UT-KBRIN Bioinformatics Summit R Workshop
Module 1: Introduction to R and RStudio, Reading Data into R

Arnold Stromberg (astro11@email.uky.edu)
Katherine Thompson (katherine.thompson@uky.edu)
Joshua Lambert (joshua.lambert@uky.edu)
Department of Statistics, University of Kentucky

Introduction to R

- *Getting Started:* R is a free software environment for statistical computing and graphics and can be downloaded from `http://www.r-project.org/`.
- According to wikipedia.com: "The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls and surveys of data miners are showing R's popularity has increased substantially in recent years."
- *Advantages of R:* The R language is part of the GNU project which means that
  - the program is freely distributed,
  - the source code is available, and
  - any users can submit code/libraries so that other users can use the methods they have developed
- There are two (related) ways you can use R:
  1. you can simply write commands and use the preloaded functions already included, or
  2. you can write your own functions.
- In either case, it is generally a bad idea to type commands directly into R, since these commands are often hard to track. Also, if a mistake is made in a command, it is hard to find and fix.
- *Instead*, use a text editor to write a script (e.g. filename.R) and either copy and paste the commands into R or use the `source()` function to run the script in R. If you use Windows, editors such as RStudio exist.

Introduction to RStudio

- *Getting Started:* RStudio is a free R-editor that can be used along with R. It can be downloaded from `http://www.rstudio.com/`. RStudio can be found under the start menu and the programs tab.
- There are four panels in the main RStudio window.
  1. *Console:* This is the place you can type R commands line-by-line.
  2. *Script Window:* This is where you can type R commands and save them so that you can reproduce or reanalyze your results.
     - To run commands, highlight the code you want to run and press `Ctrl + R` or click "Run" in the upper right hand corner of the panel.
  3. *Workspace/History:* Workspace shows all of the variables currently loaded in RStudio. History gives a list of all of the commands you have typed in this R Session.
  4. *Various Extra Features:* The two tabs I use most often are "Plots," which shows the current plot from R, and "Help," which displays the help for a function already built-in to R.

Help within R

The help files (as well as google) are very useful when learning about functions.

- If you know a function name (for instance, `mean()`) you can use either `help(mean)` or `?mean`.
- If you do not know a function name, search for applicable functions for what you want to do using either `help.search("mean")` or `??mean`.

There are *many* other resources for general help with R.

Using Projects in RStudio

RStudio allows users to create a project, which is a way to create and organize an ongoing data analysis for each data set of interest. Any data you read in to R will be associated with the project you are working on and will be loaded automatically the next time you open this project.

Creating a Project:

- Click the arrow in the upper right hand corner of the program next to "Project: (None)". Then click "New Project" and "Existing Directory". Next, click "Browse" and go to the folder containing the data you want to analyze.

Opening an Existing Project:

- To open a project, click the arrow next to "Project" in the upper right hand corner of the screen, then click "Open Project" and navigate to the project folder.

Setting the Working Directory

Before reading in data, it is convenient to set a "working directory." This specifies a default location for you to read files in from and write files to during a session. Using RStudio, you can set the working directory in two ways, a menu-driven way or by command.

Using Menus:

- In the bottom right window of the console, check to make sure the folder containing the data is shown. (If the folder is not shown, click "..." in the upper right corner of this window, and browse to the location of the folder.) Click "More" and "Set As Working Directory".

Using Commands:

Use the function `setwd()` to set the working directory path (see example below).

```
setwd("C:/Users/ukystat/Dropbox/UT-KBRIN R Workshop/") # Command to set working directory
getwd() # Displays current working directory

## [1] "C:/Users/ukystat/Dropbox/UT-KBRIN R Workshop"
```

Notes:

- The "/" are forward slashes instead of backslashes here! Two backslashes, "\\" will also work.
- The function `getwd()` will display your current working directory.
- Using `file.choose()` will bring up a window so that you can browse directly to the file you are reading in and use this path.
- Notice that some of the lines here start with a # symbol. This is the **comment character** in R. If a # symbol is found, R prints the rest of the line, but does not evaluate the code. This is so users can make notes and comments in their code about what their script is doing.

Reading in data

Each time you analyze data in R, you will need to call in the data at the beginning of your script. The two functions I most commonly use are `read.table()` and `write.table()`.

```
read.table() # Reads data into R from a file
scan() # Reads data into R from a file (good when you need to specify a data type
       # for each column.)
write.table() # Writes data from R to a file
```

Note: There are other "flavors" of `read.table` that we will not use (such as `read.csv`) since `read.table` is flexible enough (if you change the arguments) to include comma delimited data.

In RStudio: Select Tools (or File in newer RStudio versions) – Import Dataset – From...

Example: Let's read in some practice data. The data file is 'practicedata.txt' and can be downloaded from `http://web.as.uky.edu/statistics/users/klthomd/practicedata.csv` or `https://shiny.as.uky.edu/workshop/practicedata.csv`.

```
practicedata = read.table('practicedata.csv', # Give filename first
               header=TRUE, # If filename has variable names, set header to TRUE.
                     # Otherwise, use header=FALSE
               sep=",", # Symbol separating data values (comma here)
               na.strings="NA" # Characters used to denote missing values
               #comment.char='#', # Character used to indicate comments in your file
               #skip=0, # number of lines of data file to skip before reading in data
               #nrows=1000 # maximum number of lines of data file to read in
                )
```

Once the data is read in, we can check to see what it looks like by clicking on 'practicedata' in the upper right panel of the RStudio window.

```
practicedata[1:5,] # Prints first 5 rows of the data
practicedata[,1:2] # Prints first 2 columns of the data

practicedata[,"expvar"] # One way to call the variable, expvar
practicedata$expvar # Another way to call the variable, expvar
```

