
# Using R for Graphing Workshop
## Presenters: Jeremy Yagle and Jon Wayland
## Spring 2013

**R is a free software downloadable at [http://www.r-project.org/](http://www.r-project.org/)**

| Notes: | Code and Output: |
|---|---|
| **01 R Console Setup:**<br>**>** prompts you for formula or function.<br>The result appears on the next line(s). | |
| **02 R Use of the up-arrow:** ↑<br>For many of the following commands we'll be using, it will be easier to use the up-arrow on the keyboard to have R recall our last typed command. We'll remind you'll to do this with the green italicized text above the command. | **You'll see this instruction before some of the commands that follow in later steps. Using the up arrow saves you typing time and helps prevent typographic errors.**<br><br>*Use the ↑ key, and change only the highlighted areas below:* |
| **03 file.choose() command**<br>We'll start by directing R to find the Excel file that contains the data for our graph. We do this with the **file.choose( )** command. After you type the first line and hit enter, R will pop up a window that allows you to select the file. Find the file **US States Education Rates.csv** and double-click on it.<br><br>Next, we'll use **attach( )** to let R know that we want to work with this data. Use **head( )** to see the variable names.<br><br>==========<br>*Note: categories are listed in alphabetical order. Although we are not using it in these notes, you can change the order of the categories:*<br><br>*I.e. Instead of Region being ordered alphabetically "MW", "NE", "S", "W" , you can change the order to:*<br>*"NE", MW", "S", "W"*<br>*using the command:*<br>Region=factor(Region, levels=c("NE", "MW", "S", "W")) | > education=read.csv(file.choose())<br>> attach(education)<br>> head(education)<br>State Region Percent.HS.and.Above Percent.Bachelors.and.above<br>1 Alabama South 82.1 22.0<br>2 Alaska West 91.4 26.6<br>3 Arizona West 84.2 25.6<br>4 Arkansas South 82.4 18.9<br>5 California West 80.6 29.9<br>6 Colorado West 89.3 35.9 |

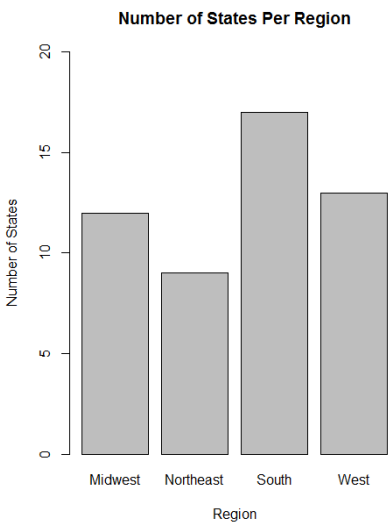
# Bar Charts

**04 generate a table of frequencies**
To organize the data for our chart, use the **table( )** command, and direct R to build a table that contains the number of states in each of the four regions.

```
> table(Region)
Region
Midwest Northeast     South      West
     12         9        17        13
```

**05 generate the basic bar chart**
Use the **barplot ( )** command, and specify that we want to use the data in **table(Region)** that we created above.

```
>barplot(table(Region))
```




**06 specify the scale on the axis**
The y-axis scale that R assigned to our graph isn't a good fit, so change the scale to go from 0 to 20, using the **ylim=c( )** argument . The boundaries for the scale are typed in the c( , )

*Use the ↑ key, and change only the highlighted areas below:*

```
> barplot(table(Region), ylim=c(0,20) )
```

**07 add a title**
Add a title to the graph with the **main=" "** argument. Each bar represents the number of states in the geographic region.

*Use the ↑ key, and change only the highlighted areas below:*

```
> barplot(table(Region),ylim=c(0,20), main="Number of States Per Region"
)
```

**08 modify the axes labels**
Label the y-axis. Similar to the argument above, use **ylab=" "** to specify what the y-axis values represent.

