Advanced Plotting

Luke Chang

Last Revised July 19, 2010

The graphing capabilities of `R` are virtually unlimited (see http://addictedtor.free.fr/graphiques/ for some examples). In this section we will examine several different types of graphs that I regularly use in my own research. We will also introduce an additional graphing package (e.g., `ggplot2`) that can add even greater flexibility to the graphing options in `R` .

1 Plotting Mixed Models

1.1 Plotting Linear Mixed Models

Often when graphically depicting the results of a linear mixed model we are interested in seeing (1) the raw data, (2) individual regression lines (i.e., the random effects), and (3) the group regression line (i.e., the fixed effect factor parameter estimates). This type of graph can be a nightmare to create in other programs such as SPSS or even excel. However, it can be created with relative ease using the basic `R` graphing tools. For this example we will return to the decision conflict example from the Ultimatum Game dataset that we previously discussed in the mixed model section. We will use the basic varying intercept and varying slope Offer amount regression on RT. Here we will rerun the same model (model 3 from the other section) and will extract both the fixed effect parameters using the `fixef()` command and the random effects coefficients using the `ranef()` command. As we discussed earlier in the mixed model section, the random coefficients are random deviations centered around the fixed effects, so we must add the fixed effects to the random effects for plotting purposes. This is accomplished by selecting the columns for the intercept and slope from `ranParam` and adding them to the corresponding columns of the

fixParam to create the params object. We will use a for loop to plot a regression line for each subject by iterating through the subNum vector, which contains all of the subject numbers and using abline to pull the intercept and slope for each iteration i of the loop.

```
> library(lme4)
> data<-read.table(paste(website,"UG_Data.txt",sep=""),
    header=TRUE,na.strings=999999)
> data<-na.exclude(data)
> data$Condition<-relevel(data$Condition,ref="Computer")

> m3<-lmer(RT~Offer+(1+Offer|Subject),data=data)
> fixParam<-fixef(m3)
> ranParam<-ranef(m3)$Subject
> params<-cbind(ranParam[1]+fixParam[1],ranParam[2]+fixParam[2])
> plot(RT~Offer,data=data,col=rgb(0,0,0,.1),pch=16,cex=4,
    ylab="Reaction Time (Seconds)",xlab="Offer Amount ($)")
> subNum<-unique(data$Subject)
> for(i in 1:length(subNum)){
    abline(a=params[i,1], b=params[i,2],col="grey",lty=2,lwd=2)
    }
> abline(fixParam,lwd=6,col="red")
```

1.2 Plotting Mixed Logit Models

It is often useful to have a plot accompanying the results of an analysis depicting the effect of interest. This can be complicated when you are interested in plotting a sigmoid function from a mixed logit analysis, such as our example from the end of the mixed model section. This example will demonstrate how to create a sigmoid plot of an interaction between the amount of money offered and the type of partner (e.g., Human or Computer). We will use the plotLMER.fnc() from the languageR package. To begin we calculate a separate regression for each level of condition, by first creating a subset of the data for each group. We then plot the sigmoid function using the plotLMER.fnc() function and overlay the plot for both models using the par(new=TRUE) command. We can create a legend indicating which condition corresponds to the level of condition using the legend command. Finally, we can create density rugs of the actual choice data by setting the values of the choices to a specific height on the y axis and then jittering them at every level of offer. To do

