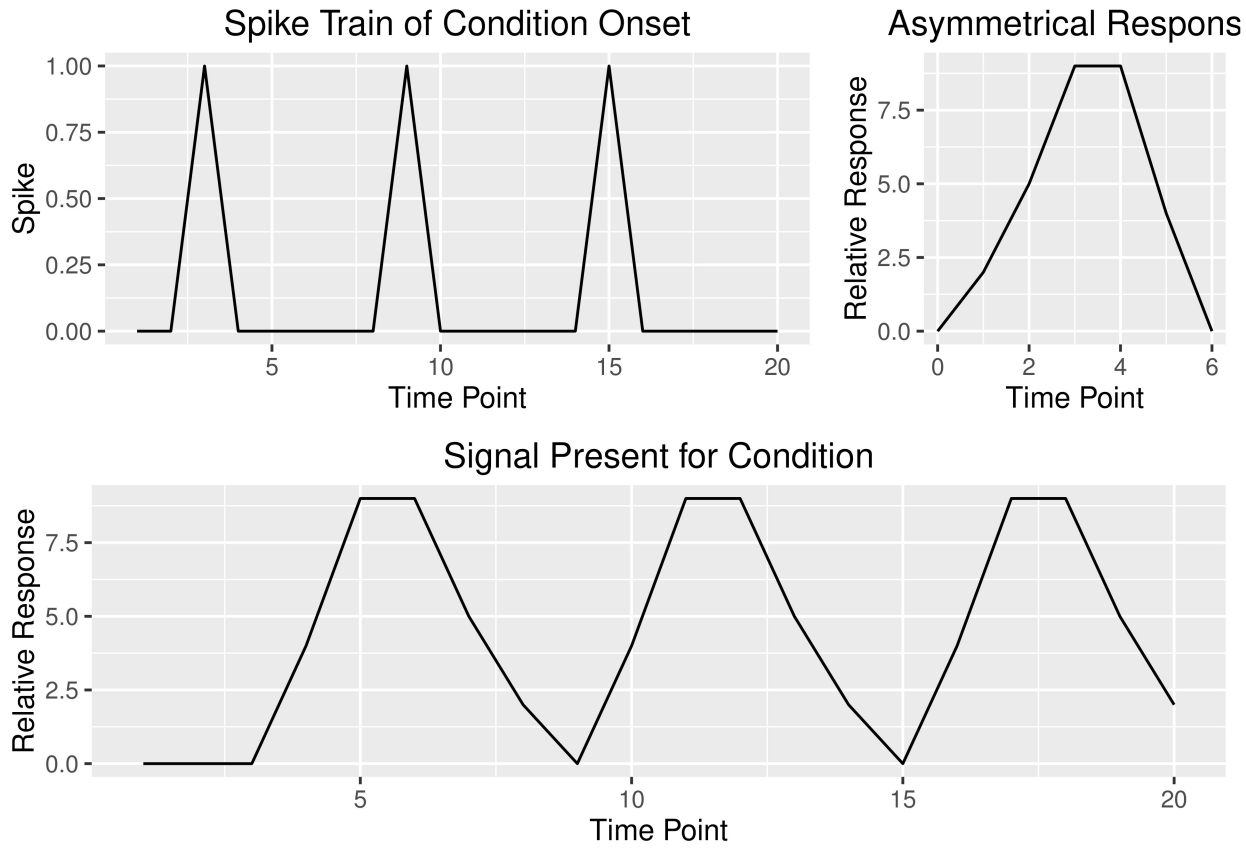
```r
conv_fx <- function(x, y) {
  return(round(convolve(x, y, type="open")[1:length(x)]))
}

spike_cond <- c(0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0)

fx <- c(0, 1, 1, 1, 1, 0)

res <- rev(conv_fx(spike_cond, fx))

func <- ggplot(data = melt(fx), aes(x = 0:5, y = value)) + geom_line() +
  xlab('Time Point') + ylab('Relative Response') + ggtitle('Rectangular Response')
sp <- ggplot(data = melt(spike_cond), aes(x = 1:length(spike_cond), y=value)) + geom_line() +
  xlab('Time Point') + ylab('Spike') + ggtitle('Spike Train of Condition Onset')
conv <- ggplot(data = melt(res), aes(x = 1:length(spike_cond), y=value)) + geom_line() +
  xlab('Time Point') + ylab('Relative Response') + ggtitle('Signal Present for Condition')

multiplot(plotlist=list(sp, func, conv), layout=matrix(c(1,1,2,3,3,3), nrow=2, byrow=TRUE))
```