Suppose the first 50 data points are from a control group and the last 50 are from a treatment group, and you want to consider only the treatment group. This means you need to define a new variable (we'll call it `trtmtdata`) containing only the data associated with the treatment group.

```
k=50 # number of observations in each group
n=100 # total number of observations
trtmtdata = practicedata[(k+1):n, ] # Save the 51st through 100th rows of the data
```

Alternatively, we can use the `subset` function to subset the data according to the group variable. This does not depend on the ordering of the data.

```
trtmtdata = subset(practicedata , groupvar=='Treatment')
controldata = subset(practicedata , groupvar=='Control')
```

Writing Data to Files

To write data to a file, the function `write.table()` is very flexible in terms of data formatting in the new file. As a default, the new file is created in the current working directory. For example, suppose you want to write the variables, `expvar, groupvar`, and the natural log of `respvar` to a new file. (Although, by saving your R script, it is not necessary to save the natural log of your data. If you need it again, you can re-run that line of code.)

```
getwd() #Check current working directory

## [1] "C:/Users/ukystat/Dropbox/UT-KBRIN R Workshop - Leader/Handout"
```

```r
resp.log=log(practicedata$respvar) # take the natural log of the response variable
data.to.write=cbind(practicedata$expvar,practicedata$groupvar,resp.log)
          # Binds the columns (variables) together
colnames(data.to.write) # print out column names of the new data
```

```
## [1] ""         ""         "resp.log"
```

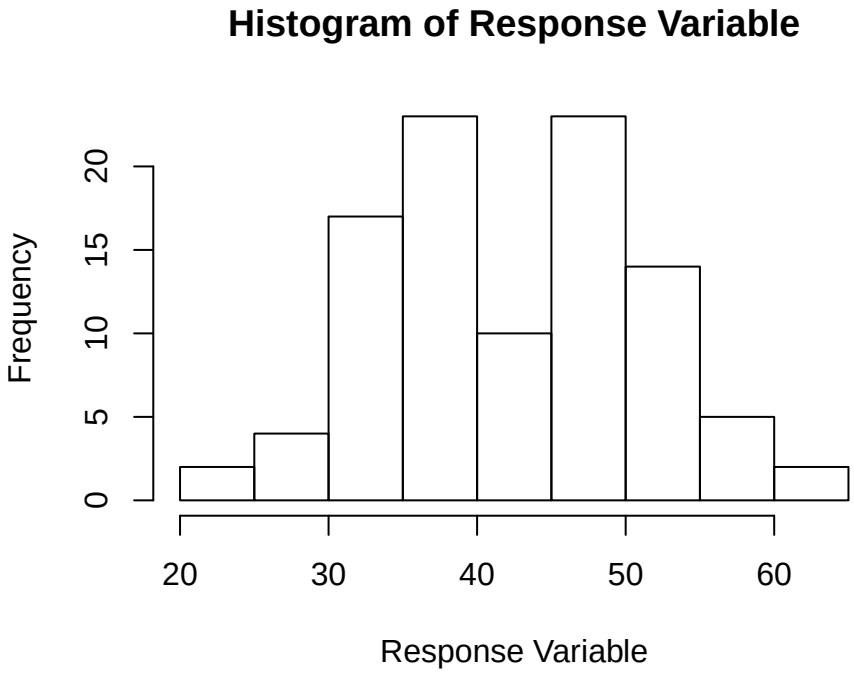
```r
colnames(data.to.write)<-c('expvar','groupvar','logrespvar') # rename the columns of
          # the new dataset
colnames(data.to.write) # print out column names of the new data again
```

```
## [1] "expvar"     "groupvar"   "logrespvar"
```

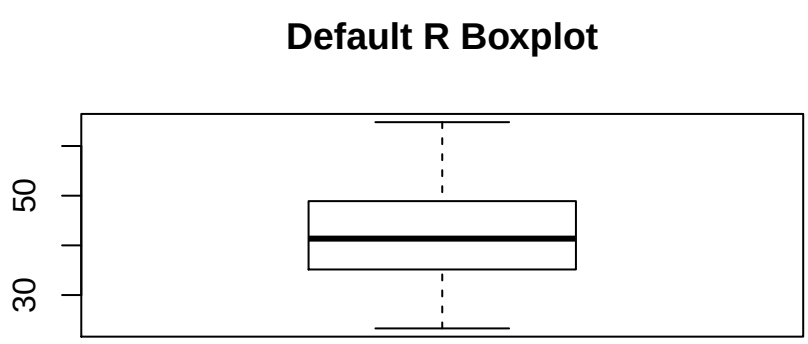
```r
##To write data to a new .csv file:
write.table(data.to.write, # data to write to a file
  file='logdata.csv', # name of file you want to save data in
  quote=FALSE, # whether or not to put quotations around data
  col.names=TRUE, # whether or not to write column names to file
  row.names=FALSE, # whether or not to write row names to file
  sep=',', # what you want to put between data entries (commas and spaces are common)
  append=FALSE, # whether or not to append existing data to the current file
  na='NA' # string to use for missing values
  )
```

Using Built-in Plotting Functions

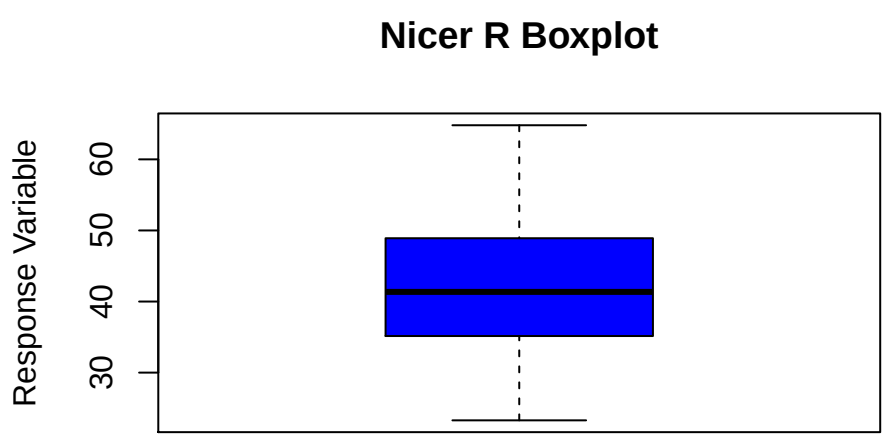
```
########################################
## Plotting One Quantitative Variable
########################################
## Creating a histogram
hist(practicedata$respvar, # what the histogram is plotting
           main='Histogram of Response Variable', # change the main axis title
           xlab='Response Variable' # change the x-axis label
           #, freq=TRUE # histogram is of counts
           # breaks="Sturges", # this can be changed to specify a series of points
                # for the breaks in the histogram
           # xlim=c(20,70), # sets the limits of the x-axis
           # ylim=c(0,25), # sets the limits of the y-axis

           )
```



