Lab 2: The Stroop Effect: Interference between cognitive processes

Goal

Certain cognitive processes seem *automatic* in the sense that the do not require effortful thought or seem beyond our conscious control. In this lab, our goal is to investigate this intuition experimentally by replicating the basic (and infamous) "Stroop Effect". Along the way we hopefully will learn something about automatic cognitive processes, interference, the use of reaction time data in analyzing human cognitive processes. In addition, we will take the first steps of expanding the tools we use to perform statistical analysis out beyond Excel to include the "R" program.

Background

The Stroop effect was first described in 1935 by J. Ridley Stroop. In his experiment, Stroop had people read aloud color words (i.e., "red", "blue", "green", etc...) that were presented in different colored fonts. The main finding was that when people had to say the name of the font color, they were faster to respond when the color of the word matched the color of the font, and slower when these mismatched. The slowed responding is considered by many to be evidence of *interference* between cognitive processes. Namely, reading is assumed to be an "automatic" process that we engage without even trying. Thus, when you see a word like this:

RED

and are asked to say the color of the font, the automatic "reading" process first steps in and determines the semantic meaning of the word (red). Then, you have to override this impression with a secondary process which is naming the color of the font (blue). Generally, this leads to more mistakes and delayed responding. Alternatively, it could be viewed as a kind of "facilitation" for responding to word-font combinations that match.

The Stroop Effect has been replicated millions of times, and is a foundational effect in psychology. Beyond a simple demonstration of interference/facilitation for particular cognitive tasks, the Stroop Effect has been used in the fMRI scanner to study the brain systems involved in executive control. In addition, it suggests that in our culture, reading words becomes a skill we use so often that it is "automatized" (i.e., is engaged automatically as soon as we see something we interpret as a word).

Present Experiment

In this lab, we will try to do the following:

1. Replicate the basic stroop effect
2. Determine if the effect can be made smaller by interfering with the reading of words (by turning them upside down)

3. Write a lab report about it.

Procedure
I have written a simple program in python that lets us measure reaction time and display words in different colored fonts on the computer screen (and can flip them upside down). Each person will have to run themselves in the experiment.

There will be six different types of trials in which you respond to font color. Words will either be color words ("red", "blue", "green", etc...), or non-color words (which you can make up). Combining this gives us:

Respond to Color

1. Neutral words (i.e., non color word - Normal)
2. Congruent (color and font match) - Normal
3. Conflicting (color of word, and actual word do not match) - Normal
4. Neutral words (i.e., non color words - Inverted)
5. Congruent (color and font match) - Inverted
6. Conflicting (color of word, and actual word do not match) - Inverted

Presented as a Table:

	Control Words	Congurent words	Conflicting
Normal	LAPTOP	BLUE	GREEN
Inverted	LAPTOP	BLUE	GREEN

Respond to Word

7. Congruent (color and font match) - Normal
8. Conflicting (color of word, and actual word do not match) - Normal
9. Congruent (color and font match) - Inverted
10. Conflicting (color of word, and actual word do not match) - Inverted

Presented as a Table:

	Congurent words	Conflicting
Normal	BLUE	GREEN
Inverted		

We'll be having people make a large number of such judgments and record their accuracy and reaction time on each trial.

What we will be primarily looking for are differences in reaction time for congruent vs. incongruent stimuli in the font-color naming condition (this is the standard stroop effect). However, it will also be useful to compare responding to color for non-color words to show that the cost is only for words that share the right semantic properties.

Second, we'll be interested in if you get interference when responding to the word (ignoring color). The idea is that it should be easier to ignore color than to ignore the word, these we shouldn't find interference here.

Finally, we will be interested in the standard Stroop effect goes away when the words are upside down (interfering with reading them). Again, this may help establish that the effect is tied to the automaticity of reading.

Data file format:

The data from our class is summarized in a file called stroop.txt.

This file has 11 columns:

1. subject #
2. condition #
3. trial #
4. presented word
5. respond to (1=word, 2=font color)
6. trial type (1=control, 2=incongruent, 3=congruent)
7. rotated? False (i.e, upright)=0, True (i.e., upside down)=1
8. correct response (y=yellow, r=red, b=blue, g=green)
9. actual response (what the subject responded, same as #8)
10. hit (i.e. were the correct?) 1=yes, 0=no
11. reaction time (in milliseconds)

Analysis

Goal 1: Setup and some R basics

1. Step 1 - open R and a text editor (text edit will work)

You will want to save your R commands as you write them to the text file so you can always re-run your analysis later.