Asymmetrical Case

We might follow with a more complex version of the response function (drawn from Functional Imaging of Cognition, Seminars in Neurology, D'Esposito et al.), shown below:

```
fx <- rev(c(0, 4, 9, 9, 5, 2, 0))

res <- conv_fx(spike_cond, fx)

func <- ggplot(data = melt(fx), aes(x = 0:(length(fx)-1), y = value)) + geom_line() +
  xlab('Time Point') + ylab('Relative Response') + ggtitle('Asymmetrical Response')
sp <- ggplot(data = melt(spike_cond), aes(x = 1:length(spike_cond), y=value)) + geom_line() +
  xlab('Time Point') + ylab('Spike') + ggtitle('Spike Train of Condition Onset')
conv <- ggplot(data = melt(res), aes(x = 1:length(spike_cond), y=value)) + geom_line() +
  xlab('Time Point') + ylab('Relative Response') + ggtitle('Signal Present for Condition')

multiplot(plotlist=list(sp, func, conv), layout=matrix(c(1,1,2,3,3,3), nrow=2, byrow=TRUE))
```



Since we are constrained by the dimensions of the number of timesteps in the dataset, we restrict our analysis to this range, and do not consider any "overflow" time points.

Exploratory Correlation Analysis

To analyze correlation, we consider our predicted response function from above to correspond to the "amount" which each timestep matters in determining how correlated two rois are for a particular condition. We can easily modify simple pairwise correlation to model this relationship, by simply considering our proportion of our response to be the number of times we should count each timepoint. In the rectangular case, the simply

corresponds to removing time points with a value of 0, and concatenating all time points with a value of 1. For the more complex case, if a particular time point expects a relative contribution of "5", we simply count that time point 5 times consecutively.

Simple Case

```
require('testthat')
```

```
## Loading required package: testthat
```

```
require('Rmisc')
require('ggplot2')
require('reshape2')
require('abind')
```

```
## Loading required package: abind
```

```
require('plyr')
source('../Code/R/drivers/plot_one_mtx.R')
source('../Code/R/drivers/convolve_trial_onsets.R')
source('../Code/R/drivers/get_trial_onsets.R')
source('../Code/R/drivers/convolve_trial_onsets.R')
source('../Code/R/drivers/timeseries_onset_plot.R')
source('../Code/R/drivers/extract_timepoint_order.R')
source('../Code/R/drivers/extract_ordered_timeseries.R')
source('../Code/R/drivers/conditional_correlation.R')
source('../Code/R/drivers/wilcox_test_conditions.R')
source('../Code/R/drivers/plot_corr_wilcox.R')
source('../Code/R/drivers/plot_condition_corr.R')
source('C:/Users/ebrid/Documents/GitHub/ugrad-data-design-team-0/data_processing/Rutils/open_timeseries
source('C:/Users/ebrid/Documents/GitHub/ugrad-data-design-team-0/data_processing/Rutils/obs2zsc.R')
options(warn=-1)
infopath <- '../data/subjects/information/'
## Load Timeseries ------------------------------------------------------------
tspath <- '../data/subjects/maxprob_vol_lh_2mm/'
fnames <- list.files(tspath, pattern="\\.rds", full.names=TRUE)
left_tsobj <- open_timeseries(fnames, dataset_pos=1, sub_pos=2, run_pos=4)
```