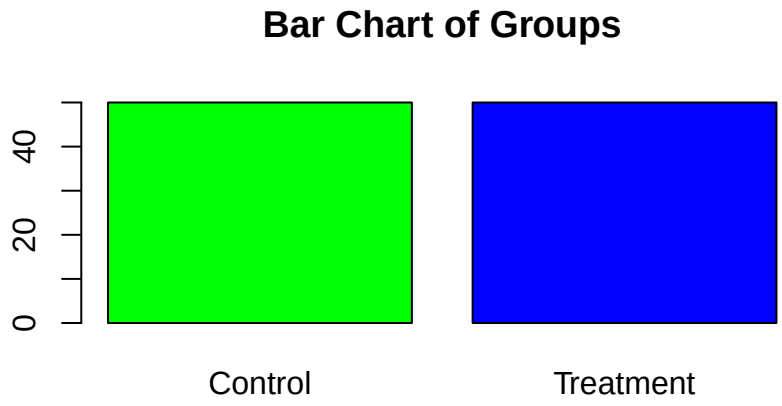
```r
## Creating a boxplot:
boxplot(practicedata$respvar, # what the boxplot is plotting
          main='Default R Boxplot' # change the main title
          )
```

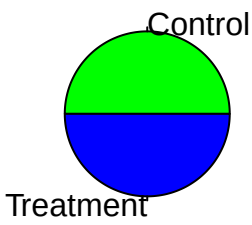



```r
## Creating a ''nicer'' boxplot
  boxplot(practicedata$respvar, # what the boxplot is plotting
          main='Nicer R Boxplot', # change the main title
          ylab='Response Variable', # change the y-axis label
          names='All Data', # change the name under the boxplot
         col='blue', # change the color of the boxplot
          outline=TRUE # Draw outliers if there are any in the data
          )
```

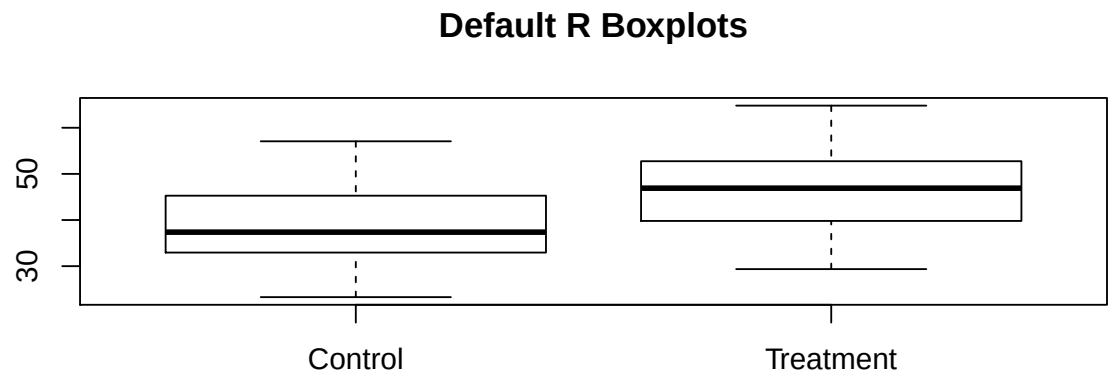


```
########################################
## Plotting One Categorical Variable
########################################
par(mfrow=c(2,1)) # Put two plots in one figure
## Bar Chart
plot.group<-table(practicedata$groupvar) # Create table of counts
barplot(plot.group, # Bar chart of the variable
        main='Bar Chart of Groups', # change main title
        col=c('green','blue') # change color of each bar
        )
## Pie Chart:
pie(plot.group, # Pie chart of the variable
    main='Pie Chart of Grouping Variable', # change main title
    col=c('green','blue')) # change the color of each slice of the pie
```