2. Step 2 - Read in your data

To read in our data file, you first need to tell R where it is. Do to this choose "Choose dir." from the FILE menu in R. Select the directory containing our class data file (stroop.txt).

The command to read in our data is:

```
mydata = read.table("stroop.txt", header=F)
```

This reads our data into a variable in R called "mydata". The read.table() function takes two parameters: the filename we want to read in ("stroop.txt") and a command that tells it if it should treat the first line of the file as a "header" column. Since we don't have column headers in our file, we want to say "False" (so header=F).

However, it will be easier if we name our columns. Luckily we can do this after the fact in R using the names() command. According to the description of the data file above we can name our columns with the following command.

```
names(mydata) = c('subj', 'cond', 'trial', 'word',
                  'respto', 'trialtype', 'rotate',
                  'cans',   'resp', 'hit', 'rt')
```

3. Step 3 - Drop outliers

Since we are going to analyze RT quite a bit in this experiment, we will want to be sure to exclude outliers. These are trials where the participant (i.e., you) stared blankly at the screen, decided to itch your foot, stretch, raise your hand to ask a question, etc... Since these reaction times will be really long, we will want to remove them from our analysis of the reaction time data.

In excel this type of thing is difficult or even tedious to do.

In R, we can refer to particular columns of our data file (using the names we set above) like this:

```
mydata$rt
```

If you type the above and press enter, R will print out the RT column.

We can also due True/False tests on the data like this.

```
mydata$rt < 2500
```

This re-scores each entry in the data file as "True" if it is less then 2500 ms, and "False" otherwise.

Even cooler, we can then use this true/false test to remove data from our file

```
mydata[mydata$rt<2500, ]
```

If you save this as a new copy of the data with the outlier RT's removed:

```
cleandata = mydata[mydata$rt<2500, ]
```

4. Step 4 - Define condition codes (for easier to read code)

Next, we want to do a little house keeping and define some of the codes we used in our data file. Enter the following variables into R

```
WORD = 1
COLOR = 2

NORMAL = 0
ROTATED = 1

CTL = 1
CON = 2
INCON = 3
```

5. Step 5 - Analyze accuracy (i.e., hit rate)

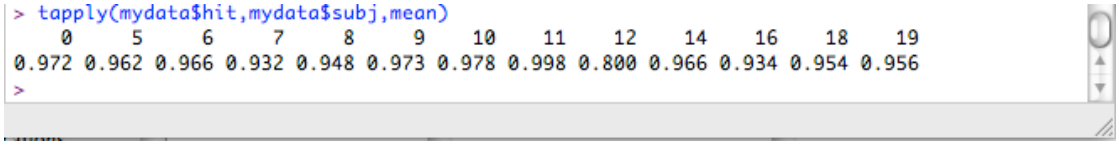
Finally, lets compute the accuracies for each subject in class. Note we want to do this on the original data (not the data with the RT outliers removed). Thus we will be dealing with our original `mydata` variable.

Now, imagine how you would do this in Excel. Want we want to do is calculate the mean accuracy for each subject in our experiment. We'd spend a lot of time making things call

pivot tables and stuff like this. Let's do it the R way. We tell R we want to find mean hit-rate (i.e., accuracy) as a function of subject number. There is a really powerful function in R called `tapply()` which applies whatever function we want (in this case `mean()`) to any combination of columns. To find accuracy as a function of subject we type

```
accs = tapply(mydata$hit, mydata$subj, mean)
```

Now, look "inside" variable `accs` by typing "`accs`" on a line by itself. What do you



see? If you did things right, you should see something like this:

The numbers along the top are the subject numbers, and below are the accuracies for each person. Who is subject 12? We may never know, but they were only 80% correct!

To find overall mean accuracy or other descriptive statistics we can just find `mean(accs)`, `sd(accs)`, `median(accs)`, `max(accs)`, `min(accs)`.

GOAL 2 - Analyze the basic stroop effect in our UPRIGHT, COLOR RESPONDING conditions. In other words, did we find the basic stroop effect?

OK, with those preliminaries out of the way, we can turn to our original question. Did we replicate the Stroop effect or not? Of course, we had a number of ancillary hypotheses we were interested in as well. To begin with let's just consider people in the RESPOND TO COLOR condition where the stimuli were UPRIGHT. This is the basic stroop condition.

