610 Handout – R1

Colleen F. Moore & Mike Amato

Written for Psychology 610, University of Wisconsin—Madison.

Introduction to R; entering and manipulating data; 1-way between groups ANOVA

This handout covers:

- Preparing your data and getting it into R
- Obtaining descriptive statistics relevant to analyses of variance
- Performing a one-way between groups ANOVA with the aov command

The first thing you need to do is download and install R from http://cran.r-project.org/ . Windows users should select the "base package".

- A good intro to R is: http://www.matfys.kvl.dk/stat/kurser/basic/notes/r-note1.pdf
- Another good reference on data input and anovas is: http://www.personality-project.org/r

IMPORTANT NOTE: In this tutorial, commands that you should enter are in red, and the output you should see on your screen is in blue. Don't enter the > symbol that begins each command; it is automatically added by R. Anything following a # symbol is a comment.

1. Prepare your data

We'll use the same example data from class handout #3, shown at right. We have 25 participants, randomly assigned to one of 5 levels of a single factor, with one measurement per participant. Each cell contains a measurement from 1 participant.

Factor A				
level 1	level 2	level 3	level 4	level5
8	4	5	3	6
6	5	3	4	7
7	5	3	2	5
7	6	6	2	4
6	3	2	3	6

1. Enter the data in an Excel spreadsheet using 2 columns. Put labels at the top, calling the first column "group" and the second column "score". When you're done you should have 26 rows in your spreadsheet: one for the labels at the top, and one for each participant showing what group they were in and their score.

2. Save your data in plain text format so that R will be able to read it (because R can't read Excel format). Instead of clicking "Save", click "Save As...". At the bottom of the save window that pops up, next to where it says "save as type", you can choose the format to save in. Save as "Plain Text", which will have the extension ".txt". Name your file something useful, like *r1data.txt*

3. You're now ready to load the data into R, so open it up. The first thing you need to do is change the working directory of R to wherever on your computer you saved your data file.

windows users:	click on "File", then "Change Dir", then browse to the correct folder and click "Ok"
mac users:	Under 'Misc' click "change working directory" and navigate to your location, click "ok"

4. Once R is pointed at the correct folder we're ready to get to work. Load your data by typing the following command then hitting enter. Note that R is case senstive.

```
> exdata = read.table("r1data.txt", header=T)
```

>	lines starting with this symbol were commands entered by the user
exdata	this is a variable name we just made up. We could have named it anything.
=	tells R that we want to calculate and store a new value for our variable *exdata*. The new value is calculated from the stuff to the right of the equal sign.
read.table( )	A function used to read tables of data into R.
"r1data.txt"	An argument of the function specifying the file name. If it weren't in quotes, R would think it was a variable instead of the actual file name.
,	separates arguments of the function
header=T	tells R that the first line of the file is column names (rather than data) by setting

	the optional header argument to TRUE

Alternatively, you can use the following command which will open a window for you to choose the file.

```
> exdata = read.table( file.choose(), header=T)
```

Check to make sure everything worked properly by typing the variable name. R will display the value of that variable, which in this case is our table of data.

```
> exdata
   group score
1     1     8
2     1     6
3     1     7
4     1     7
5     1     6
6     2     4
7     2     5
...   ...   ...
22     5     7
23     5     5
24     5     4
25     5     6
```

Notice the extra column of sequential numbers on the left. R automatically adds a column for row number when you import a table.

Commands are in RED
Output is in BLUE

In the next steps we're going to want to refer to specific columns in our data, for example just the column with participants' scores. We can do this easily by first *attaching* our data frame ("exdata" is a *data frame*).

```
> attach(exdata)
> score
 [1] 8 6 7 7 6 4 5 5 6 3 5 3 3 6 2 3 4 2 2 3 6 7 5 4 6
```

Attaching is one way to go, but it's possible to refer to particular rows or columns in a data frame even if they aren't attached.

```
   > exdata$score      # dataframe$columnname
   > exdata[ ,2]       # second column of matrix exdata
   > exdata[3, ]       # third row in exdata
```

2. Get descriptive statistics

Before we can look at group means we have to tell R that "group" is a categoric variable, otherwise it will think it is continuous. We could have avoided this step by using letters instead of numbers for the different levels of factor A, but this is a learning exercise, so go ahead and enter the following command:

```
> as.factor(group)
 [1] 1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5
Levels: 1 2 3 4 5          # this tells us we have five levels of factor "group", labeled 1,2,3,4, and 5
```

