Stat 3355
Statistical Methods for Statisticians and Actuaries

The notes and scripts included here are copyrighted by their author, Larry P. Ammann, and are intended for the use of students currently registered for Stat 3355. They may not be copied or used for any other purpose without permission of the author.

Software for Statistical Analysis

Examples presented in class are obtained using the statistical programming language and environment **R**. This is freely available software with binaries for Linux, MacIntosh, Windows that can be obtained from:
`http://cran.r-project.org`

A very useful extension for **R** is another freely available open-source package, **RStudio**. There are versions available for Windows, Mac, and Linux which can be downloaded from:
`https://www.rstudio.com/products/rstudio/download/`
Be sure to download the free version of this software. This package provides an intelligent editor for script files, it allows specific projects to be assigned to their own directories to aid in organization of your work, **RStudio** includes an interactive debugger to help identify and correct errors, and it provides an interactive GUI for easily exporting graphics to image files for incorporation into documents. Examples in class will be done using **RStudio**.

Syllabus

Stat 3355 Course Information

Instructor:	Dr. Larry P. Ammann
Office hours:	Tues, 2:30-3:30 pm, others by appt.
Email:	ammann@utdallas.edu
Office:	FO 2.410C
Phone:	(972) 883-2164
Text:	Course notes and web resources

Topics

- Graphical tools
- Numerical summaries
- Bivariate summaries
- Simulation

- Sampling distributions
- One sample estimation and hypothesis tests
- Two sample estimation and hypothesis tests
- Introduction to inference for regression and ANOVA

Notes
1. Very little course time will be spent on probability theory. The basic concepts of probability will be illustrated instead via simulations of Binomial and Normal distributions.
2. This course includes an introduction to **R**, a computer platform for data visualization and analysis. Bring your laptops to class on Thursdays until further notice. Those classes will be devoted to using **R**.

Grading Policy
Course grade will be based on quizzes, homework projects and the final project.

Note: the complete syllabus is available here:
`http://www.utdallas.edu/~ammann/stat3355_syllabus.pdf`

R Notes

The following links provide an excellent introduction to the use of **R**:
`https://cran.r-project.org/doc/contrib/Owen-TheRGuide.pdf`
`https://cran.r-project.org/doc/contrib/Robinson-icebreaker.pdf` (somewhat more advanced)
Other contributed books about **R** can be found on the **CRAN** site. Use the **Contributed** link under Documentation on the left side of the **CRAN** web page. Additional notes are provided below.

The **S** language was developed at Bell labs as a high-level computer language for statistical computations and graphics. It has some similarities with **Matlab**, but has some structures that **Matlab** does not have, such as *data frames*, that are natural data structures for statistical models. There are two implementations of this language currently available: a commercial product, *S-Plus*, and a freely available open-source product, **R**. **R** is available at
`http://cran.r-project.org`
These implementations are mostly, but not completely, compatible.

Note: in the examples below the **R** prompt, `>` , is included but this would not be typed on the command line. It is used here to differentiate between input to **R** and output that is returned to the console after a command is entered.

On Linux or Macs, **R** can be run from a shell by entering
`R`
at a shell prompt. The **R** session is ended by entering
`q()`
This will generate a query from **R** whether to save the workspace image. Enter `n`.

On Windows and Macs **R** is packaged as a windowed application that starts with a command window. **RStudio** also is a windowed application that includes a window for entering commands, a window that describes the property of objects that have been created during the session, and a window for graphics.

R's Workspace. The *Workspace* contains all the objects created or loaded during an **R** session. These objects only exist in the computer's memory, not on the physical hard drive and will disappear when **R** is exited. **R** offers a choice to the user when exiting: save the workspace or do not save it. If the Workspace is not saved, all objects created during the session will be lost. That's no problem if you are using it only as a mathematical or statistical calculator. If you are performing an analysis, but must exit before completing it, then you don't want to lose what you have already done. There is an alternative that I recommend instead of saving the workspace: write the commands you wish to enter into a text file and then copy/paste from the edit window into the **R** console. Even though this may seem like extra work, it has three advantages:

- Any mistakes can be corrected immediately with the editor.
- You won't have to remember what the objects in a workspace represent since the file

contains the commands that created those objects.

- If you need to perform a similar analysis at a later time, you can just copy the original file to a new name and modify/extend the commands in the new file to complete the analysis.

You must use a plain text editor to edit command files, not a document editor like `Word`. Both **R** and **Rstudio** include an editor for scripts that is accessed from their *File* menu.

Rstudio has an extensive set of resources to help users. Go to the *Help* tab on the right-hand window and click on **An Introduction to R** under *Manuals*. See section 2.1-2.7 for details about the following.