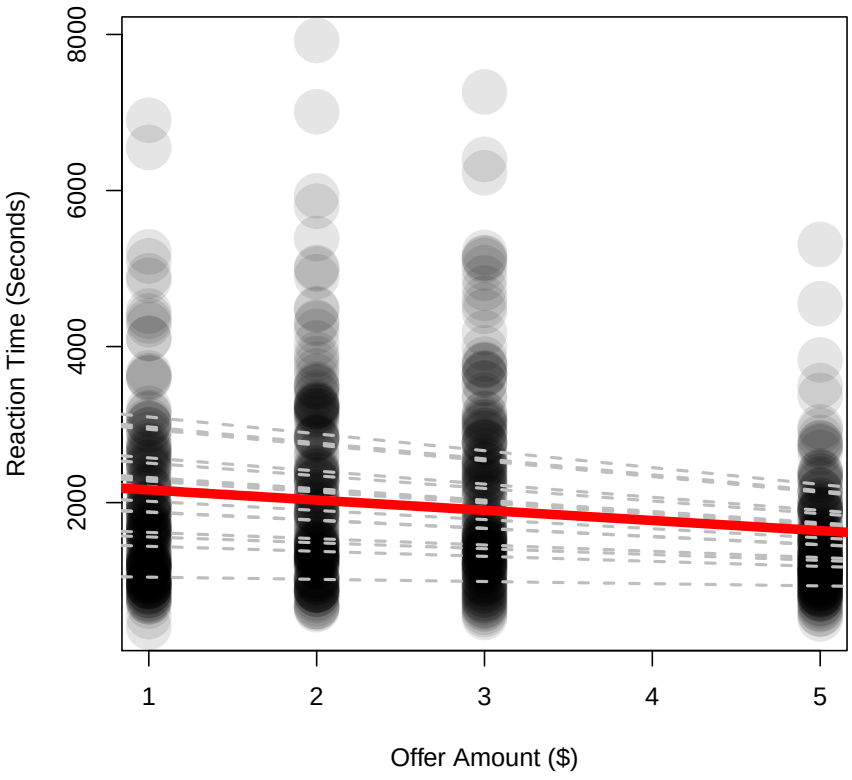

Figure 1: Increased decision conflict for decreasing offer amounts

this we search for observations which correspond to either accept (i.e., 1) or reject (i.e. 0) and replace them with the height we would like them to appear on our graph, which spans from 0 to 1. We then apply `jitter` to the x values, which we have extracted using the `model.matrix` command and assign these to `points` on the graph. Our graph illustrates that participants were more likely to accept unfair offers from computers compared to humans.

```
> library(languageR)
> h<-subset(data,data$Condition=="Human")
> c<-subset(data,data$Condition=="Computer")
> m1<-glmer(Decision~Offer+(1|Subject),data=h,
    family=binomial(link="logit"))
> m2<-glmer(Decision~Offer+(1|Subject),data=c,
    family=binomial(link="logit"))
> plotLMER.fnc(m1,ylimit=0:1,lockYlim=TRUE,linecolor="red",
    lwd=4,xlabel="Offer Amount ($)",
    ylabel="Probability of Accepting Offer")

log odds are back-transformed to probabilities

> par(new=TRUE)
> plotLMER.fnc(m2,ylimit=0:1,lockYlim=TRUE,linecolor="blue",
    lwd=4,xlabel="Offer Amount ($)",
    ylabel="Probability of Accepting Offer")

log odds are back-transformed to probabilities

> legend("bottomright", c("Human","Computer"), pch=15,
    col=c("red","blue"),title="Condition")
> x1<-h$Decision
> x2<-c$Decision
> x1<-replace(x1,which(x1==1),1.03)
> x1<-replace(x1,which(x1==0),-.03)
> x2<-replace(x2,which(x2==1),1)
> x2<-replace(x2,which(x2==0),.0)
> points(jitter(model.matrix(m1)[,2]),x1,col=rgb(1,0,0,.1),
    pch=15,cex=2)
> points(jitter(model.matrix(m2)[,2]),x2,col=rgb(0,0,1,.1),
    pch=15,cex=2)
```

```
log odds are back-transformed to probabilities
log odds are back-transformed to probabilities
```