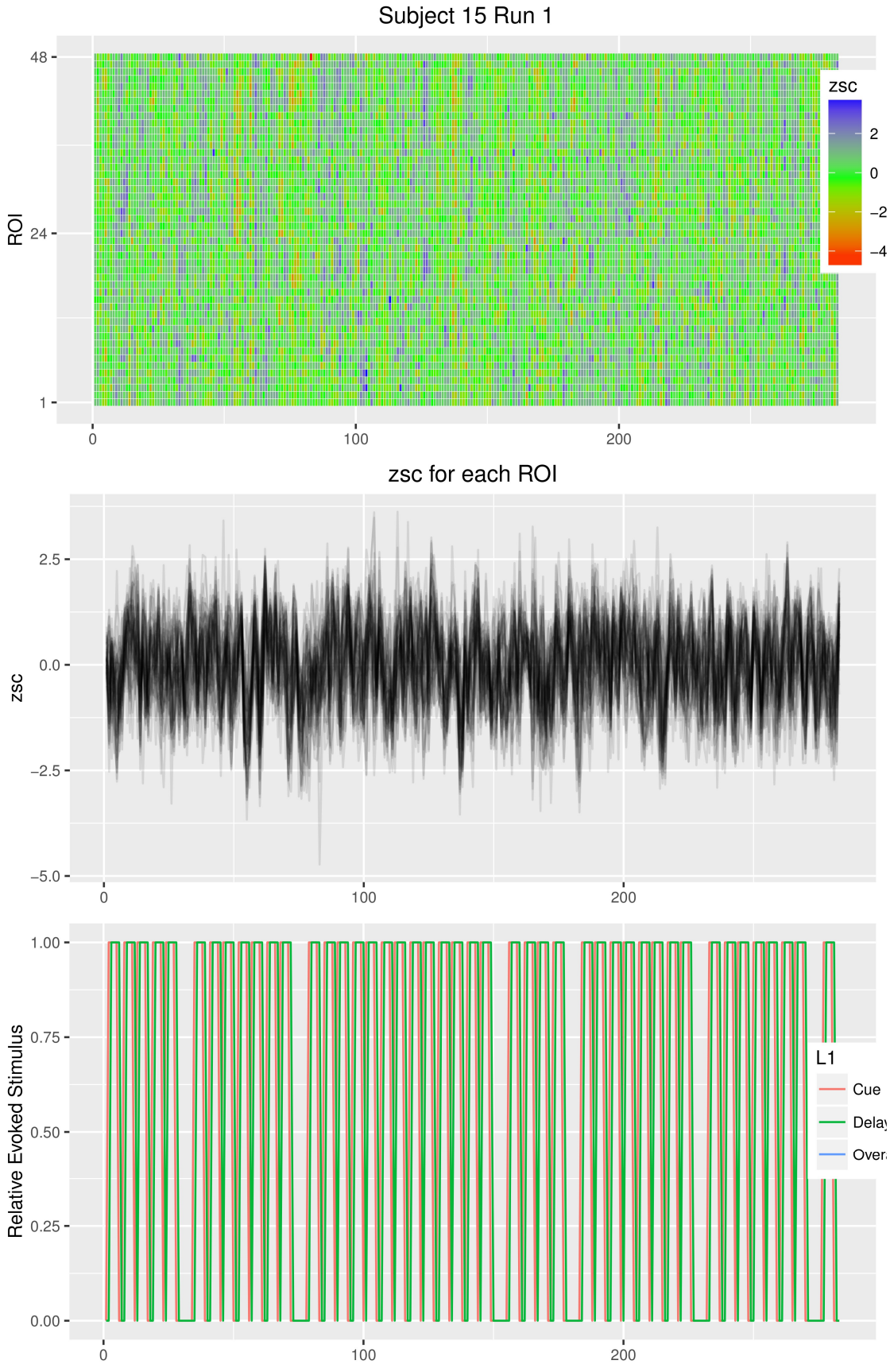
```
## [1] "opening timeseries..."
```

```
left_ts <- left_tsobj[["ts"]]
subs <- left_tsobj[["subjects"]]
runs <- left_tsobj[["runs"]]

tspath <- '../data/subjects/maxprob_vol_rh_2mm/'
fnames <- list.files(tspath, pattern="\\.rds", full.names=TRUE)
right_tsobj <- open_timeseries(fnames, dataset_pos=1, sub_pos=2, run_pos=4)
```