To do this we need to throw away all the other data and just concentrate on trials that fit this description. Given what we did in Step 3 can you think of a way to keep only these trials?

Hint:
To select all trials where the response to was to color we can do the following (note we are using the `cleandata` variable because we want to throw away the outliers in our RT analysis):

```
cleandata[cleandata$respto==COLOR, ]
```

To get the trials where the stimulus was upright we do

```
cleandata[cleandata$rotate==NORMAL, ]
```

How do we get trials where respond to was COLOR and the words were upright?

```
cleandata[cleandata$respto==COLOR &
          cleandata$rotate==NORMAL, ]
```

Poof! The & will do a logical "AND" on the two conditions and let us select out only the trials we want. We should save this as

```
coloruprightdata = cleandata[cleandata$respto==COLOR &
                cleandata$rotate==NORMAL, ]
```

6. Let's plot a histogram of our RT data in each condition to see the total RT distribution

Lets look at the distribution of RTs in the CONTROL, INCONGRUENT and CONGRUENT conditions (again limiting to only the UPRIGHT, COLOR trials).

To get all the RTs from the control trials we simply perform another sub-selection of our data:

```
coloruprightdata[coloruprightdata
     $trialtype==CTL, ]$rt
```

Note the thing on the end mean we'll select only the "rt" column after looking only at control (i.e., non color-word trials).

To plot a histrogram just use the hist() function

```
hist(coloruprightdata[coloruprightdata
     $trialtype==CTL,]$rt,
     breaks=seq(0,2500,50), col="red1", ylim=c(0,200))
```

A couple notes, the first parameter is our data. breaks is the number of "bins" in our histogram. c(0,2500,50) makes a list of bins each 50ms wide all the way up to 2.5 seconds (2500 ms). col tells us the color of our bars, and ylim is the range of the yaxis (here going from 0 to 200). You can play with some of these parameter to change your plot.

Not make three of these plots, one for the CONTROL trials, one for INCONGRUENT trials, and one for CONGRUENT trials. You just have to change the above command for each one to select out a different set of RT data.

Note, there is another argument for hist() which is add=T. This means each new plot is

laid on top of the next. This is useful if you want to be able to better visually compare the distributions. i.e.,

```
hist(data1, breaks=seq(0,2500,50), col="red1",
     ylim=c(0,200))
hist(data2, breaks=seq(0,2500,50), col="green1",
     ylim=c(0,200), add=T)
hist(data3, breaks=seq(0,2500,50), col="blue1",
     ylim=c(0,200), add=T)
```

lays them on top of one another (with data3 in the front). Note you don't *literally* want to type `data1`, `data2`, `data3` here, but copy the syntax from the line above to select out the different types of trials.

Task: Make individual plots for each condition, and then a combined plot. What do you see? Is one distribution shifted compared to the other?

Note, you can save the plots using the File->Save as.. command in from the menu system in R (options include PDF, BMP, JPG, etc... all are compatible with Word).

A final note, hist() takes a couple other arguments which allow to set the range of the y-axis, x-axis, put a label on both axes, and a title for the plot.

`ylim= c(0,100)` <- this makes the range of the y-axis be from 0 to 100
`xlim = c(0,2500)` <- this makes the range of the x-axis be from 0 to 2500 (not necessary in hist() since your breaks=seq(0,2500,50) determines the range

`ylab = "Frequency"` <- this is the label for the y-axis
`xlab = "RT (ms)"` <- this is the label for the x-axis

`main = "My Title"` <- set a title for the plot.

In APA format, plots don't usually have titles so use the following to remove the title altogether