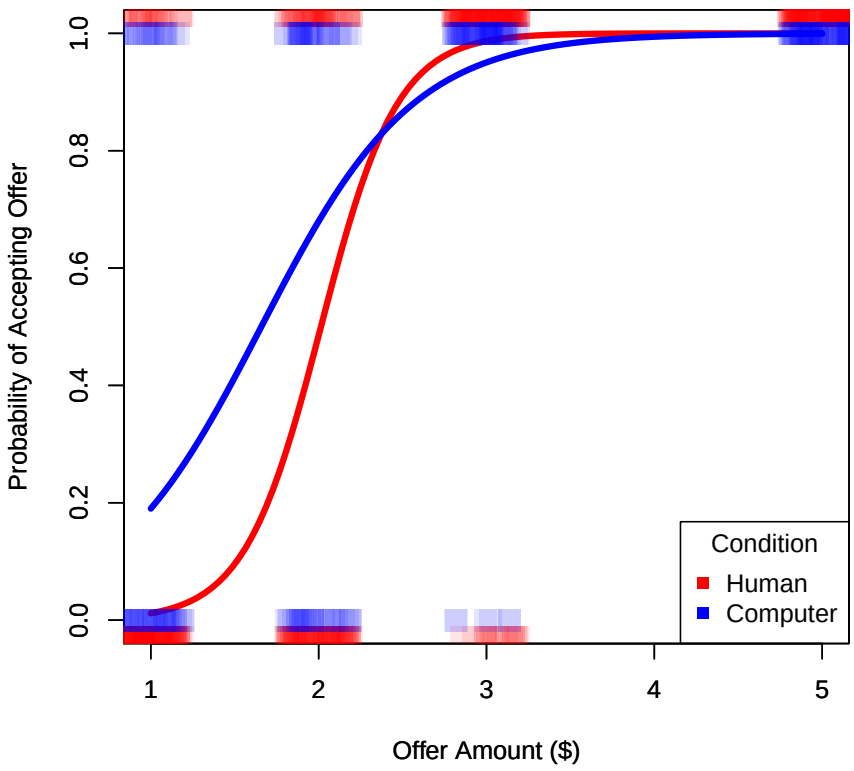

Figure 2: Output from pval.fnc()

2 gplot

Earlier we mentioned that the base `barplot` function did not have a straightforward way of adding error bars to a plot. Here we will illustrate how this can be done using the `barplot2` function from the `gplots` package. For this example we will be plotting the average acceptance rate for each level of offer amount. We first need to load the `gplots` package. Next we will create a summary of the variables we would like to plot, which will be the mean and also the upper and lower bounds of our confidence interval, which will be ± one standard deviation. We then put all of these variables as input into the `barplot2` function.

```
> library(gplots)
> mn<-tapply(data$Decision,data$Offer,mean, na.rm=TRUE)
> ciu<-tapply(data$Decision,data$Offer,sd, na.rm=TRUE)+mn
> cil<--1*tapply(data$Decision,data$Offer,sd, na.rm=TRUE)+mn

> barplot2(mn,plot.ci=TRUE,ci.u=ciu,ci.l=cil,col="gray20",ci.col="red",
    ci.lwd = 4,ci.lty=1,xlab="Offer Amount ($)",
    ylab="Average Acceptance Rate")
```

3 ggplot2

While we have mainly explored creating graphs using the basic `R` plotting functions there are a number of other graphics engines. `ggplot2` is a graphing package for `R` that uses its own grammar to allow for plots that are precisely suited to address a specific problem. The package has a very helpful accompanying websiteand book.

3.1 Barplots

To introduce the power of the package we will continue with the Ultimatum Game example and plot RT as a function of Offer Amount. First, we will load the `ggplot2` package.

```
> library(ggplot2)
```

Next we will create a data frame that summarizes our data in terms of mean and standard error at each level of offer amount. We will use the `tapply()` function to