*Use the ↑ key, and change only the highlighted areas below:*

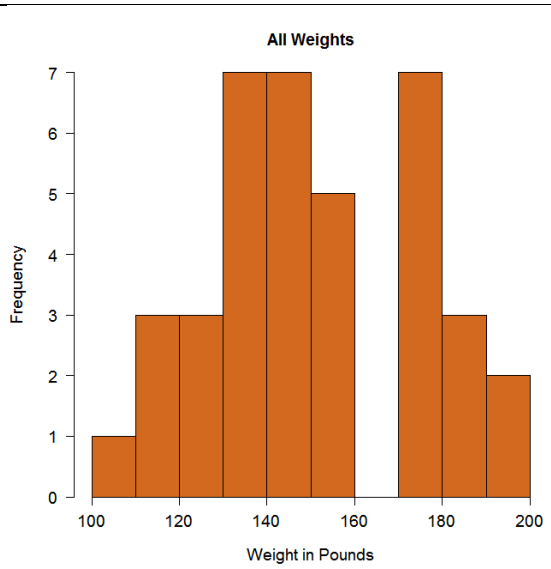
```
> barplot(table(Region),ylim=c(0,20), main="Number of States Per Region",
ylab="Number of States" , xlab="Region")
```

**09 file.choose( ) command**
To investigate other options for graphing data, switch to the data set we used in the Introduction workshop. Use **file.choose()** to direct R to find the Excel file that contains the data for our graphs.

After you type the first line and hit enter, R will pop up a window that allows you to select the file. Find the file **body.csv** and double-click on it.

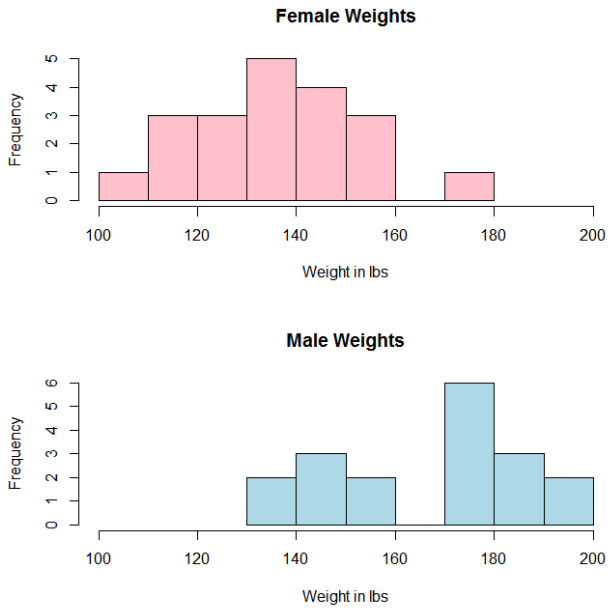
Use **attach( )** to let R know that we want to work with this data. Use **head( )** to see the variable names.

```
> body=read.csv(file.choose())



> attach(body)
> head(body)
  Gender Weight Height
1 Female    118   64.5
2   Male     NA   72.5
3   Male    143   73.3
4   Male    172   68.8
5 Female    147   65.0
6 Female    146   69.0
```

| | |
|---|---|
| **Histograms** |  |
| **10 create a histogram**<br>Use the **hist( )** command to plot a histogram of all weights of the people in the data set. | > hist(Weight) |
| **11 add a label and title**<br>Modify the command to include a title for the graph, and a label for the x-axis. Use the arguments **main=" "** to name our title, and **xlab=" "** to label the axis. | *Use the ↑ key, and change only the highlighted areas below:*<br><br>> hist(Weight, **main="All Weights", xlab="Weight in Pounds")** |
| **12 display available colors in R**<br>R has hundreds of available colors. To see a full listing of the color choices, use the **colors( )** command. Just a sample of the colors is shown here. | > colors()<br>[1] "white" "aliceblue" "antiquewhite"<br>[4] "antiquewhite1" "antiquewhite2" "antiquewhite3"<br>[7] "antiquewhite4" "aquamarine" "aquamarine1"<br>*…(and a lot more!)…*<br>[655] "yellow3" "yellow4" "yellowgreen" |
| **13 add color to the bars**<br>Add some color to our plot by using the **col=** argument. Here, we make chocolate-colored bars. You can experiment with other colors by changing the color name inside of the " ". | *Use the ↑ key, find this command, and change only the highlighted areas below:*<br><br>> hist(Weight, main="All Weights", xlab="Weight in Pounds", **col="chocolate")** |
| **14 rotate numbers on the y-axis**<br>Use the **las=1** argument to write all numbers on the axes parallel to the bottom of the plot.<br>Default: **las=0** : parallel to the axis | *Use the ↑ key, find this command, and change only the highlighted areas below:*<br><br>> hist(Weight, main="All Weights", xlab="Weight in Pounds", **col="chocolate", las=1)** |
| **15 increase the font size of the axes**<br>Use the **cex.axs=** and **cex.lab=** to amount by which the size of the axes numbers and labels should be scaled relative to the default.<br>So 0.8 means 80% of the default size and 1.2 means 120% of the default size. | *Use the ↑ key, find this command, and change only the highlighted areas below:*<br><br>> hist(Weight, main="All Weights", xlab="Weight in Pounds", **col="chocolate", las=1, cex.axis=1.2, cex.lab=1.2)** |