1. The basic data structure in **R** is a vector. This is a set of objects all of which must have the same mode, either *numeric, logical, character*, or *complex*.

2. Assignment is performed with the character `=` or the two characters `<-`. The second assignment operator is older but `=` is used more commonly now since it is just a single character. When an assignment is made, its value is not echoed to the terminal. Lines with no assignment do result in the value of the expression being echoed to the terminal.

3. Sequences of integers can be generated by the colon expression,

   ```
   > x = 2:20
   > y = 15:1
   ```

   More general sequences can be generated with the `seq()` function. These operations produce vectors. Some examples:

   ```
   > seq(5)
   [1] 1 2 3 4 5
   > x = seq(2,20,length=5)
   > x
   [1]  2.0  6.5 11.0 15.5 20.0
   > y = seq(5,18,by=3)
   > y
   [1]  5  8 11 14 17
   ```

   The function `c` can be used to combine different vectors into a single vector.

   ```
   > c(x,y)
    [1]  2.0  6.5 11.0 15.5 20.0  5.0  8.0 11.0 14.0 17.0
   ```

   All vectors have an attribute named *length* which can be obtained by the function `length()`

```
> length(c(x,y))
[1] 10
```

A scalar is just a vector of length 1.

4. A useful function for creating strings is `paste()`. This function combines its arguments into strings. If all arguments have length 1, then the result is a single string. If all arguments are vectors with the same length, then the pasting is done element-wise and the result is a vector with the same length as the arguments. However, if some arguments are vectors with length greater than 1, and the others all have length 1, then the other arguments are replicated to have the same length and then pasted together element-wise. Numeric arguments are coerced to strings before pasting. Floating point values usually need to be rounded to control the number of decimal digits that are used. The default separator between arguments is a single space, but a different separator can be specified with the argument, `sep=`.

```
> s = sum(x)
> paste("Sum of x =",s)
[1] "Sum of x = 55"
> paste(x,y,sep=",")
[1] "2,5"     "6.5,8"   "11,11"   "15.5,14" "20,17"
> paste("X",seq(length(x)),sep="")
[1] "X1" "X2" "X3" "X4" "X5"
```

5. Vectors can have names which is useful for printing and for referencing particular elements of a vector. The function `names()` returns the names of a vector as well as assigning names to a vector.

```
> names(x) = paste("X",seq(x),sep="")
> x
  X1   X2   X3   X4   X5
 2.0  6.5 11.0 15.5 20.0
```

Elements of a vector are referenced by the function `[]`. Arguments can be a vector of indices that refer to specific positions within the vector:

```
> x[2:4]
  X2   X3   X4
 6.5 11.0 15.5
> x[c(2,5)]
  X2   X5
 6.5 20.0
```

Elements can be referenced by their names or by a logical vector

```
> x[c("X3","X4")]
  X3   X4
11.0 15.5
> xl = x > 10
> xl
   X1    X2    X3    X4    X5
FALSE FALSE  TRUE  TRUE  TRUE
> x[xl]
  X3   X4   X5
11.0 15.5 20.0
```

The length of the referencing vector can be larger than the length of the vector that is being referenced as long as the referencing vector is either a vector of indices or names.

```
> ndx = rep(seq(x),2)
> ndx
 [1] 1 2 3 4 5 1 2 3 4 5
> x[ndx]
  X1   X2   X3   X4   X5   X1   X2   X3   X4   X5
 2.0  6.5 11.0 15.5 20.0  2.0  6.5 11.0 15.5 20.0
```

This is useful for table lookups. Suppose for example that `Gender` is a vector of elements that are either *Male* or *Female*:

```
> Gender
[1] "Male"   "Male"   "Female" "Male"   "Female"
```

and `Gcol` is a vector of two colors whose names are the two unique elements of `Gender`

```
> Gcol = c("blue","red")
> names(Gcol) = c("Male","Female")
> Gcol
  Male Female
"blue"  "red"
> GenderCol = Gcol[Gender]
> GenderCol
  Male   Male Female   Male Female
"blue" "blue"  "red" "blue"  "red"
```

This will be useful for plotting data.

6. **R** supports matrices and arrays of arbitrary dimensions. These can be created with the `matrix` and `array` functions. Arrays and matrices are stored internally in column-major order. For example,

   ```
   X = 1:10
   ```

   assigns to the object `X` the vector consisting of the integers 1 to 10.