There are several ways to look at descriptive statistics about our data:

```
> summary(score)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  2.00    3.00    5.00    4.72    6.00    8.00

> mean(score)     # this is the mean of score
[1] 4.72
> sd(score)       # this is the standard deviation of score
[1] 1.768238
> var(score)      # this is the variance of score
[1] 3.126667
> length(score)   # this tells us how many score measurements we have, i.e. how many participants
[1] 25
```

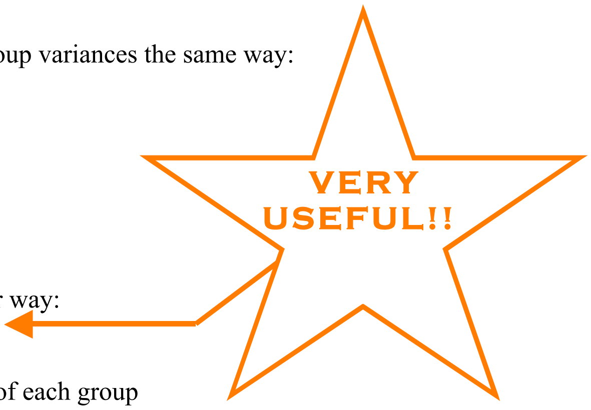
There are several methods we could use to find the **marginal means,** or **group means**. For example, to find the mean score for all participants in level 1 only, we could

1. make a new variable called "group1"
2. store in it all the "score" values for only those rows where group is equal to 1
3. calculate the mean of "group1"

```
> group1 = score[group=="1"]        # put the 1 in quotes because "group" is a categoric variable
> group1                            # this variable is a list of values, also called an array or a vector
[1] 8 6 7 7 6
> group1.mean = mean(group1)        # don't be frightened by the period in the variable name
> group1.mean
[1] 6.8
```

We can find the group standard deviations and group variances the same way:

```
> group1.sd = sd(group1)
> group1.sd
 [1] 0.83666
> group1.var = var(group1)
> group1.var
[1] 0.7
```



That's all well and good, but there's a much faster way:

```
> groups.mean = tapply(score, group, mean)
> groups.mean
  1    2    3    4    5      # these are the names of each group
6.8  4.6  3.8  2.8  5.6      # these are the mean scores for each group
```

The tapply function can also be used to calculate the group standard deviation (sd), variance (var), length, etc.

3. Perform a one-way between-groups ANOVA

Sure the group means are different. But are they *significantly* different?

```
> anova1 = aov(score~as.factor(group), data=exdata)
```
NB: if 'group' is not a factor and has numerical values, then the 'aov' function will consider 'group' as a continuous variable and will not give you the results you want.

anova1	a variable name that will store the output of the "aov" function
aov( )	a function in R that can perform ANOVAs type "help(aov)" in the R terminal if you want to learn more about it
score~group	"score" is the DV and "group" is the IV, "group" must be a factor, which means that it should either have categorical values, or if it is numerical, it must be called a factor inside R.
data=exdata	tells R what data to use unnecessary here since "exdata" is already attached

```
> summary(anova1, intercept=T)
Df Sum Sq Mean Sq F value    Pr(>F)
(Intercept)       1 556.96  556.96  415.64 7.486e-15 ***
as.factor(group)  4  48.24   12.06    9.00 0.0002508 ***
Residuals        20  26.80    1.34
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

4. Calculate standard errors for the groups

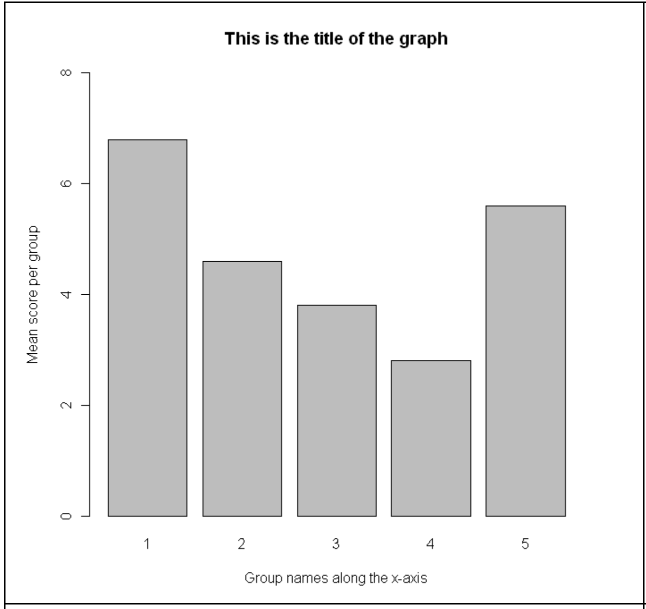
Before we make a graph showing the results of the ANOVA, we need to calculate the standard error for each group. Remember that se = sd/√n , where n is the number of independent observations in each group.