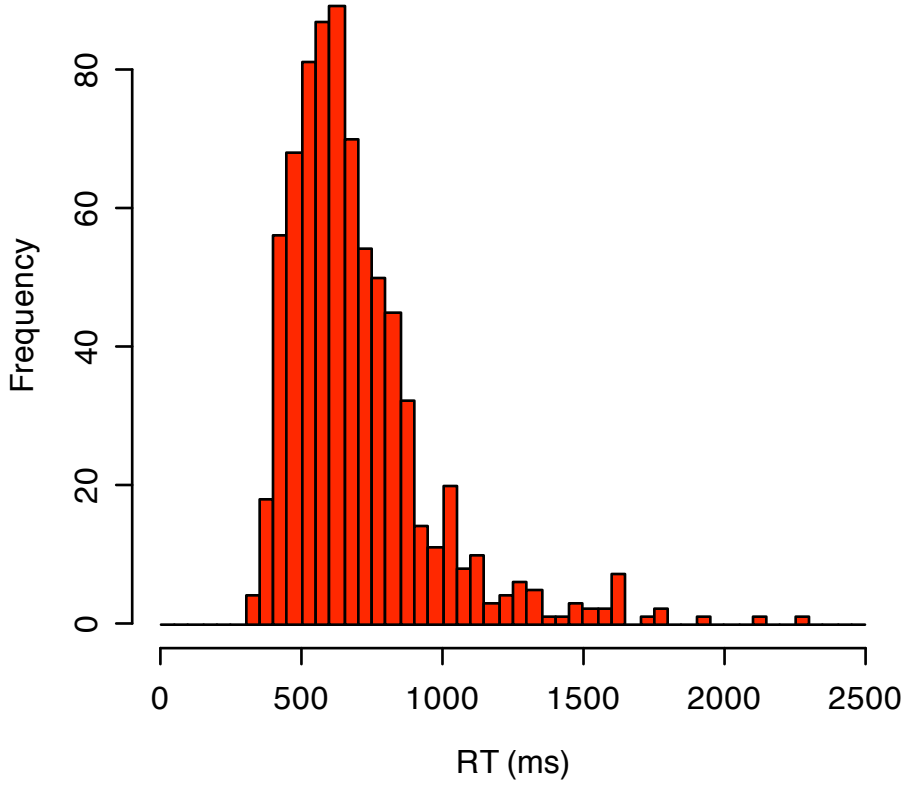
```
main = ""
```

Putting it all together, let's plot the RT distribution from the control condition:

```
data1= coloruprightdata[coloruprightdata$trialtype==CTL,]$rt

hist(data1, breaks=seq(0,2500,50), col="red1",
     ylim=c(0,200), xlab="RT (ms)", main="")
```

You should get something like this:



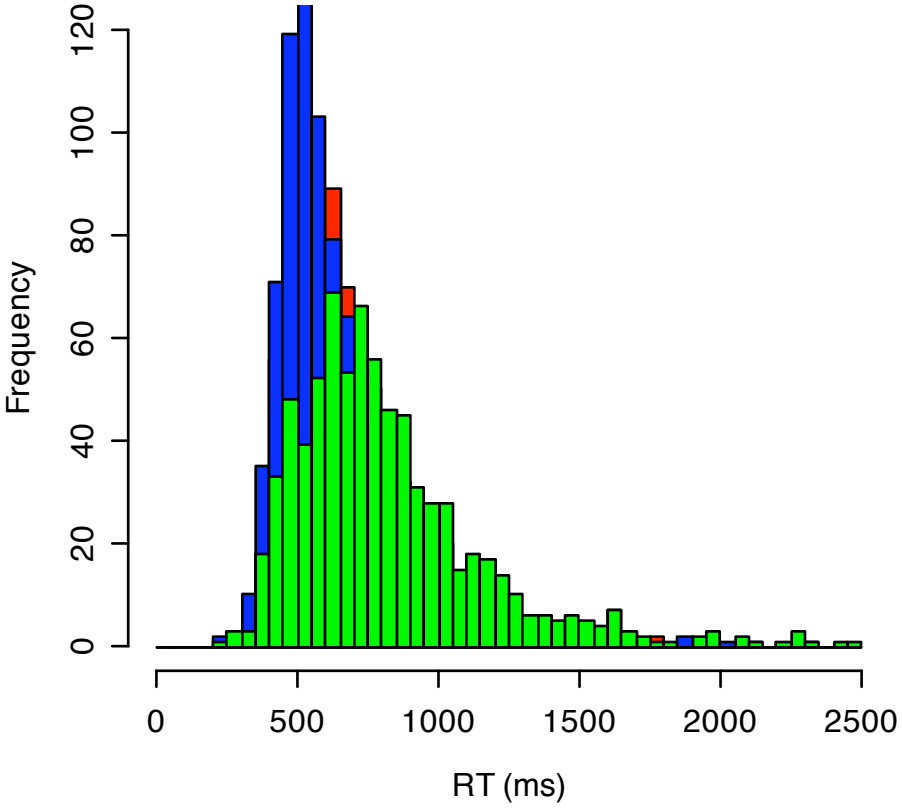
Then we can add the data for the congruent and incongruent conditions:

```
data1= coloruprightdata[coloruprightdata$trialtype==CTL,]$rt
data2= coloruprightdata[coloruprightdata$trialtype==CON,]$rt
data3= coloruprightdata[coloruprightdata$trialtype==INCON,]$rt

hist(data1, breaks=seq(0,2500,50), col="red1",
     ylim=c(0,150), xlab="RT (ms)", main="")
hist(data2, breaks=seq(0,2500,50), col="blue1",
     ylim=c(0,150), xlab="RT (ms)", main="", add=T)
hist(data3, breaks=seq(0,2500,50), col="green1",
     ylim=c(0,150), xlab="RT (ms)", main="", add=T)
```