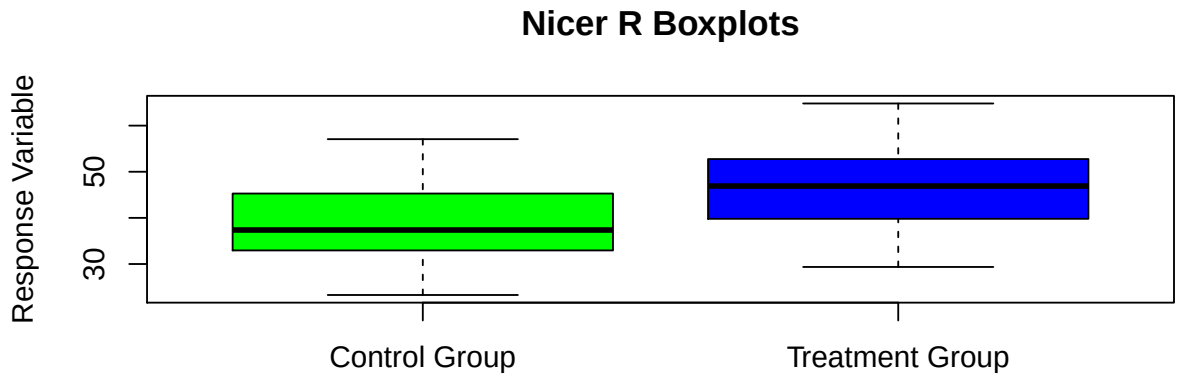




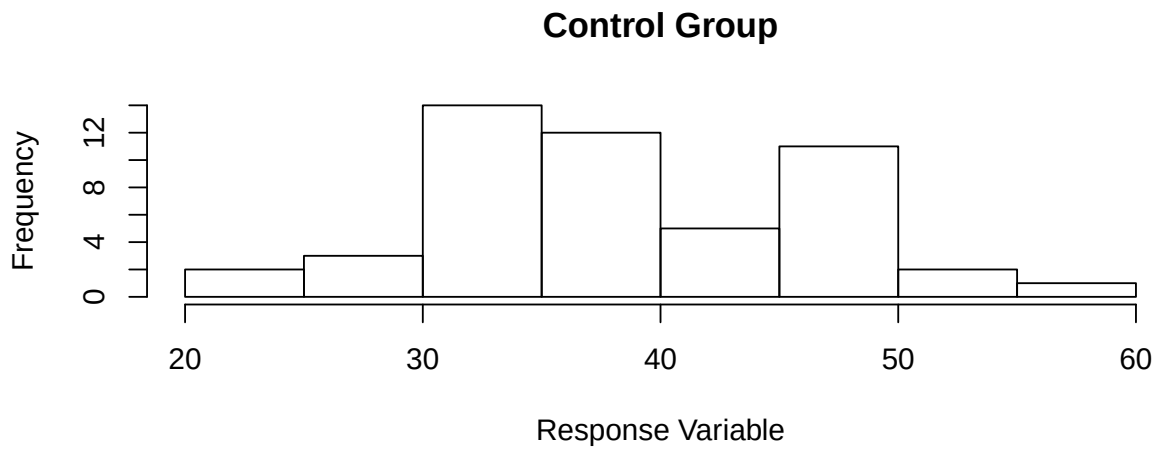
```
##########################################
## Plotting Two Variables
##########################################

#### Plotting One Quantitative and One Categorical Variable:

#### Side-by-side boxplots
boxplot(practicedata$respvar~practicedata$groupvar, main='Default R Boxplots')
```



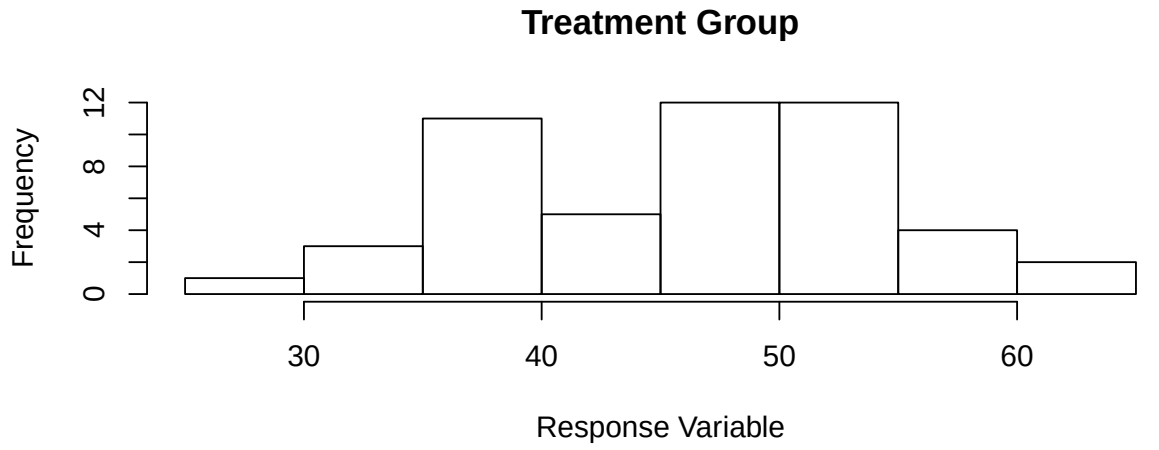

```
boxplot(practicedata$respvar~practicedata$groupvar,
        main='Nicer R Boxplots',
        names=c('Control Group', 'Treatment Group'),
        col=c('green','blue'), # change color of boxes
        ylab='Response Variable' # change y-axis label
        # horizontal=TRUE # change boxplots to horizontal
)
```



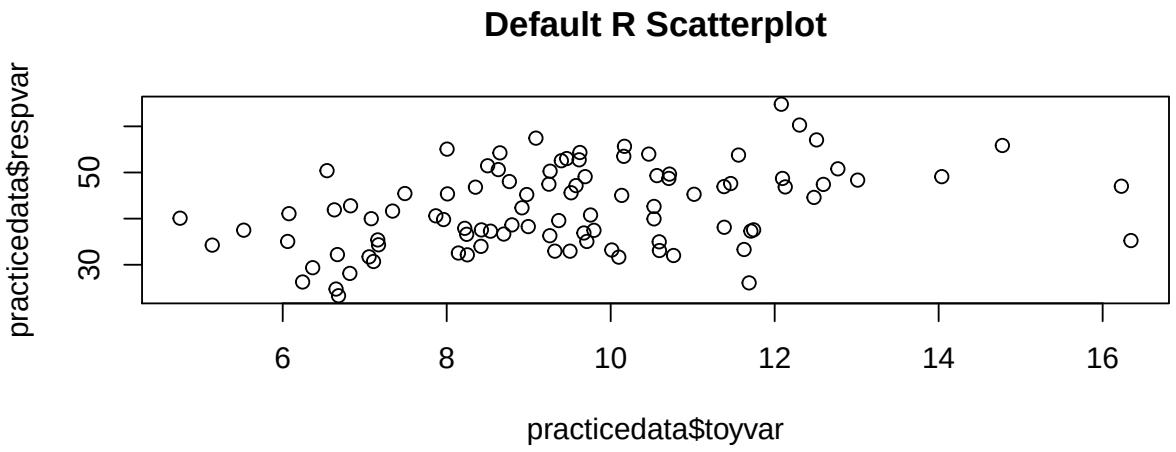
```r
#### Two Histograms:
# Create a two data subsets for control and treatment individuals
n.controls=50
n.treatments=50
# Extract the first n.controls rows from the data
controldata=practicedata[1:n.controls,]
# Extract everything except the first n.controls rows from the data
treatmentdata=practicedata[-(1:n.controls),]
## Histograms for each group
hist(controldata$respvar,
    main='Control Group', # change the main title
    xlab='Response Variable'
    )
```