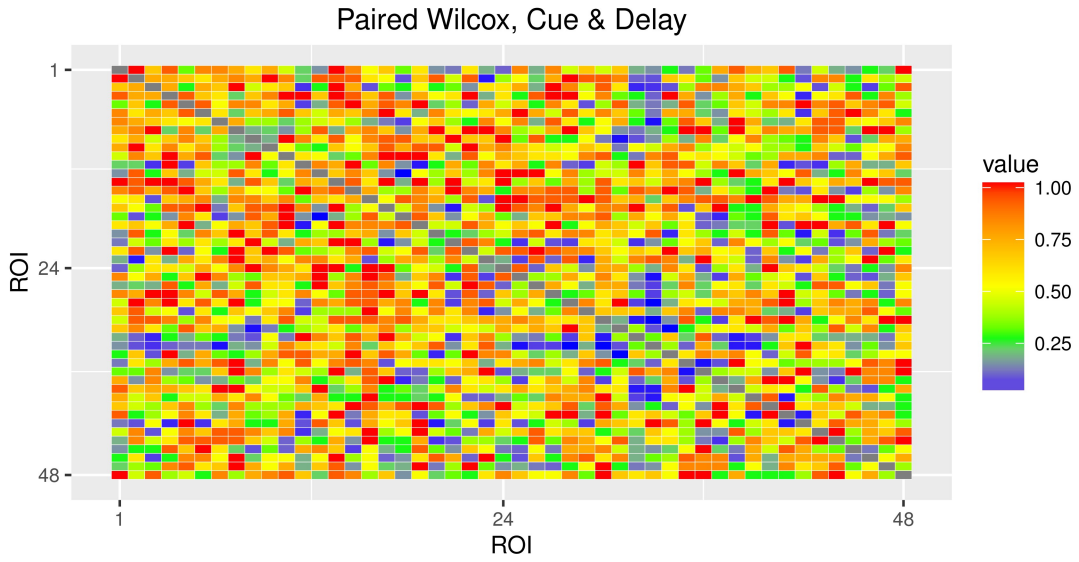
```
## [1] "opening timeseries..."
```

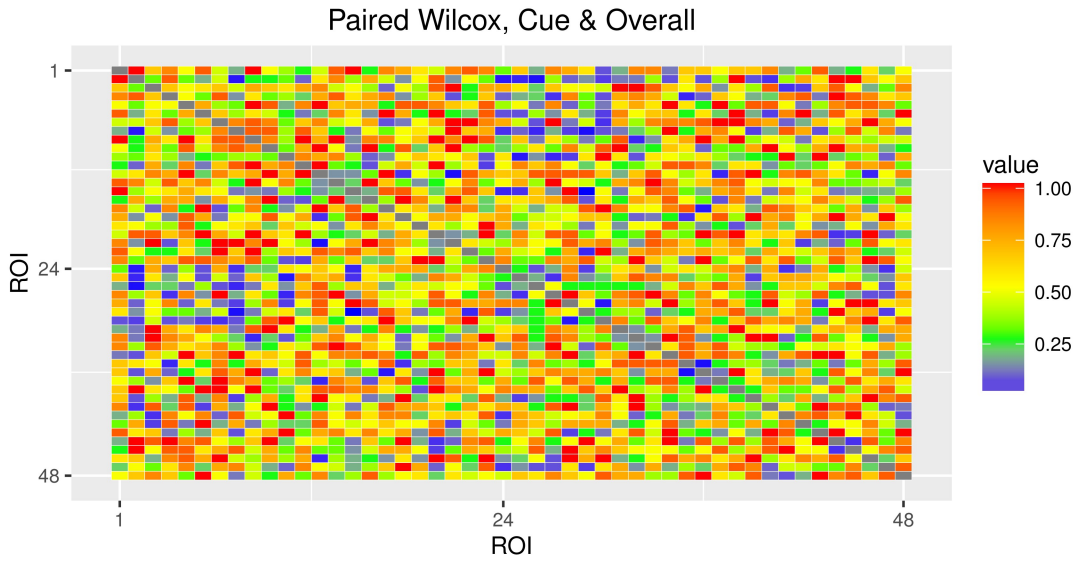
```
right_ts <- right_tsobj[["ts"]]
expect_true(isTRUE(all.equal(right_tsobj[["subjects"]], subs)))
expect_true(isTRUE(all.equal(runs, right_tsobj[["runs"]])))


## Bind the timeseries we have info for ---------------------------------------
```

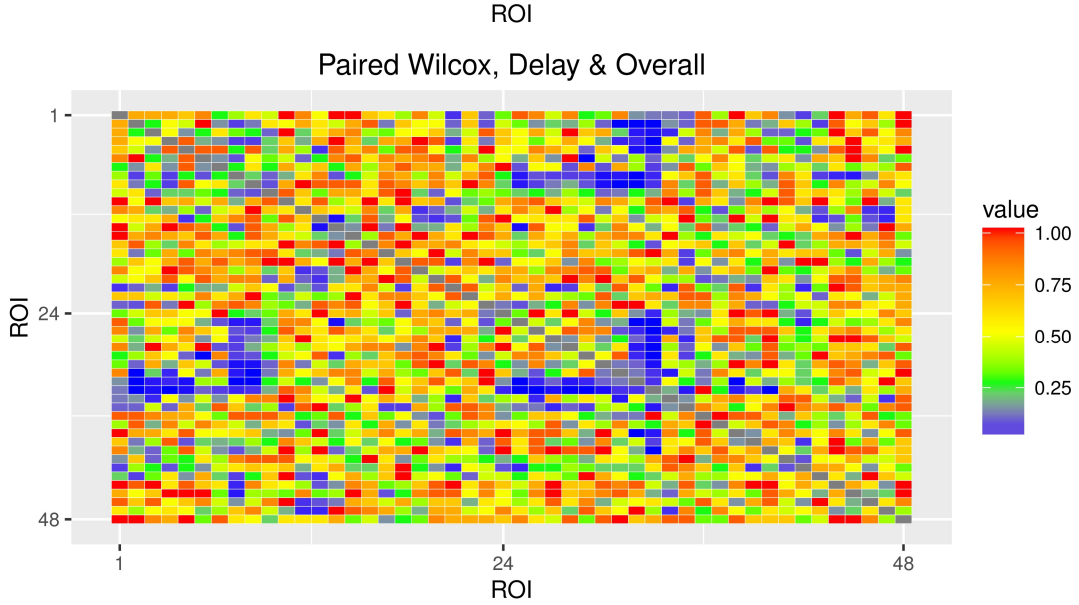
```r
ts <- sapply(names(left_ts), function(x) abind(left_ts[[x]], right_ts[[x]], along=2),
             simplify=FALSE, USE.NAMES=TRUE)

ts <- obs2zsc(ts)

## Condition Onset/Correlation ------------------------------------------------
cond_key <- c("Cue", "Delay", "Incorrect")
onsets <- get_trial_onsets(infopath, ts, subs, runs, cond_key=cond_key)

# Rectangular window
fx <- c(0,1,1,1,1,0)
# Asymmetric Window
#fx <- rev(c(0, 4, 9, 9, 5, 2, 0))

pi_vecs <- convolve_trial_onsets(onsets, fx)
pi_vecs <- sapply(pi_vecs, function(sub) {
   return(list(Cue=sub$Cue, Delay=sub$Delay, Overall=rep(1, (length(sub$Cue)))))
}, USE.NAMES=TRUE, simplify=FALSE)
cond_key <- c("Cue", "Delay", "Overall")

ordered_timesteps <- extract_timepoint_order(pi_vecs)
# Plot the timeseries with the pi vectors.
value_multiplots <- timeseries_onset_plot(ts, pi_vecs, subs, runs, textsize=10, type="zsc")

ordered_timeseries <- extract_ordered_timeseries(ts, ordered_timesteps)

multiplot(plotlist=value_multiplots$`1`, layout=matrix(c(1,2,3), nrow=3, byrow=TRUE))
```