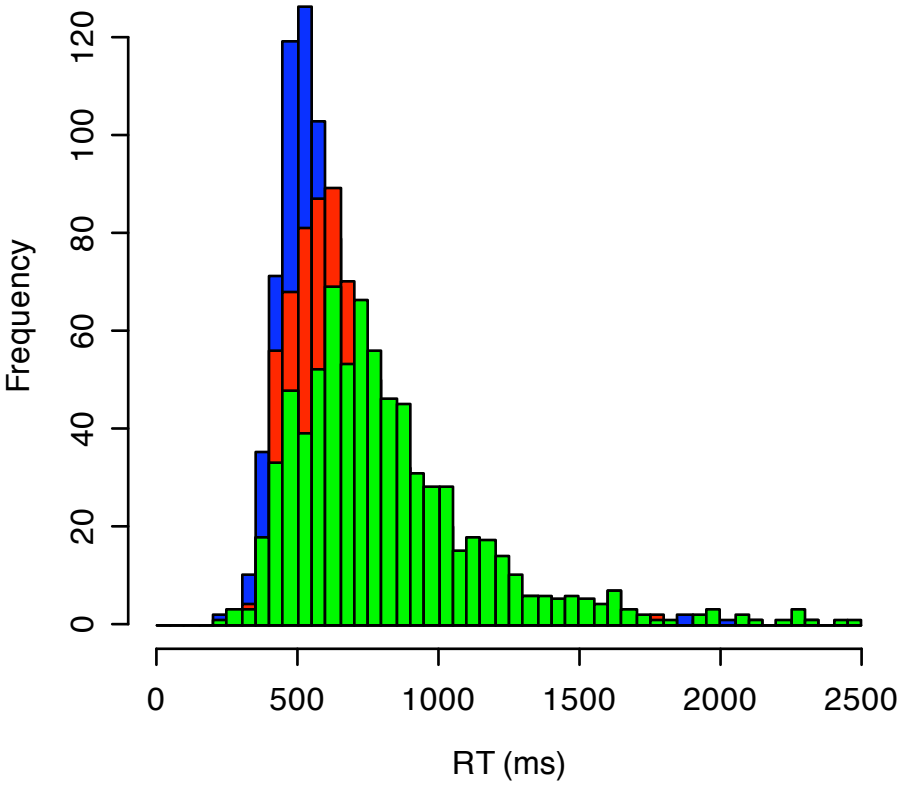
Note how I highlighted in green the addition "add=T" argument to the second and third histogram. This means they will be pasted on top of the first one.

The final result:



However, the control distribution (red) is kind of hidden. If you change the order of the command we can put the blue one first (be sure to change the colors too!):

```
hist(data2, breaks=seq(0,2500,50), col="blue1",
     ylim=c(0,150), xlab="RT (ms)", main="")
hist(data1, breaks=seq(0,2500,50), col="red1",
     ylim=c(0,150), xlab="RT (ms)", main="", add=T)
hist(data3, breaks=seq(0,2500,50), col="green1",
     ylim=c(0,150), xlab="RT (ms)", main="", add=T)
```