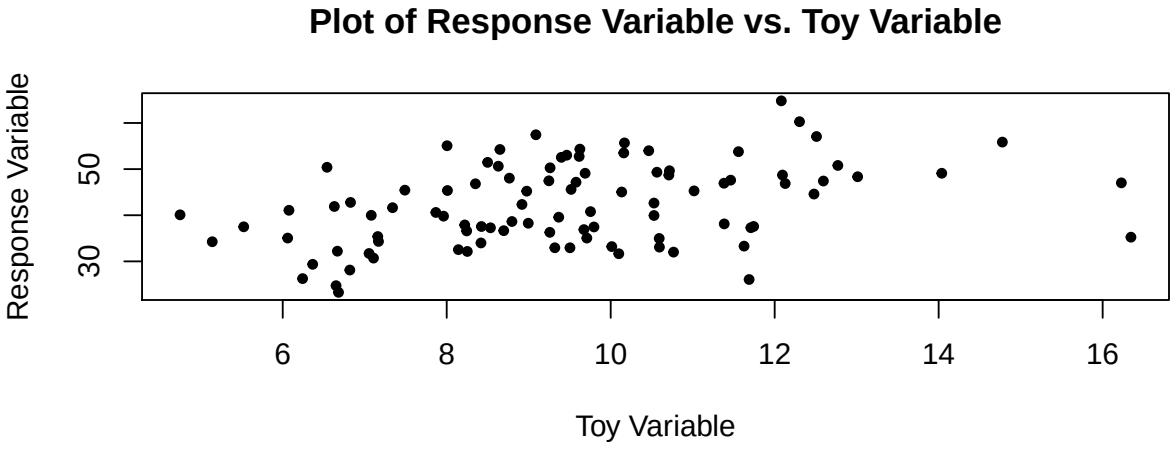

```r
hist(treatmentdata$respvar,
    main='Treatment Group', # change the main title
    xlab='Response Variable'
    )
```



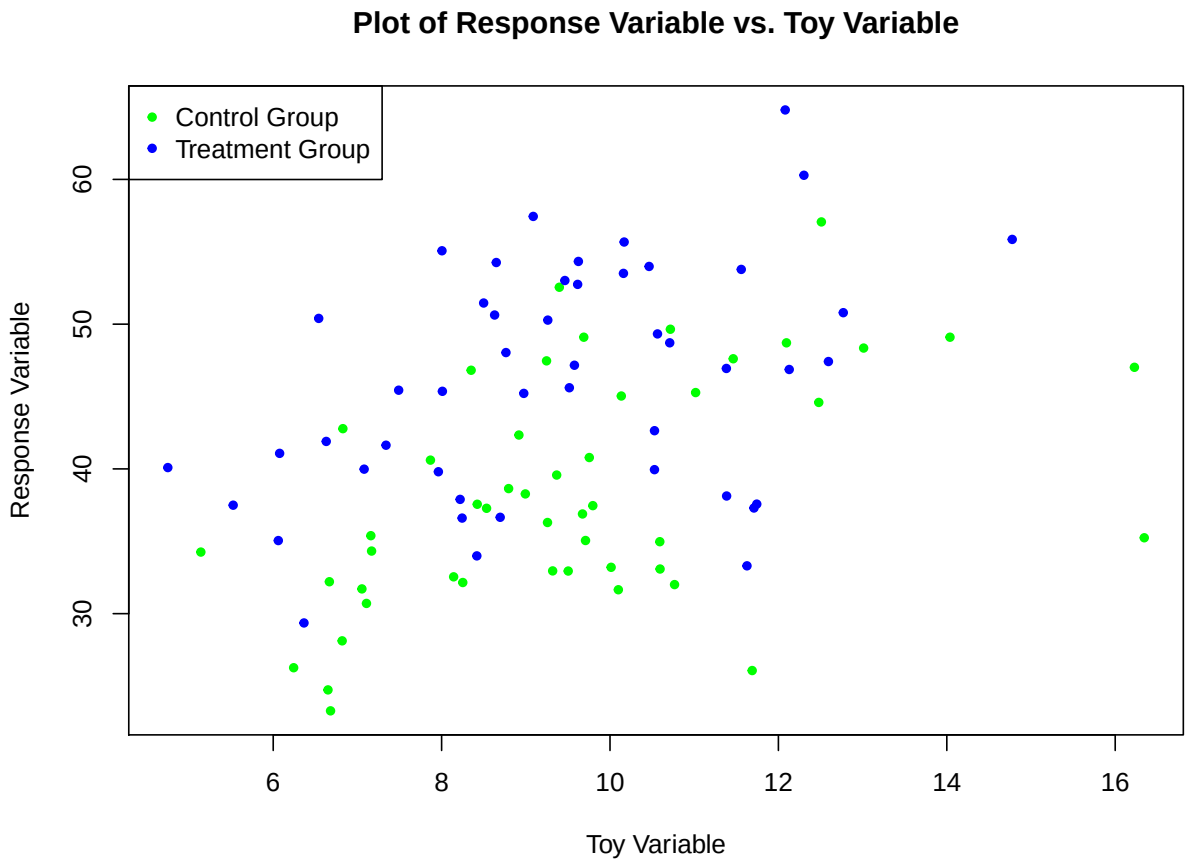
```r
#### Plotting two quantitative variables using a scatterplot:
plot(practicedata$toyvar,practicedata$respvar, # x variable, y variable
    main="Default R Scatterplot")
```