**Arranging two graphs on the same page**

**Example: Two Histograms aligned in a 2 row x 1 column format**




**16 creating new vectors**

Create two new vectors; one for weights of all female subjects, and one for weights of male subjects. To exclude the missing values, use **!is.na( )**

>fw=Weight[Gender=="Female" & !is.na(Weight)]

*Use the ↑ key, and change only the highlighted areas below:*

> **mw**=Weight[Gender=="**Male**" & !is.na(Weight)]

**17 divide the output window**

Use the **par ( )** command to change the set-up of the output window. By using **mfrow=c(2,1)**, we divide the window into two rows. This will give us a top and bottom plot. We could make it side-by-side histograms if we change it to **mfrow=c(1,2).**

> par(mfrow=c(2,1))

**18 create top/bottom plots**

Using the **hist ( )** command, create 2 histograms in our output window – one for females, and one for males.

Note that as you enter commands, R automatically places the graph in the next open spot in the output window.

> hist(fw,main="Female Weights", xlab="Weight in lbs", col="pink")

*Use the ↑ key, and change only the highlighted areas below:*

> hist(**mw**,main="**Male** Weights", xlab="Weight in lbs", col="light blue")

**19 scaling and labeling the x-axis**

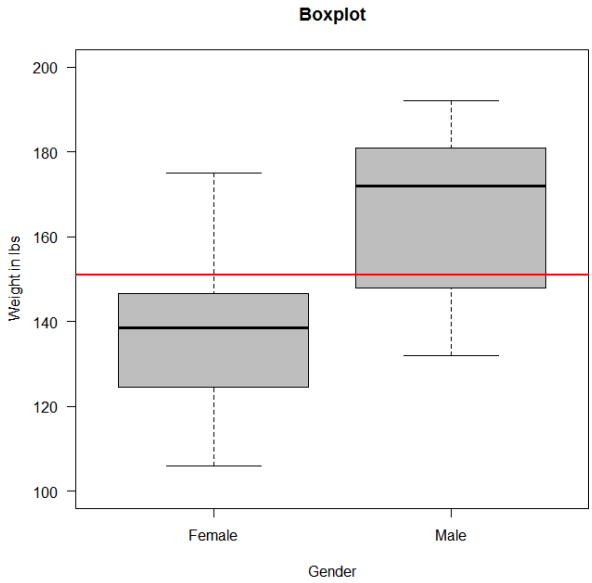
We can improve the quality of our graphs by keeping the scale of the x-axis consistent for both of our categories (female/male). Here we set the scale with the **xlim=c( )** argument.

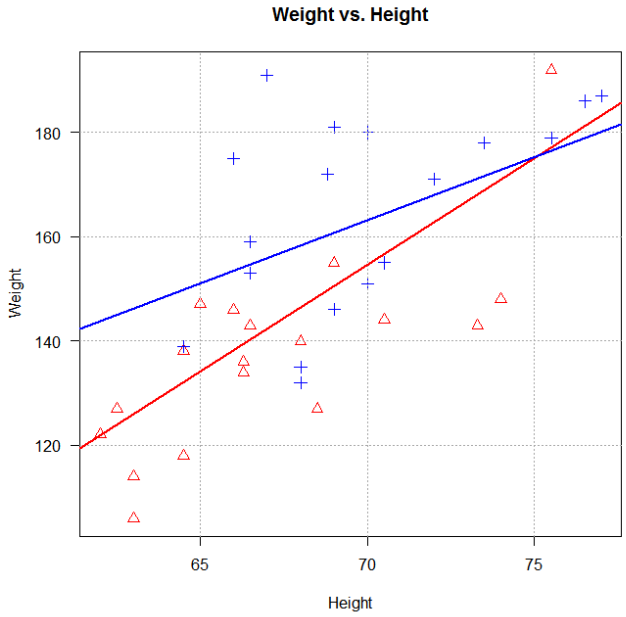
*Use the ↑ key to find the "Female Weights" command and change only the highlighted areas below:*

> hist(fw,main="Female Weights", xlab="Weight in lbs", col="pink", **xlim=c(100,200)**)