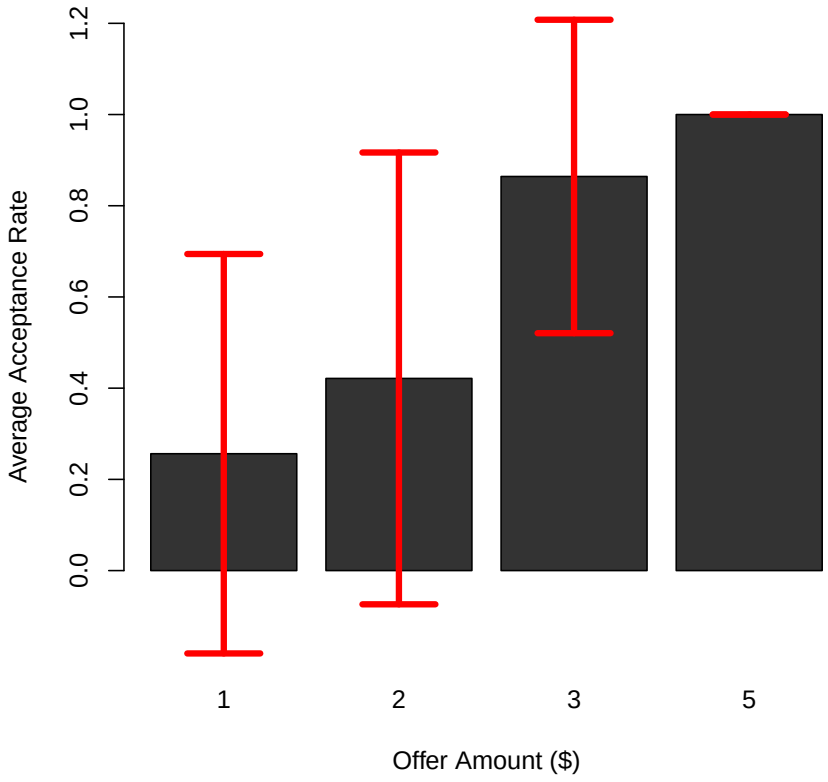

Figure 3: Barplot with error bars of average acceptance rates as a function of offer amount

calculate the mean and standard deviation for each level of offer amount. We will also use the `tapply()` function to determine the number of observations for each level of offer using the `length` command. This will allow us to transform the standard deviation into standard error by dividing the standard deviation by the `sqrt()` of the number of observations. We then add and subtract the se from the mn.

```
> data$Offer[data$Offer==5]<-4
> offer<-as.factor(c(1,2,3,4))
> se<-tapply(data$RT,data$Offer,sd)/sqrt(tapply(data$RT,data$Offer,length))
> mn<-tapply(data$RT,data$Offer,mean)
> semax<-mn+se
> semin<-mn-se
> dat<-data.frame(cbind(offer,mn,semax,semin))
```

The `ggplot2` package can create standard plot using the `qplot()` command. However, it can also build fully customizable plots by building them layer by layer with the `ggplot()` command. We will demonstrate how to build a barplot layer by layer by first creating a `ggplot()` object, which will use to build different types of graphs. To do this we specify the dataset to use and also the x and y variables to use as the defaults in the plots using the `aes()` command.

```
> ug<-ggplot(dat,aes(offer,mn))
```

We can create a simple barplot with lines as error bars by adding the bars with the `geom_bar()` command and error bars, here represented as lines, using the the `geom_linerange()` command. We can change the x and y labels by adding `xlab("?")` and `ylab("?")` respectively. `ggplot` automatically creates legends for figures, which we will suppress here by setting `legend.position` to none in the `opts` statement.

```
> print(ug+geom_bar(stat="identity")
    +geom_linerange(aes(ymin=semin, ymax=semax,
    col="red", size=2))+xlab("Offer Amount ($)")
    +ylab("Mean Response Time (Seconds)")
    + opts(legend.position = "none") )
```

We can easily change the type of plot from a barplot to a lineplot using the `geom_line()` command. We can also change the error bars from lines to whiskers using the `geom_errorbar()` command.

```
> print(ug+geom_line(aes(size=2))
    +geom_errorbar(aes(ymin=semin,ymax=semax,col="red",width=.25,size=2))
```