Subject 15 Run 1
ROI
48
24
1
0
100
200
zsc
2
0
−2
−4
zsc for each ROI
zsc
2.5
0.0
−2.5
−5.0
0
100
200
Relative Evoked Stimulus
1.00
0.75
0.50
0.25
0.00
0
100
200
L1
Cue
Delay
Over

```r
# calculate correlation based on the condition, where time points
# that should see a higher response are factored in most
corr_cond_ts <- conditional_correlation(ordered_timeseries)

corr_cond_plots <- plot_condition_corr(corr_cond_ts, subs, runs, textsize=10)

# reorder so that the conditions are our L1 keys, and subjects are L2
corr_cond_reordered <- sapply(cond_key, function(cond) sapply(names(corr_cond_ts), function(sub) corr_c
                                                              USE.NAMES=TRUE, simplify=FALSE),
                              USE.NAMES=TRUE, simplify=FALSE)


## Wilcoxon Tests ------------------------------------------------------------

corr_wilcox <- wilcox_test_conditions(corr_cond_reordered, subs, runs)

wilcox_plots <- plot_corr_wilcox(corr_wilcox, textsize=10)

multiplot(plotlist=wilcox_plots$overall, layout=matrix(c(1,2,3), nrow=3, byrow=TRUE))
```


Paired Wilcox, Cue & Delay
ROI
1
24
48
value
1.00
0.75
0.50
0.25


Paired Wilcox, Cue & Overall
ROI
1
24
48
value
1.00
0.75
0.50
0.25


Paired Wilcox, Delay & Overall
ROI
1
24
48
value
1.00
0.75
0.50
0.25

Less Simple Case

```
require('testthat')
require('Rmisc')
require('ggplot2')
require('reshape2')
require('abind')
require('plyr')
source('../Code/R/drivers/plot_one_mtx.R')
source('../Code/R/drivers/convolve_trial_onsets.R')
source('../Code/R/drivers/get_trial_onsets.R')
source('../Code/R/drivers/convolve_trial_onsets.R')
source('../Code/R/drivers/timeseries_onset_plot.R')
source('../Code/R/drivers/extract_timepoint_order.R')
source('../Code/R/drivers/extract_ordered_timeseries.R')
source('../Code/R/drivers/conditional_correlation.R')
source('../Code/R/drivers/wilcox_test_conditions.R')
source('../Code/R/drivers/plot_corr_wilcox.R')
source('../Code/R/drivers/plot_condition_corr.R')
source('C:/Users/ebrid/Documents/GitHub/ugrad-data-design-team-0/data_processing/Rutils/open_timeseries
source('C:/Users/ebrid/Documents/GitHub/ugrad-data-design-team-0/data_processing/Rutils/obs2zsc.R')

options(warn=-1)
infopath <- '../data/subjects/information/'
## Load Timeseries ------------------------------------------------------------
tspath <- '../data/subjects/maxprob_vol_lh_2mm/'
fnames <- list.files(tspath, pattern="\\.rds", full.names=TRUE)
left_tsobj <- open_timeseries(fnames, dataset_pos=1, sub_pos=2, run_pos=4)
```