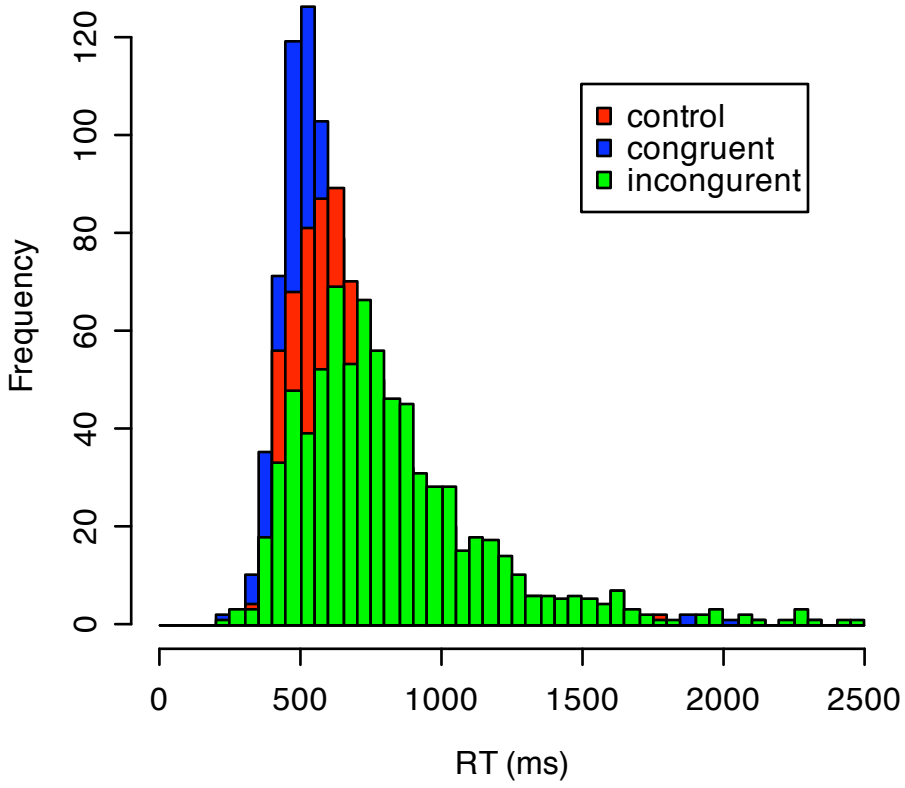
Looks pretty good!

Someone asked in class about making a legend or color key. It's actually pretty easy, just type

```
legend(1500,100,c("control","congruent","incongruent"),
     fill=c("red1","blue1","green1"))
```

The first two arguments are the position of the top left corner of the legend display. There is some free space in our plot around a Frequency of 100 and an RT of 1500, so we put those two numbers in. Next we put a list of our conditions, finally `fill =` gives the list of colors. Make sure they are in the same order as the first list (i.e., `red1` comes first since the red is the control condition).

Our final plot:



7. Summary statistics: let's mean/median compute RT as a function of condition

As a final step for today lets compare the mean and median RT for each condition. To do this, we revisit our old tapply() friend. In this case we want RT as a function of both subject and condition (CTL, CON, INCON)

Fortunately, it is really simple, we simply specify TWO columns to tapply using the list() command (compare to the first encounter with tapply above).

```
rtbycond = tapply(coloruprightdata$rt,
     list(coloruprightdata$subj, coloruprightdata
          $trialtype), mean)
```

or

```
rtbycond = tapply(coloruprightdata$rt,
     list(coloruprightdata$subj, coloruprightdata
          $trialtype), median)
```

Look inside rtbycond by typing `rtbycond` and hitting enter. What do you get?

As a final step, lets convert this result into a data "frame" so we can name the columns (this is just an R-oddity, every program has one).

```
rtbycond = as.data.frame(rtbycond)
names(rtbycond) = c("CTL", "INC", "CON")
```

Look at rtbycond now. Is should show mean (or median) RT for each subject in each of our three conditions. To find the mean of the columns of this we use apply()

```
ms = apply(rtbycond, 2, mean)
```

The apply() function simply finds the averages of either the rows or the columns of our matrix. The first variable is the data, the second is which way you want to average the data (2= in columns, 1 = in rows), and third is the function you want to perform. (this can be sd, median, etc...)

For example, try this:

```
apply(rtbycond, 2, mean)
```

Then this:

```
apply(rtbycond, 1, mean)
```

See how one case we get 3 numbers (the means of the columns) and the other case we get 13 or so (means of the rows).

Using this we can compute the standard errors:

```
nsubj = length(rtbycond[,1])
sds = apply(rtbycond, 2, sd)
se = sds/sqrt(nsubj)
```

The first line computes the number of subject we had by finding the length of the first ROW of our matrix. Then in the second line we computed the standard deviations of each COLUMN (i.e., the standard deviation in each condition). Finally we divide the standard deviations by the square root of the number of subjects (this one line divides all three standard deviations at once.