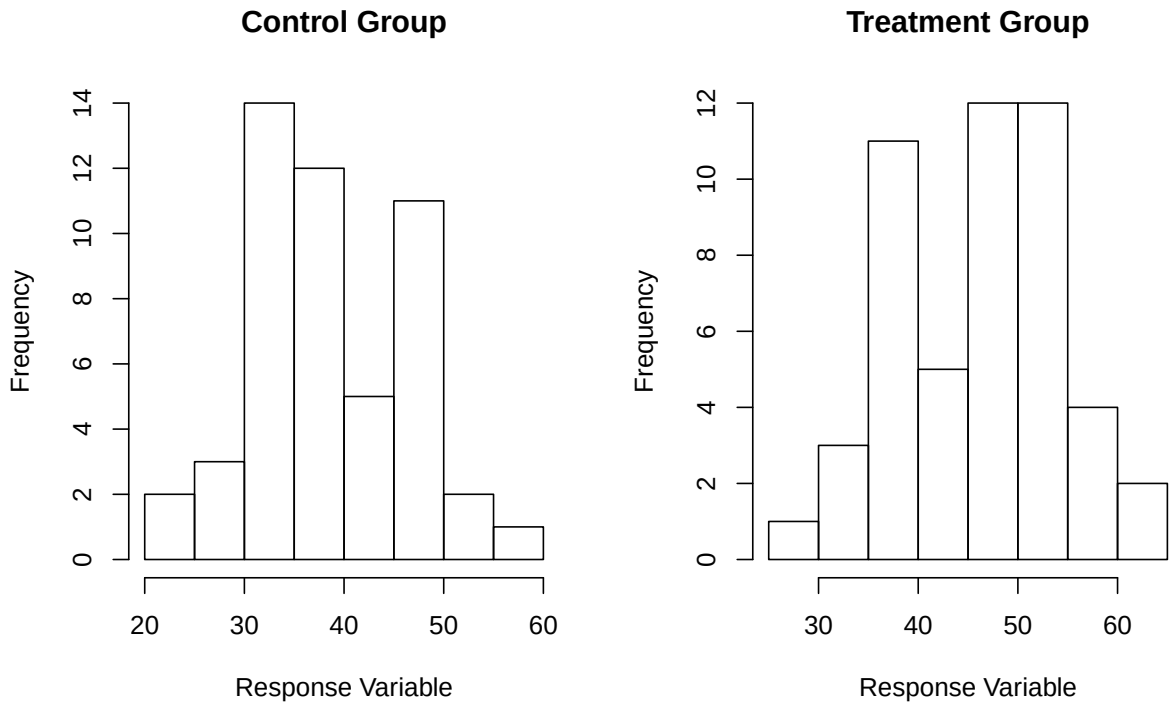

```r
plot(practicedata$toyvar,practicedata$respvar, # x variable, y variable
    main="Plot of Response Variable vs. Toy Variable", # change main label
    ylab='Response Variable', # change y-axis label
    xlab='Toy Variable', # change x-axis label
    pch=20 # change the plotting symbol
    #, type='l', # instead of pch, you can create a line plot
                # (make sure your x's are ordered if you do this.)
    # col='black' # change color of plotting symbol
     )
```



```
########################################
## Plots Showing More than Two Variables
########################################

## Find the range of the variables we are plotting for BOTH groups
ydatarange=range(practicedata$respvar,na.rm=TRUE)
xdatarange=range(practicedata$toyvar,na.rm=TRUE)

## Create scatterplot with all data
plot(controldata$toyvar,controldata$respvar, # x variable, y variable
    main="Plot of Response Variable vs. Toy Variable", # change main label
    ylab='Response Variable', # change y-axis label
    xlab='Toy Variable', # change x-axis label
    pch=20, # change the plotting symbol
    col='green', # change color of plotting symbol
    xlim=xdatarange, # change the x-axis to cover the range
            # of both the treatment and control groups
    ylim=ydatarange, # change the y-axis to cover the range
            # of both the treatment and control groups
    type='n' # create the plotting window, but not the points
    )

## Add observations for the control data to the plot
points(controldata$toyvar,controldata$respvar, # x variable, y variable
    pch=20, # change the plotting symbol
    col='green' # change color of plotting symbol
  )
## Add observations for the treatment data to the plot
points(treatmentdata$toyvar,treatmentdata$respvar, # x variable, y variable
       pch=20, # change the plotting symbol
       col='blue' # change color of plotting symbol
)

## Adding a Legend to the Plot
legend('topleft', # location of legend
       legend=c('Control Group','Treatment Group'), # lines of text in the legend
       pch=20, # symbol used in the legend
       col=c('green','blue') # colors of the symbol in the same
              # order as the lines of text in 'legend'
       # lty=1, lwd=1 # change the line type and width if lines are on the plot
       )
```


Plot of Response Variable vs. Toy Variable
Control Group
Treatment Group
Response Variable
30
40
50
60
6
8
10
12
14
16
Toy Variable

```r
########################################
## Using par()
########################################

# Example: Using the par function to change the number of plots in the panel
par(mfrow=c(1,2))
  #mfrow=c(1,2) creates a plotting window with 1 row and 2 columns of plots
  #mfrow=c(2,1) creates a plotting window with 2 rows and 1 column of plots

#### Histograms for each group
hist(controldata$respvar,
    main='Control Group', # change the main title
    xlab='Response Variable'
    )
hist(treatmentdata$respvar,
    main='Treatment Group', # change the main title
    xlab='Response Variable'
    )
```