   ```
   M = matrix(X,nrow=5)
   ```

   puts the entries of `X` into a matrix named `M` that has 5 rows and 2 columns. The first column of `M` contains the first 5 elements of `X` and the second column of `M` contains the remaining 5 elements. If a vector does not fit exactly into the dimensions of the matrix, then a warning is returned.

   ```
   > M
        [,1] [,2]
   [1,]    1    6
   [2,]    2    7
   [3,]    3    8
   [4,]    4    9
   [5,]    5   10
   ```

   The dimensions of a matrix are obtained by the function `dim()` which returns the number of rows and number of columns as a vector of length 2.

   ```
   > dim(M)
   [1] 5 2
   ```

7. Elements of matrices and arrays are referenced using `[]` but with the number of arguments equal to the number of dimensions. A matrix has two dimensions, so `M[2,1]` refers to the element in row 2 and column 1.

   ```
   > X = matrix(runif(100),nrow=20)
   > X[2:5,2:4]
   [1,] 0.731622617 0.6578677 0.7446229
   [2,] 0.023472598 0.2111300 0.7775343
   [3,] 0.001858455 0.2887734 0.8103568
   [4,] 0.269611100 0.7527248 0.2127048
   ```

   Note: the function `runif(n)` returns a vector of n random numbers between 0 and 1. If one of the arguments to `[,]` is empty, then all elements of that dimension are returned. So `X[2:4,]` gives all columns of rows 2,3,4 and so is a matrix with 3 rows and the same number of columns as `X`.

8. **Example**. The file `http://www.utdallas.edu/~ammann/stat3355scripts/sunspots.txt` contains yearly sunspot numbers since 1700. Note that the first row of this file is not data but represents names for the columns. This file is an example of tabular data. Such data can be imported into **R** using the function `read.table()`. Further details about his function are given below.

```
Sunspots = read.table("www.utdallas.edu/~ammann/stat3355scripts/sunspots.txt",heade
```

Note that the filename argument in this case is a web address. The argument also can be the name of a file on your computer. The second argument indicates that the first row of this file contains names for the columns. These are accessed by

```
names(Sunspots)
```

Suppose we wish to plot sunspot numbers versus year. There are several ways to accomplish this.

```
plot(Sunspots[,1],Sunspots[,2])
plot(Sunspots[,1],Sunspots[,2], type="l")
plot(Number ~ Year, data=Sunspots, type="l")
```

The last method uses what is referred to as the formula interface for the plot function. Now let's add a title to make the plot more informative.

```
title("Yearly mean total sunspot numbers")
```

To be more informative, add the range of years contained in this data set.

```
year.range = range(Sunspots["Year"])
title("Yearly mean total sunspot numbers, 1700-2016")
```

The title can be split into two lines as follows

```
title("Yearly mean total sunspot numbers\n1700-2016")
```

using the newline character `\n`. Note that this requires that we already know the range of years contained in the data. Alternatively, we could obtain that range from the data. That would make our command file more general. The following file contains the these commands:
`http://www.utdallas.edu/~ammann/stat3355scripts/sunspots.r`

9. **Lists**. A *list* is a structure whose components can be any type of object of any length. Lists can be created by the *list* function, and the components of a list can be accessed by appending a $ to the name of the list object followed by the name of the component. The dimension names of a matrix or array are a list with components that are the vectors of names for the respective dimensions. Components of a list also can be accessed by position using the `[[]]` function

```
> X = seq(20)/2
> Y = 2+6*X + rnorm(length(X),0,.5)
> Z = matrix(runif(9),3,3)
> All.data = list(Var1=X,Var2=Y,Zmat=Z)
> names(All.data)
[1] "Var1" "Var2" "Zmat"
> All.data$Var1[1:5]
[1] 0.5 1.0 1.5 2.0 2.5
> data(state)
> state.x77["Texas",]
Population     Income Illiteracy   Life Exp     Murder    HS Grad
   12237.0     4188.0        2.2       70.9       12.2       47.4
      Frost       Area
       35.0   262134.0
```

10. The dimension names of a matrix can be set or accessed by the function `dimnames()`. For example, the row names for `state.x77` are given by
`dimnames(state.x77)[[1]]`
and the column names are given by
`dimnames(state.x77)[[2]]`
These also can be used to set the dimension names of a matrix. For example, instead of using the full state names for this matrix, suppose we wanted to use just the 2-letter abbreviations:

```
> StateData = state.x77
> dimnames(StateData)[[1]] = state.abb
```

11. **Example**. Suppose we wanted to find out which states have higher Illiteracy rates than Texas. We can do this by creating a logical vector that indicates which elements of the Illiteracy column are greater than the Illiteracy rate for Texas. That vector can be used to extract the names of states with lower Illiteracy rates.

```
> txill = state.x77["Texas","Illiteracy"]
> highIll = state.x77[,"Illiteracy"] > txill
> state.name[highIll]
[1] "Louisiana"      "Mississippi"    "South Carolina"
```