The final step let's make a bar-chart of our data. R provide a function called barplot() which creates the barplot

```
x=barplot(ms, col=c("red1","blue1","green1"),main="RT
     in each condition",xlab="Condition",ylab="RT
     (ms)",ylim = c(0,1000))
```

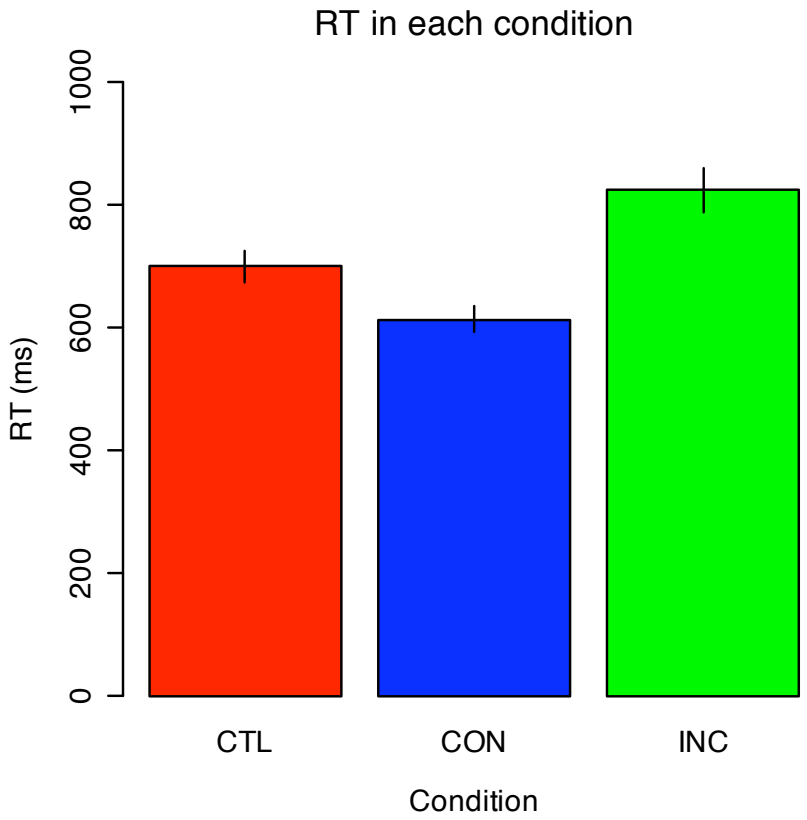
Some of this should look familiar, (for example xlab, ylab, ylim, etc...) These are mostly common things you use to tell R to format plots.

The first variable to barplot is our list of means (ms). Next we have the colors of our three bars. Next a plot title, a x-axis label (xlab), etc...

Finally, we want to plot the error bars. To do this we use the segments command. notice above that when we created the barplot we set it equal to x. As a result x has the location of the bars we drew. To plot the error bars, we just use the following command:

```
segments(x, ms-se, x, ms+se)
```

The result is:



Niiiiice! People were fastest in the congruent condition (where the word and

color matched), slowest in the incongruent condition (where they mismatched), and in between in the control condition (where the words were non-color associated or non-sense). In other words, we replicated the stroop effect.

8. Nice pictures, but are the results significant?

The title describes it all. The first step is to see if there is a significant effect across all three conditions. The way to provide an answer to this is to do a one-way, repeated measures ANOVA on RT as a function of condition.

Here's how to do it:

First we need to reform our `rtbycond` matrix we computed above. Remember this shows mean RT as a function of condition and subject number. It looks roughly like this

	control	congruent	incongruent
subj 1	456.87	238.48	298.12
subj 2	3894.00	2398.00	28934.9
subj 3	898	878	878
...			

The format for the ANOVA is a reworking like this:

subj1	condition	RT
1	1	456.87
1	2	238.48
1	3	298.12
2	1	3894.00
2	2	2398.00
2	3	28934.9
3	1	898
3	2	878
3	3	878
.....		

So basically we create a new column that kind of "flattens" our `rtbycond` matrix so that RT again has its own column, but now subject and condition codes are provided.