Two-Sample *t*-tests

```
  ## Find group means for response variable
  aggregate((practicedata$respvar)~practicedata$groupvar*practicedata$groupvar2,FUN=mean)
```

```
##   practicedata$groupvar practicedata$groupvar2 (practicedata$respvar)
## 1               Control                     A               39.92996
## 2             Treatment                     A               46.32736
## 3               Control                     B               37.24586
## 4             Treatment                     B               45.49279
```

```
  ## Find group standard deviations for response variable
  aggregate((practicedata$respvar)~practicedata$groupvar*practicedata$groupvar2,FUN=sd)
```

```
##   practicedata$groupvar practicedata$groupvar2 (practicedata$respvar)
## 1               Control                     A               8.523437
## 2             Treatment                     A               7.430197
## 3               Control                     B               7.160007
## 4             Treatment                     B               8.608150
```

```
  ## Perform a t-test
  t.test(respvar~groupvar,data=practicedata,
         var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  respvar by groupvar
## t = -4.6117, df = 98, p-value = 1.207e-05
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -10.472948  -4.171384
## sample estimates:
##   mean in group Control mean in group Treatment
##                38.58791                45.91007
```

```
  t.test(respvar~groupvar2,data=practicedata,
         var.equal=TRUE)
```

```
##
##  Two Sample t-test
##
## data:  respvar by groupvar2
## t = 1.0097, df = 98, p-value = 0.3151
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.698629  5.217300
## sample estimates:
## mean in group A mean in group B
##        43.12866        41.36932
```

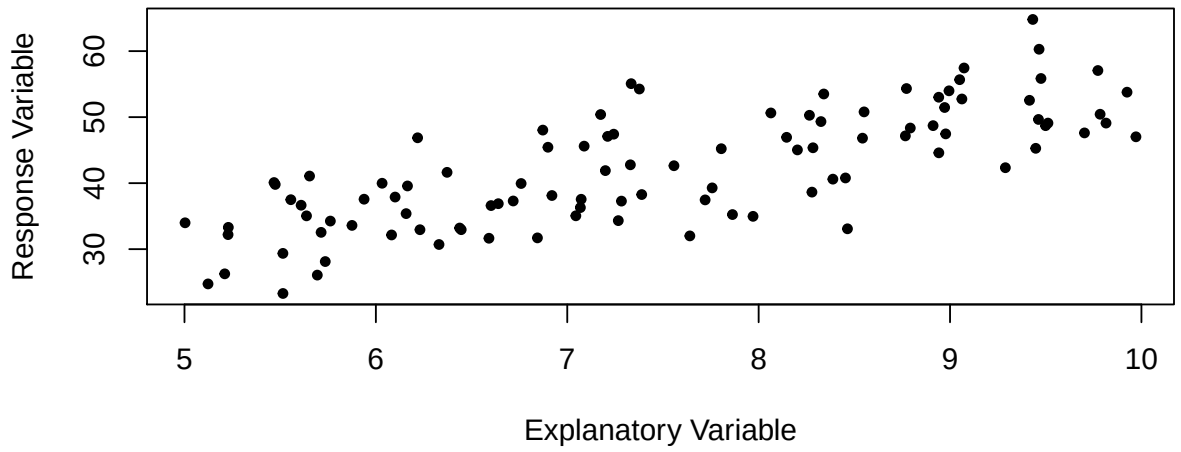
Analysis of Variance (ANOVA)

```
  ## Perform an ANOVA
  anova.results=lm(respvar~groupvar+groupvar2+groupvar*groupvar2,data=practicedata)

  anova.results

##
## Call:
## lm(formula = respvar ~ groupvar + groupvar2 + groupvar * groupvar2,
##     data = practicedata)
##
## Coefficients:
##                   (Intercept)              groupvarTreatment
##                        39.930                          6.397
##                    groupvar2B  groupvarTreatment:groupvar2B
##                        -2.684                          1.850

  anova(anova.results)

## Analysis of Variance Table
##
## Response: respvar
##                     Df Sum Sq Mean Sq F value    Pr(>F)
## groupvar             1 1340.4 1340.35 21.1727 1.283e-05 ***
## groupvar2            1   77.4   77.38  1.2223    0.2717
## groupvar:groupvar2   1   21.4   21.38  0.3377    0.5625
## Residuals           96 6077.3   63.31
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

  summary(anova.results)

##
## Call:
## lm(formula = respvar ~ groupvar + groupvar2 + groupvar * groupvar2,
##     data = practicedata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -16.6440  -5.7905   0.0287   5.7551  19.3054
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    39.930      1.591  25.093  < 2e-16 ***
## groupvarTreatment               6.397      2.250   2.843  0.00546 **
## groupvar2B                     -2.684      2.250  -1.193  0.23593
## groupvarTreatment:groupvar2B    1.850      3.183   0.581  0.56251
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.956 on 96 degrees of freedom
## Multiple R-squared:  0.1915,Adjusted R-squared:  0.1662
## F-statistic: 7.578 on 3 and 96 DF,  p-value: 0.0001333
```

Linear Regression

For this example, we will investigate the relationship between the variables `respvar` and `expvar` from the data set, `practicedata`. Remember than by using a '$', we can refer to the variable as `practicedata$respvar` in the following code. To fit a linear model, we use the function `lm()` as follows.