*Use the ↑ key, and change only the highlighted areas below:*

> hist(mw,main="Male Weights", xlab="Weight in lbs", col="light blue", **xlim=c(100,200)** )

| | |
|---|---|
| **Side-by-side Boxplots** |  |
| **20 Set the window to display one graph**<br>Change back to a single plot per figure. | >par(mfrow=c(1,1)) |
| **21 create a box plot**<br>Now, make a boxplot of the weights. When we use the **~** symbol in a command, we direct R to organize one variable in the data by another variable. (The **~** key is located to the left of the "1" key on the keyboard)<br><br>The range= argument sets the whiskers to go to the last points that are within 1.5*IQR of the quartiles | > boxplot(Weight~Gender, range=1.5) |
| **22 scale the y-axis**<br>We can change the scaling of the y-axis using the **ylim=c( )** argument. | *Use the ↑ key, and change only the highlighted areas below:*<br>> boxplot(Weight~Gender, range=1.5, **ylim=c(100,200))** |
| **23 add color**<br>Now, let's add in some color to the plots, using the **col=" "** argument. | *Use the ↑ key, and change only the highlighted areas below:*<br>> boxplot(Weight~Gender, range=1.5, ylim=c(100,200), **col="grey")** |
| **24 add a title and axes labels** | *Use the ↑ key, and change only the highlighted areas below:*<br>> boxplot(Weight~Gender, range=1.5, ylim=c(100,200),col="grey",<br>**main="Boxplot", xlab="Gender", ylab="Weight in lbs", las=1)** |
| **25 add a horizontal line at the overall mean**<br>Use the abline() command | >abline(h=mean(Weight, na.rm=T), col="red", lwd=2) |

| | |
|---|---|
| **Scatter Plots** |  |
| **26 create a scatter plot**<br>We can create a scatter plot of all heights vs. all weights using the features we used earlier | > plot(Weight~Height, las=1, main="Weight vs. Height", ylab="Weight in lbs", xlab="Height in inches") |
| **27 Modify the plotting symbols**<br>The **pch=** argument directs R to choose from 25 different plotting symbols. Here, we're using two different types - triangles and crosses.<br><br>For plotting symbols, see: http://www.statmethods.net/advgraphs/parameters.html | *Use the ↑ key, and change only the highlighted areas below:*<br><br>> plot(Weight~Height,las=1,main="Weight vs. Height", ylab="Weight in lbs", xlab="Height in inches", **col=c("red", "blue") , pch=2:3, cex=1.2)** |
| **28 add vertical grid lines**<br>Add grid lines to the graph using the **grid( )** command. The first argument inside the parentheses is the condition for vertical bars, and the second is for the horizontal bars (NA=off, NULL=aligned with scale values). | > grid(NA, NULL, col="darkgrey") |
| **29 add grid lines in both directions**<br>By changing both arguments to **NULL**, we will have vertical and horizontal grid lines on the graph. | > grid(NULL, NULL, col="darkgrey") |
| **30 add a fit line**<br>We can also add the fit lines to our graph for each gender. To do so, we first need to generate the line using the **lm( )** command. We'll assign the result of that command to a variable we'll call lmF and lmM<br><br>Now, we can put the line the on plot with the **abline( )** command. | > lmF=lm(Weight[Gender=="Female"]~Height[Gender=="Female"])<br>> lmM=lm(Weight[Gender=="Male"]~Height[Gender=="Male"])<br><br>> abline(lmF, col="red", lwd=2)<br>> abline(lmM, col="blue", lwd=2) |

| | |
|---|---|
| **31 save your graph as a .png file**<br>While it's possible to copy/paste the graph from the output window, saving the graph in an image file format will make it more efficient to work with, and more professional-looking in your documents. | |
| First, use **getwd( )** to make sure you are saving to the correct destination. | > getwd()<br>[1] "C:/Users/*Your user name*/Documents" |
| Then, choose what format you want to save your graph in; (pdf, types jpeg, bmp, or tiff). In this example, we will save the graph a .png file, using the command **png( )** | > png("weight height.png") |
| Next, enter the R command to generate the graph you wish to save. (NOTE: you will not see the graph in the output window, since the information that generates the graph is being saved instead of being displayed.) | > plot(Weight~Height) |
| Finally, type in the command **dev.off( )**, which saves the graph in the file format you specified, and writes the file to the location you specified. | > dev.off()<br>windows<br>2 |