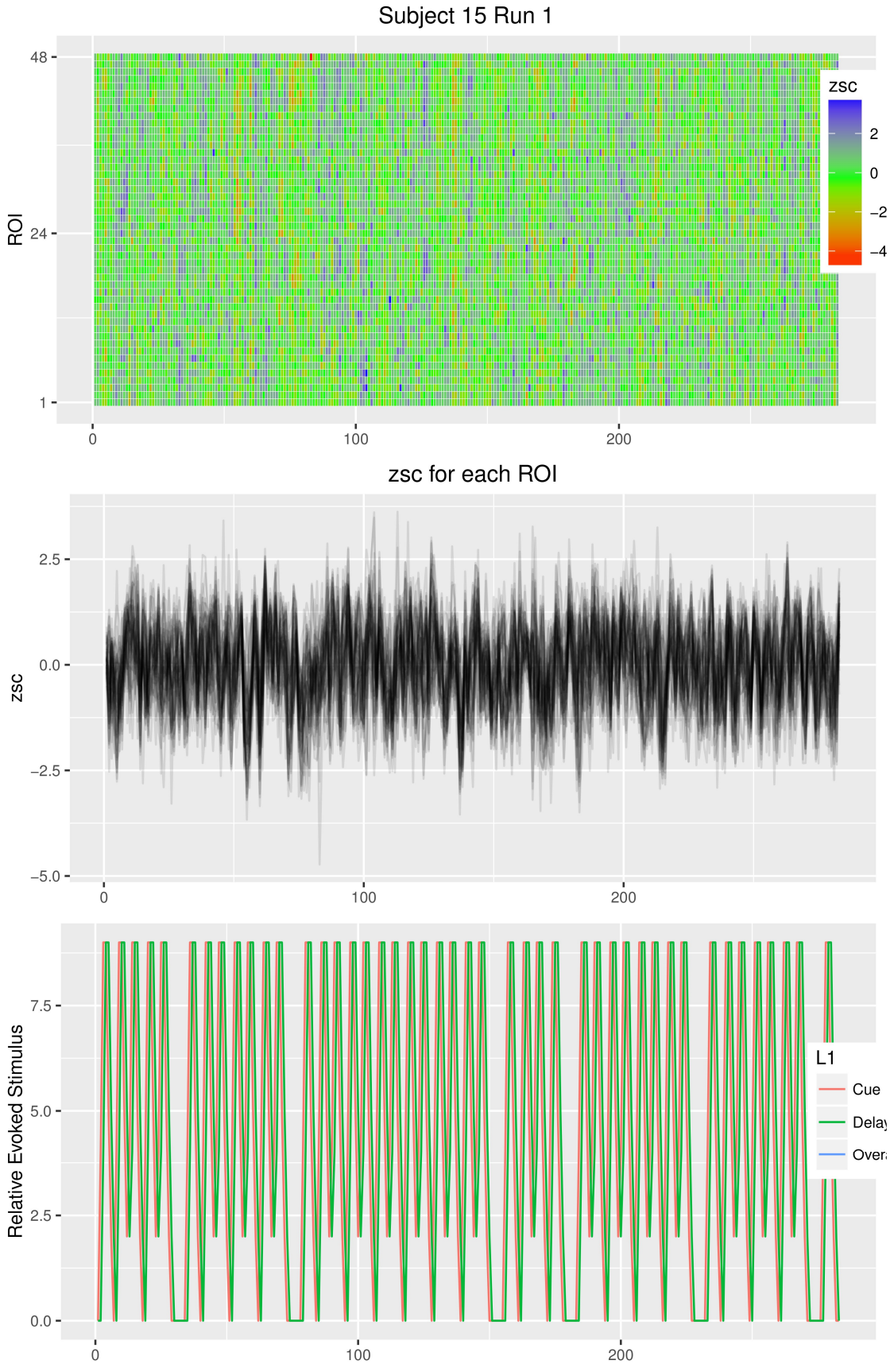
```
## [1] "opening timeseries..."
```

```
left_ts <- left_tsobj[["ts"]]
subs <- left_tsobj[["subjects"]]
runs <- left_tsobj[["runs"]]

tspath <- '../data/subjects/maxprob_vol_rh_2mm/'
fnames <- list.files(tspath, pattern="\\.rds", full.names=TRUE)
right_tsobj <- open_timeseries(fnames, dataset_pos=1, sub_pos=2, run_pos=4)
```