Here's a trick to do the conversion:

```
found = which(rtbycond!=-999,arr.ind=T)
rtANOVA = data.frame(cbind(found,rtbycond[found]))
names(rtANOVA) = c('subj','cond','rt')
```

Afterwards rtANOVA should contains something like on the left side of the page below:

```
   subj cond       rt
1     1    1 785.6279
2     2    1 647.9091
3     3    1 847.2500
4     4    1 659.3889
5     5    1 646.6889
6     6    1 688.7355
7     7    1 821.3182
8     8    1 634.4483
9     9    1 560.6600
10   10    1 633.1228
11   11    1 828.3774
12   12    1 720.3659
13   13    1 616.1273
14    1    2 678.2321
15    2    2 583.0536
16    3    2 731.7292
17    4    2 588.1346
18    5    2 548.5424
19    6    2 641.1500
20    7    2 715.3148
21    8    2 553.9362
22    9    2 497.8113
23   10    2 576.5000
24   11    2 648.2857
25   12    2 694.4878
26   13    2 524.7917
27    1    3 824.2407
28    2    3 808.0909
29    3    3 1149.5122
30    4    3 737.0891
31    5    3 760.0962
32    6    3 867.3423
33    7    3 878.5000
34    8    3 762.2453
35    9    3 607.4615
36   10    3 747.9524
37   11    3 933.5417
38   12    3 889.6182
39   13    3 741.2619
>
```

Next, we need to tell the ANOVA function in R to treat the subject and condition columns as "factors" of the ANOVA (rather than observations, which our 'rt' column is). To do this:

```
rtANOVA$subj = factor(rtANOVA$subj)
rtANOVA$cond = factor(rtANOVA$cond)
```

Ok, now we've reorganized our data in a way that the ANOVA function (aov) in R can handle. Now we're set to do our ANOVA:

```
myaov = aov(rtANOVA$rt~rtANOVA$cond+Error(rtANOVA$subj))
summary(myaov)
```

The first line runs an one way ANOVA on the RT data as a function of condition (CONTROL, CONGRUENT, INCONGRUENT) treating the SUBJ numbers as the error term (meaning that condition is treated as a repeated measure).

The last line prints out our ANOVA table which should look something like this:

```
> summary(myaov)

Error: byslot$subj
          Df Sum Sq Mean Sq F value Pr(>F)
Residuals 12 325317   27110

Error: Within
            Df Sum Sq Mean Sq F value    Pr(>F)
byslot$cond  2 288917  144459   70.45 9.033e-11 ***
Residuals   24  49212    2050
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This shows that condition had a highly significant effect on RT. We'd report this in a paper as "A one way ANOVA on RT treating trial type as a repeated measure found a highly significant effect of stimulus condition, $F(2,24)=70.45$, $p<.001$."

Finally, we can do pairwise t-tests to get a sense of exactly how the conditions differed. In particular, we want to select the RT data for the CTL condition and compare it to the CON condition. In our `rtbycond` structure we can pull out these columns directly:

look at `rtbycond:`

```
rtbycond
```

then,

```
rtbycond$CTL
rtbycond$CON
rtbycond$INCON
```

Let's t-test them. Remember this is a PAIRED t-test since the observations are repeated on each person (we have here the average RT for each person to each type of trial).

To compare the control condition with the congruent condition:

```
t.test(rtbycond$CTL,rtbycond
$CON,paired=T,mu=0,alternative="two.sided",var.equal=T)
```

The control condition with the INCongruent condition:

```
t.test(rtbycond$CTL,rtbycond$
INC,paired=T,mu=0,alternative="two.sided",var.equal=T)
```

And finally the CONgurent and INCongruent:

```
t.test(rtbycond$CON,rtbycond
$INC,paired=T,mu=0,alternative="two.sided",var.equal=T)
```

Here's what I get, respectively:

```
> t.test(rtbycond$CTL,rtbycond
+   $CON,paired=T,mu=0,alternative="two.sided",var.equal=T)

    Paired t-test

data:  rtbycond$CTL and rtbycond$CON
t = 7.9493, df = 12, p-value = 4.013e-06
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
  61.87277 108.59657
sample estimates:
mean of the differences
               85.23467

> t.test(rtbycond$CTL,rtbycond$ INC,paired=T,mu=0,alternative="two.sided",var.equal=T)

    Paired t-test

data:  rtbycond$CTL and rtbycond$INC
t = -6.4207, df = 12, p-value = 3.3e-05
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -166.58638  -82.17245
sample estimates:
mean of the differences
              -124.3794

> t.test(rtbycond$CON,rtbycond  $INC,paired=T,mu=0,alternative="two.sided",var.equal=T)

    Paired t-test

data:  rtbycond$CON and rtbycond$INC
t = -9.8144, df = 12, p-value = 4.383e-07
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -256.1489 -163.0792
sample estimates:
mean of the differences
              -209.6141

>
```

Wow! It appears that all three contrasts are HIGHLY significant ($p<.001$).
This means we can be pretty sure that all three of those bars really are different, and thus our replication of the Stroop effect worked.