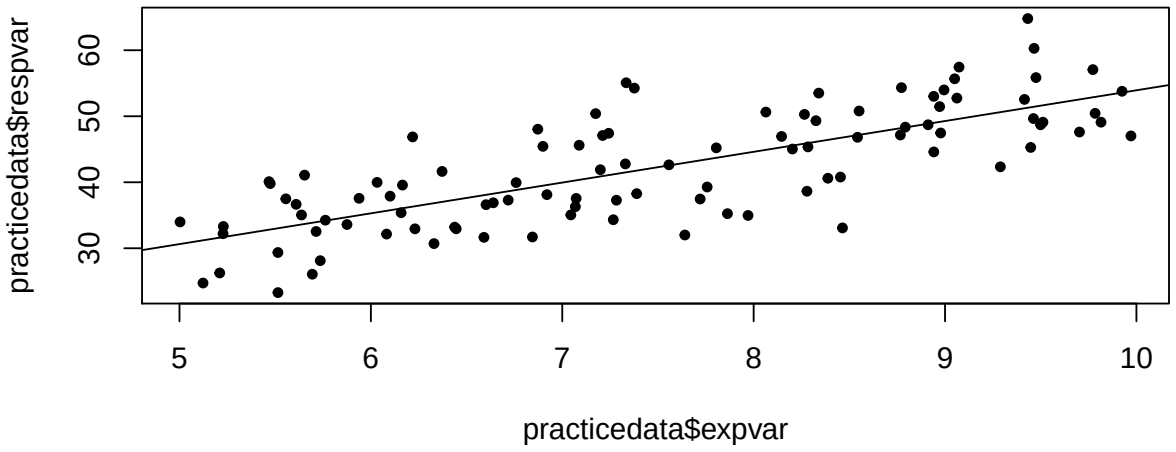
```
## Scatter plot of the data
plot(practicedata$expvar,practicedata$respvar,xlab='Explanatory Variable',
     ylab='Response Variable',pch=20)
```



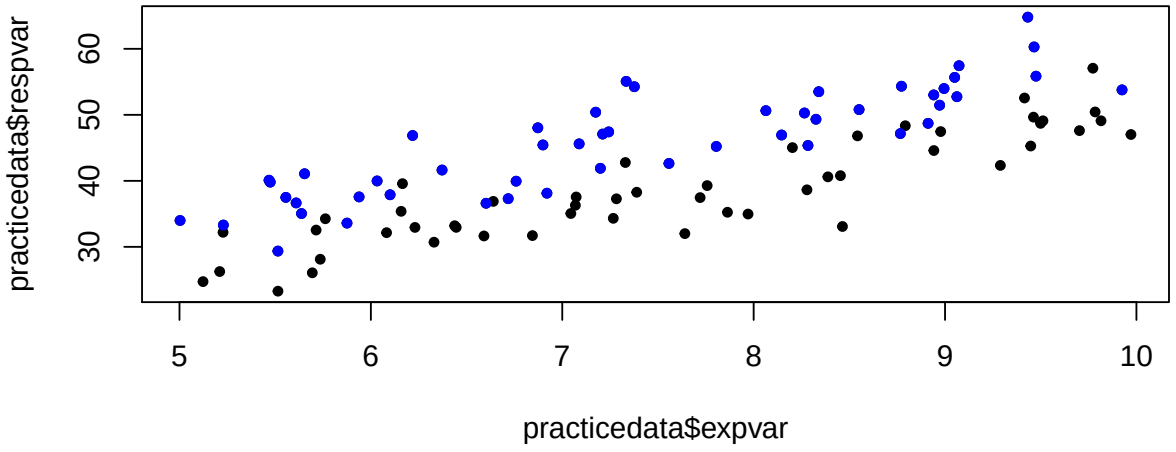

```
## Fit a linear model to the Data
fitted.model = lm(respvar~expvar,
                  data=practicedata) # Fit a linear model with
                  # y-variable respvar and x-variable expvar
summary(fitted.model) #Summarize the linear model

##
## Call:
## lm(formula = respvar ~ expvar, data = practicedata)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.6918  -4.1475  -0.2382   4.2102  13.5615
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   7.3299     3.0495   2.404   0.0181 *
## expvar        4.6604     0.3999  11.654   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.67 on 98 degrees of freedom
## Multiple R-squared:  0.5809,Adjusted R-squared:  0.5766
## F-statistic: 135.8 on 1 and 98 DF,  p-value: < 2.2e-16
```

```r
## Check to make sure that the model looks appropriate
plot(fitted.model)
```

```r
#Plot the data with the fitted regression line
plot(practicedata$expvar,practicedata$respvar,pch=20)
abline(fitted.model)
```




```r
##Plotting the data and the fitted model
plot(practicedata$expvar,practicedata$respvar,pch=20)
points(practicedata$expvar[practicedata$groupvar=='Treatment'],
       practicedata$respvar[practicedata$groupvar=='Treatment'],
       col='blue',pch=20)
```



```r
##You can also fit a linear model with more than one variable
fitted.group.model = lm(respvar~expvar + groupvar,
                        data=practicedata) # Fit a linear model with
                        # y-variable respvar and x-variables expvar and groupvar
summary(fitted.group.model) #Summarize the linear model
```

```
##
## Call:
## lm(formula = respvar ~ expvar + groupvar, data = practicedata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.7226 -2.4108 -0.1681  2.1640  9.4101
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)         1.4628     2.1175   0.691    0.491
## expvar              4.8846     0.2694  18.129   <2e-16 ***
## groupvarTreatment   8.3739     0.7640  10.960   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.809 on 97 degrees of freedom
## Multiple R-squared:  0.8128,Adjusted R-squared:  0.8089
## F-statistic: 210.5 on 2 and 97 DF,  p-value: < 2.2e-16
```

```r
##You can also fit a linear model with both variables and their interaction
fitted.int.model = lm(respvar~expvar + groupvar + expvar*groupvar,
                      data=practicedata) # Fit a linear model with
                      # y-variable respvar and
                      # x-variables expvar, groupvar and their interaction
summary(fitted.int.model) #Summarize the linear model
```

```
##
## Call:
## lm(formula = respvar ~ expvar + groupvar + expvar * groupvar,
##     data = practicedata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -9.6342 -2.4548 -0.0682  2.2381  9.4161
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)                2.2408     2.8744   0.780    0.438
## expvar                     4.7822     0.3714  12.875   <2e-16 ***
## groupvarTreatment          6.7406     4.1313   1.632    0.106
## expvar:groupvarTreatment   0.2182     0.5423   0.402    0.688
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.826 on 96 degrees of freedom
## Multiple R-squared:  0.8131,Adjusted R-squared:  0.8072
## F-statistic: 139.2 on 3 and 96 DF,  p-value: < 2.2e-16
```