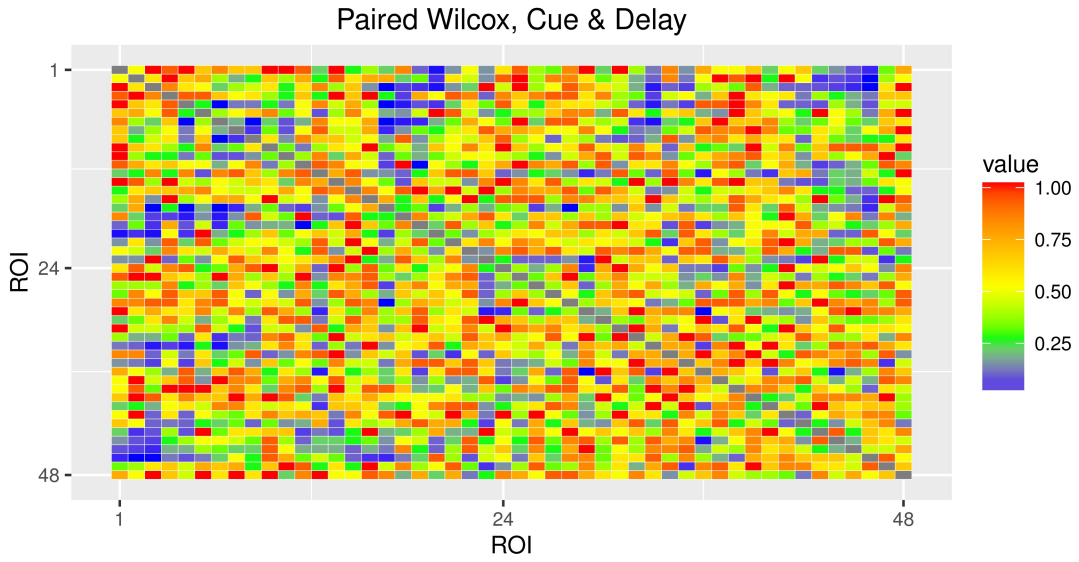
```
## [1] "opening timeseries..."
```

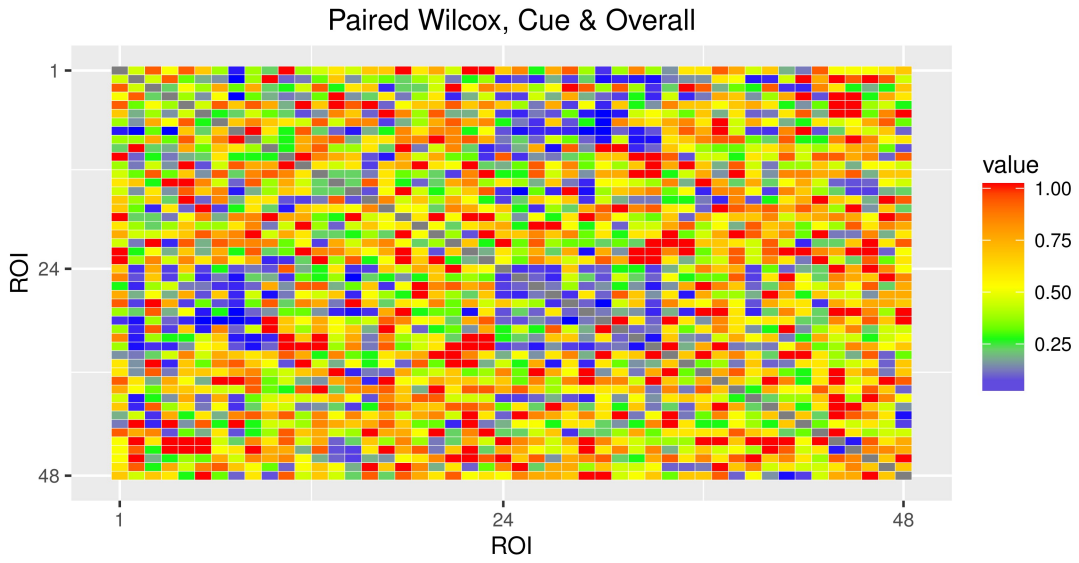
```
right_ts <- right_tsobj[["ts"]]
expect_true(isTRUE(all.equal(right_tsobj[["subjects"]], subs)))
expect_true(isTRUE(all.equal(runs, right_tsobj[["runs"]])))


## Bind the timeseries we have info for ---------------------------------------

ts <- sapply(names(left_ts), function(x) abind(left_ts[[x]], right_ts[[x]], along=2),
             simplify=FALSE, USE.NAMES=TRUE)

ts <- obs2zsc(ts)

## Condition Onset/Correlation ------------------------------------------------
cond_key <- c("Cue", "Delay", "Incorrect")
onsets <- get_trial_onsets(infopath, ts, subs, runs, cond_key=cond_key)
```