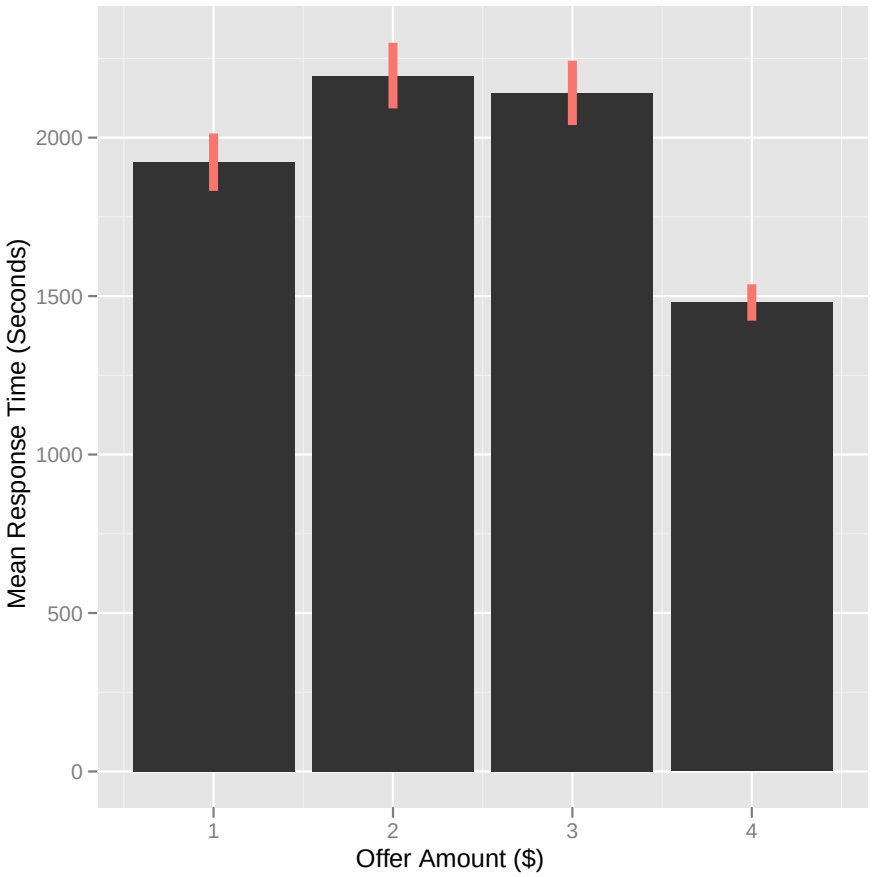

Figure 4: Barplot with error bars of average response time as a function of offer amount