```
> groups.sd = tapply(score, group, sd)          # get the sd for each group
> groups.se = groups.sd/5^.5                    # divide by the square root of the number of observations per group (5)
> groups.se
        1         2         3         4         5 
0.3741657 0.5099020 0.7348469 0.3741657 0.5099020 
```

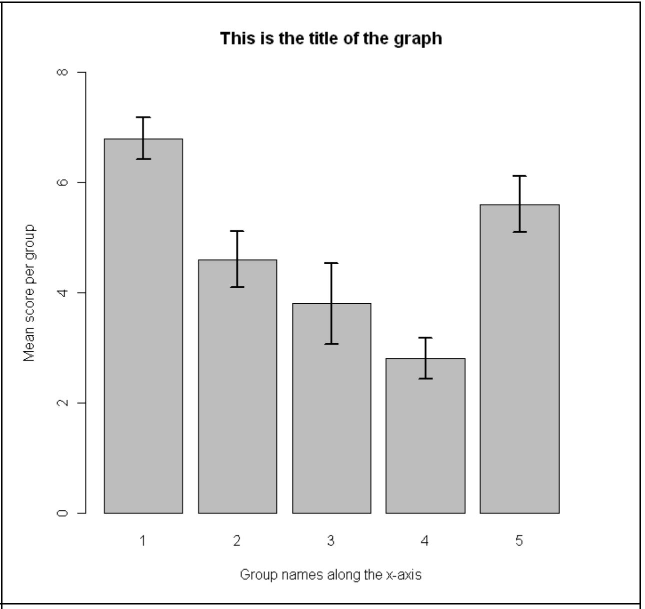
5. Make a graph of the results, including standard error bars

R has many powerful options for graphically displaying your data. **The handout titled "2x3_bargraph" covers some of those options in more detail.** Here we will just draw a basic graph (all one long command):

```
> graph1 = barplot(groups.mean, beside=T, ylim=c(0,8), main="This is the title of the graph", xlab="Group names along the x-axis", ylab="Mean score per group")
```



no error bars :(



error bars!

Of course no bar graph is complete without error bars. Using R's built in function *errbar( )* is a little tricky; it's actually easier to define our own function *superpose.eb( )* to do it. Don't worry about understanding exactly how it works. The *superpose.eb* function is discussed in (slightly) more detail in the "2x3_bargraph" handout.

```
> superpose.eb = function (x, y, ebl, ebu = ebl, length = 0.08, ...) arrows(x, y + ebu, x, y - ebl, angle = 90, code = 3,
   length = length, ...)                    # enter this exactly as is
> graph1.temp = graph1                      # this creates a new vector of the x-values of the bars on graph1
> superpose.eb(x=graph1.temp, y=groups.mean, ebl=groups.se, col="black", lwd=2)
```

Alternative method for one-way between-groups ANOVA (for the statistically curious)

We could also do this ANOVA using the general linear model, which is what R does behind the scenes when you use aov anyway.
Doing it this way reminds us of the relationship between ANOVA and Regression.

```
> model1 = lm(score~as.factor(group)) # fit a model predicting "score" as a function of "group"
> summary(model1)

Call:
lm(formula = score ~ as.factor(group))

Residuals:
   Min     1Q Median     3Q    Max
  -1.8   -0.8    0.2    0.4    2.2

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)        6.8000     0.5177  13.135 2.71e-11 ***
as.factor(group)2 -2.2000     0.7321  -3.005  0.00700 **
as.factor(group)3 -3.0000     0.7321  -4.098  0.00056 ***
as.factor(group)4 -4.0000     0.7321  -5.464 2.39e-05 ***
as.factor(group)5 -1.2000     0.7321  -1.639  0.11683
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.158 on 20 degrees of freedom
Multiple R-squared: 0.6429,     Adjusted R-squared: 0.5714
F-statistic:     9 on 4 and 20 DF,  p-value: 0.0002508
```

We can then ask R to show us the result of that linear model in familiar ANOVA terms:

```
> anova(model1)
Analysis of Variance Table

Response: score
                 Df Sum Sq Mean Sq F value    Pr(>F)
as.factor(group)  4  48.24   12.06       9 0.0002508 ***
Residuals        20  26.80    1.34
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```