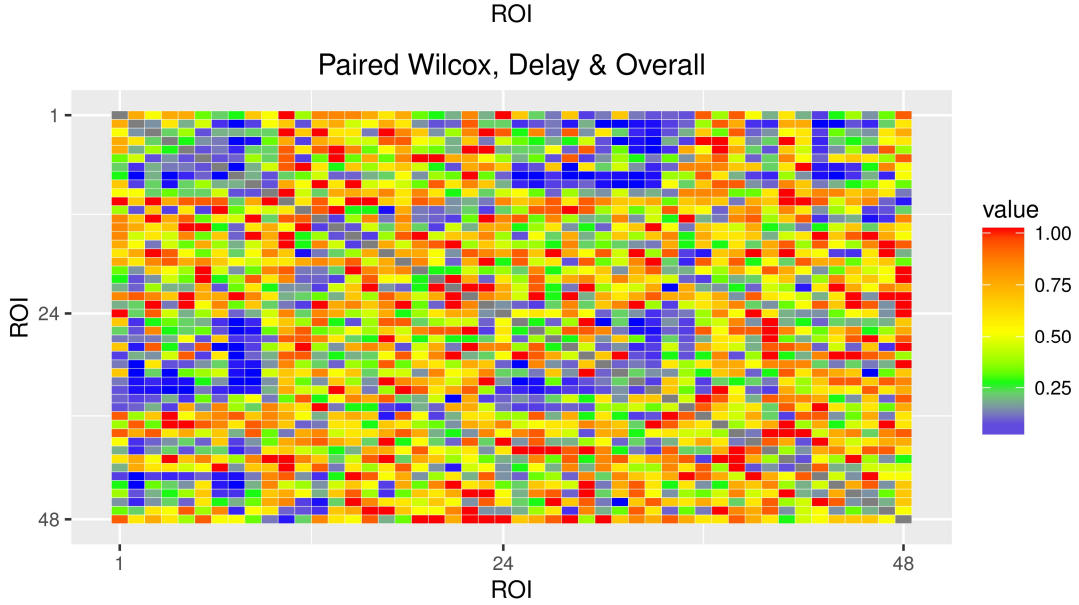
```r
# Rectangular window
#fx <- c(0,1,1,1,1,0)
# Asymmetric Window
fx <- rev(c(0, 4, 9, 9, 5, 2, 0))

pi_vecs <- convolve_trial_onsets(onsets, fx)
pi_vecs <- sapply(pi_vecs, function(sub) {
   return(list(Cue=sub$Cue, Delay=sub$Delay, Overall=rep(1, (length(sub$Cue)))))
}, USE.NAMES=TRUE, simplify=FALSE)
cond_key <- c("Cue", "Delay", "Overall")

ordered_timesteps <- extract_timepoint_order(pi_vecs)
# Plot the timeseries with the pi vectors.
value_multiplots <- timeseries_onset_plot(ts, pi_vecs, subs, runs, textsize=10, type="zsc")

ordered_timeseries <- extract_ordered_timeseries(ts, ordered_timesteps)
multiplot(plotlist=value_multiplots$`1`, layout=matrix(c(1,2,3), nrow=3, byrow=TRUE))
```


Subject 15 Run 1
ROI
48
24
1
0
100
200
zsc
2
0
−2
−4
zsc for each ROI
zsc
2.5
0.0
−2.5
−5.0
0
100
200
Relative Evoked Stimulus
7.5
5.0
2.5
0.0
0
100
200
L1
Cue
Delay
Over

```r
# calculate correlation based on the condition, where time points
# that should see a higher response are factored in most
corr_cond_ts <- conditional_correlation(ordered_timeseries)

corr_cond_plots <- plot_condition_corr(corr_cond_ts, subs, runs, textsize=10)

# reorder so that the conditions are our L1 keys, and subjects are L2
corr_cond_reordered <- sapply(cond_key, function(cond) sapply(names(corr_cond_ts), function(sub) corr_c
                                                              USE.NAMES=TRUE, simplify=FALSE),
                              USE.NAMES=TRUE, simplify=FALSE)


## Wilcoxon Tests ------------------------------------------------------------

corr_wilcox <- wilcox_test_conditions(corr_cond_reordered, subs, runs)

wilcox_plots <- plot_corr_wilcox(corr_wilcox, textsize=10)

multiplot(plotlist=wilcox_plots$overall, layout=matrix(c(1,2,3), nrow=3, byrow=TRUE))
```


Paired Wilcox, Cue & Delay
ROI
1
24
48
value
1.00
0.75
0.50
0.25


Paired Wilcox, Cue & Overall
ROI
1
24
48
value
1.00
0.75
0.50
0.25


Paired Wilcox, Delay & Overall
ROI
1
24
48
value
1.00
0.75
0.50
0.25

Obvious limitations: choice of function is INCREDIBLY important. Goal: find some way of analyzing that gives relatively similar answers, independent of how "correct" our model choice for the hemodynamic response is. Maybe EM holds answer?