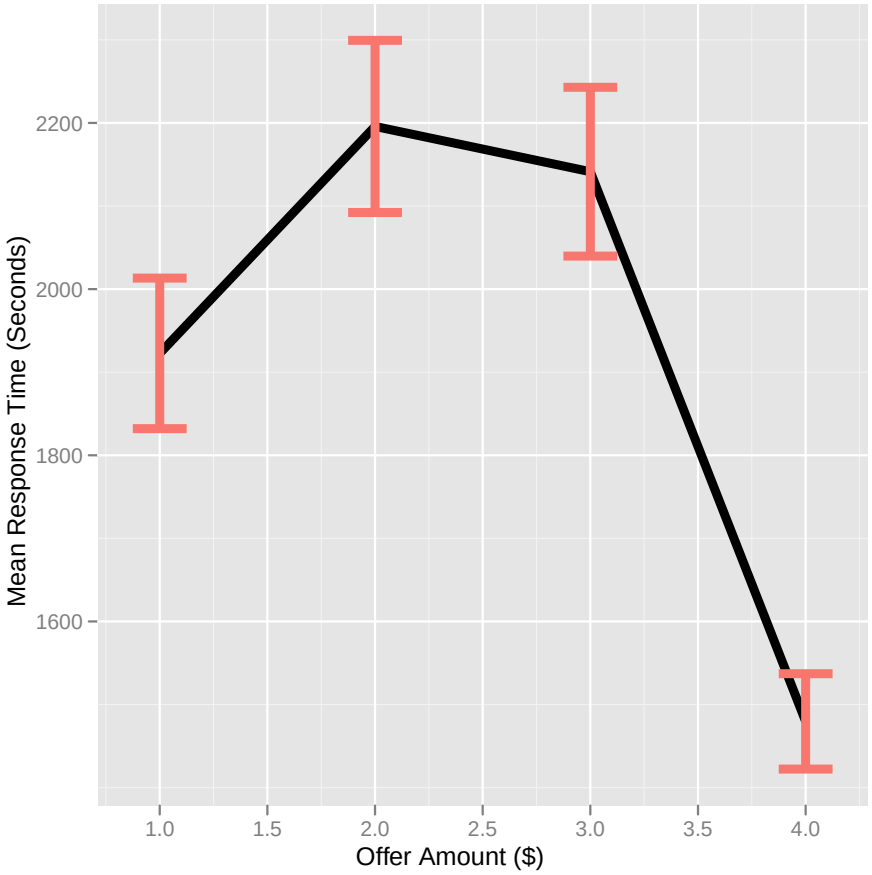

Figure 5: Line with error bars of average response time as a function of offer amount

```
    +xlab("Offer Amount ($)")+ylab("Mean Response Time (Seconds)")
    +opts(legend.position = "none"))
```

Another nice feature of building plots layer by layer in `ggplot2`, is that you can combine multiple datasets. Here we will overlay the actual RT data over the barplot by instructing the `geom_point()` command to use the original dataset rather then the summary data frame that we use to plot the bars.

```
> print(ug+geom_bar(stat="identity")+geom_point(data=data,aes(Offer,RT))
    +xlab("Offer Amount ($)")+ylab("Mean Response Time (Seconds)"))
```

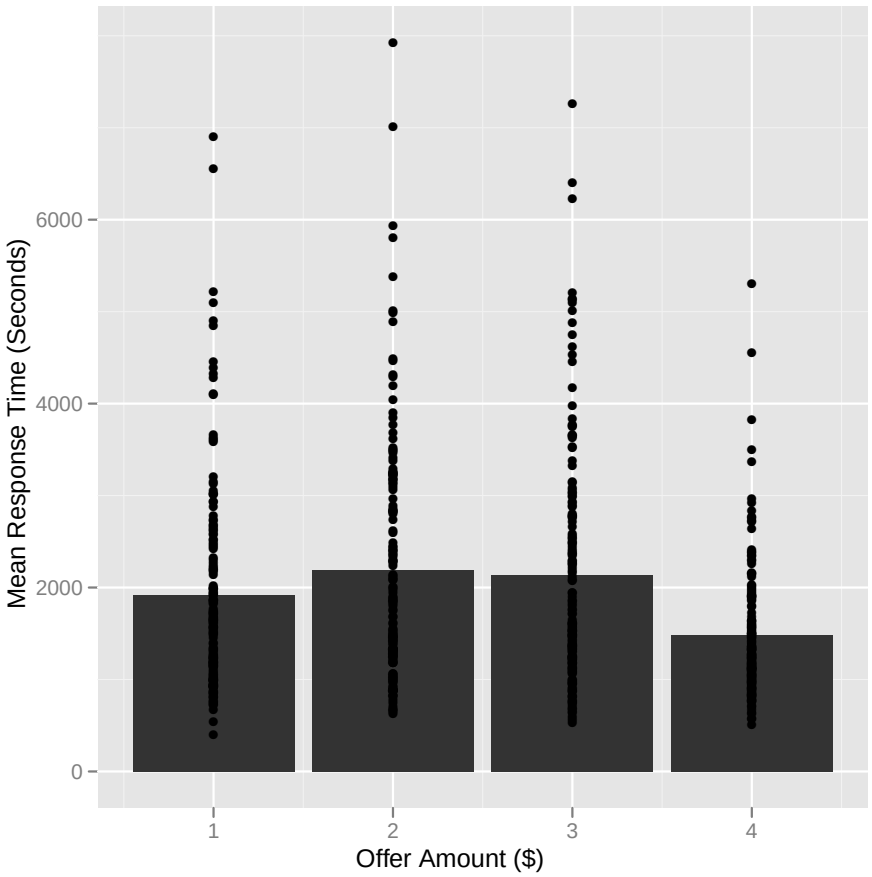

Figure 6: Barplot of average response time as a function of offer amount with distribution of data

3.2 Line Plots with Confidence Band

In neuroimaging analyses it is common to report peristimulus plots, or the average BOLD activity following the onset of a stimuli for a given condition. This is similar to an ERP in electrophysiology. This example will demonstrate how we can use `ggplot2` to create a peristimulus plot of BOLD activity in the insula for two different conditions (e.g., Equal and Less)[1]. We will also plot a ribbon of the confidence of the average, which is ± 1 standard error. Notice in the data file that we have already calculated the mean and upper and lower standard error for every time point for each condition. First we create a `ggplot` object that specifies the data and the x and y variables, which are Trial and MN respectively. Next we build the plot layer by layer by adding each graphical item sequentially to the `ggplot` object `Eq`. First we add the confidence band for the first condition `Equal` using the `geom_ribbon()` command and then the line with the `geom_line()` command. Next we do the same thing for the `Less` condition and finally we specify the colors and some additional options to customize the plot. `ggplot2` automatically creates an accompanying legend. While it seem like a lot of steps, this example shows the power of `ggplot2` in generating fully customizable publication ready plots.

```
> dat<-read.table(paste(website,"BoldPlot_Data.txt",sep="")
    ,header=TRUE)
> print(dat)
```

	Trial	MN	SEU	SEL	Condition
1	1	2.500000e-03	4.417412e-03	0.0005825875	Equal
2	2	5.000000e-03	7.619684e-03	0.0023803158	Equal
3	3	8.500000e-03	1.254428e-02	0.0044557190	Equal
4	4	1.000000e-02	1.527046e-02	0.0047295372	Equal
5	5	1.450000e-02	2.079996e-02	0.0082000363	Equal
6	6	9.000000e-03	1.490779e-02	0.0030922119	Equal
7	7	1.000000e-02	1.536266e-02	0.0046373356	Equal
8	8	8.500000e-03	1.348527e-02	0.0035147276	Equal
9	9	2.272727e-03	6.380240e-03	-0.0018347851	Equal
10	10	2.941176e-04	1.777518e-03	-0.0011892824	Equal
11	11	-5.882353e-04	5.319995e-04	-0.0017084701	Equal
12	12	-1.225490e-03	-3.184619e-04	-0.0021325185	Equal
13	13	-6.862745e-04	6.849609e-04	-0.0020575099	Equal
14	14	-2.500000e-03	-1.144185e-03	-0.0038558154	Equal

[1]This data is taken from Chang, Smith, Dufwenberg, & Sanfey (Under Review)

```
15     1  1.250000e-03  2.804338e-03 -0.0003043379      Less
16     2 -1.250000e-03  1.627273e-04 -0.0026627273      Less
17     3 -2.500000e-03  1.048223e-05 -0.0050104822      Less
18     4  1.250000e-03  4.341735e-03 -0.0018417347      Less
19     5  1.250000e-03  4.785534e-03 -0.0022855339      Less
20     6 -6.875000e-03 -3.269594e-03 -0.0104804056      Less
21     7 -4.375000e-03  7.877911e-04 -0.0095377911      Less
22     8 -8.125000e-03 -5.379683e-03 -0.0108703169      Less
23     9 -7.000000e-03 -5.466070e-03 -0.0085339300      Less
24    10  1.528945e-04  9.831155e-04 -0.0006773265      Less
25    11 -2.135854e-04  1.062466e-03 -0.0014896374      Less
26    12 -1.796919e-03 -8.790570e-04 -0.0027147805      Less
27    13 -4.035948e-04  1.304397e-03 -0.0021115862      Less
28    14 -7.329599e-05  1.227175e-03 -0.0013737672      Less
```

```
> Eq<-ggplot(dat,aes(Trial,MN))
> print(Eq+geom_ribbon(data=subset(dat,dat$Condition=="Equal"),
    aes(ymin=SEL,ymax=SEU),fill="light yellow")
    +geom_line(data=subset(dat,dat$Condition=="Equal"),
    size=2,aes(colour="Equal"))
    +geom_ribbon(data=subset(dat,dat$Condition=="Less"),
    aes(Trial,MN,ymin=SEL,ymax=SEU),fill="light blue")
    +geom_line(data=subset(dat,dat$Condition=="Less"),
    aes(Trial,MN,colour="Less"),size=2)
    +scale_colour_manual("Condition",
    c("Equal"="dark orange","Less"="dark blue"))
    +opts(title = "Insula Activity", aspect.ratio = 1,
    panel.border= theme_rect(colour = "black",size=2))
    +ylab("Percent Signal Change")
    +xlab("Volume"))
```

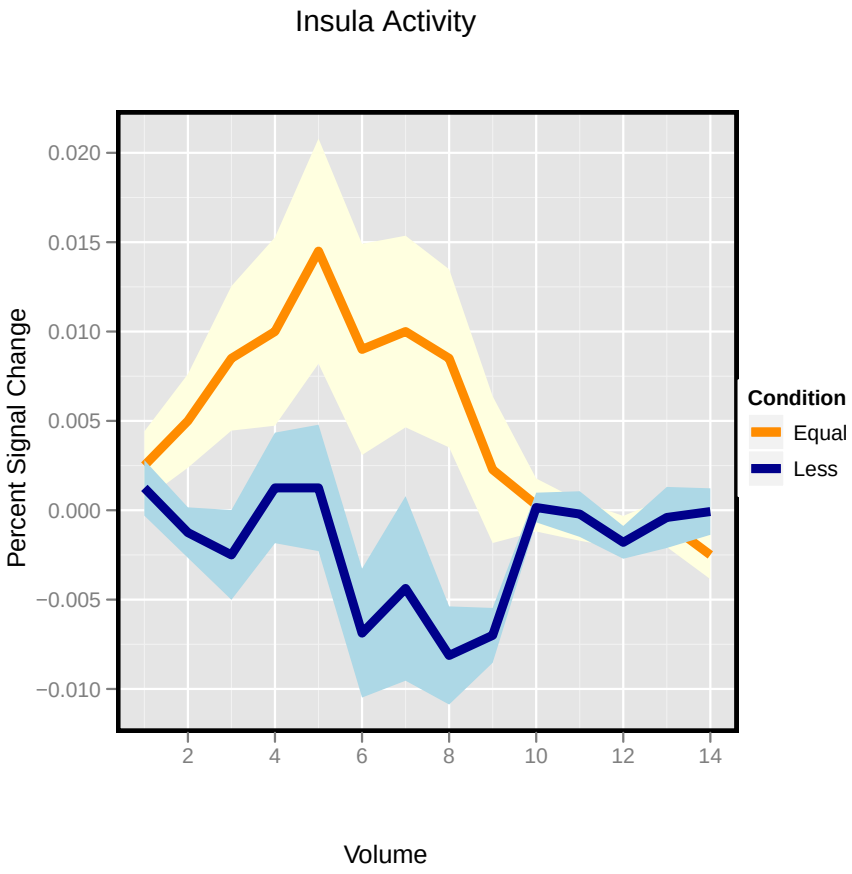

Figure 7: Average Insula Activity