12. **Matrix Operations**. Matrix-matrix multiplication can be performed only when the two matrices are conformable, that is, their inner dimensions are the same. For example, if A is $n \times r$ and B is $r \times m$, then matrix-matrix multiplication of A and B is defined and results in a matrix C whose dimensions are $n \times m$. Elementwise multiplication of two matrices can be performed when both dimensions of the two matrices are the same. If for example D,E are $n \times m$ matrices, then

```
F = D*E
```

results in an $n \times m$ matrix F whose elements are

$$F[i,j] = D[i,j] * E[i,j],\ 1 \leq i \leq n,\ 1 \leq j \leq m.$$

These two different types of multiplication operations must be differentiated by using different symbols, since both types would be possible if the matrices have the same dimensions. Matrix-matrix multiplication is denoted by $A\% * \%B$ and elementwise multiplication is denoted by $A * B$. The same situation occurs if A and B are both vectors that have the same length n. In that case, $A\% * \%B$ represents the *dot product* of these vectors,

$$A\% * \%B = \sum_{i=1}^{n} A_i B_i.$$

Note that this result is a scalar. $A * B$ represents elementwise multiplication. The result is a vector C with $c[i] = a[i] * b[i]$.

13. **Factors**. A *factor* is a special type of character vector that is used to represent categorical variables. This structure is especially useful in statistical models such as ANOVA or general linear models. Associated with a factor variable are its levels, the set of unique character values in the vector. Although print methods for a factor will by default print a factor as a character vector, it is stored internally using integer positions of the values corresponding to the levels.

14. A fundamental structure in the **S** language is the *data frame*. A data frame is like a matrix in that it is a two-dimensional array, but the difference is that the columns can be different data types. The following code generates a data frame named `SAMP` that has two numeric columns, one character column, and one logical column. It uses the function `rnorm` which generates a random sample for the standard normal distribution (bell-curve). Each time this code is run, different values will be obtained since each use of `runif()` and `rnorm()` produces new random samples.

```
> y = matrix(rnorm(20),ncol=2)
> x = rep(paste("A",1:2,sep=""),5)
```

```
> z = runif(10) > .5
> SAMP = data.frame(y,x,z)
           Y1         Y2  x     z
1   0.2402750  1.3561348 A1 FALSE
2   0.3669875 -1.4239780 A2 FALSE
3  -1.5042563  1.2929657 A1  TRUE
4   1.2329026  0.3838835 A2  TRUE
5  -0.1241536 -0.5596217 A1  TRUE
6  -0.1784147  1.2920853 A2 FALSE
7  -1.2848231  1.7107087 A1  TRUE
8   0.7731956  0.6520663 A2 FALSE
9  -0.3515564  0.3169168 A1  TRUE
10 -1.3513955  1.3663698 A2  TRUE
```

Note that the rows and columns have names, referred to as `dimnames`. Arrays and data frames can be addressed through their names in addition to their position. Also note that variable x is a character vector, but the *data.frame* function automatically coerces that component to be a factor:

```
> is.factor(x)
[1] FALSE
> is.factor(SAMP$x)
[1] TRUE
```

15. The **S** language is an object-oriented language. Many fundamental operations behave differently for different types of objects. For example, if the argument to the function `sum()` is a numeric vector, then the result will be the sum of its elements, but if the argument is a logical vector, then the result will be the number of TRUE elements. Also, the *plot* function will produce an ordinary scatterplot if its *x,y* arguments are both numeric vectors, but will produce a boxplot if the x argument is a factor:

```
> plot(SAMP$Y1,SAMP$Y2)
> plot(SAMP$x,SAMP$Y2)
```

A better way to produce these plots is to use the formula interface along with the *data=* argument if the variables are contained within a data frame.

```
> plot(Y2 ~ Y1, data=SAMP)
> plot(Y2 ~ x, data=SAMP)
```

16. **Reading Data from files**. The two main functions to read data that is contained in a file are *scan()* and *read.table()*.

scan(Fname) reads a file whose name is the value of `Fname`. All values in the file must be the same type (numeric, string, logical). By default, *scan()* reads numeric data. If the values in this file are not numeric, than the optional argument `what=` must be included. For example, if the file contains strings, then

```
x = scan(Fname,what=character(0))
```

will read this data. Note that `Fname` as used here is an **R** object whose value is the name of the file that contains the data.

Note: if the file is not located in the working directory, then full path names must be used to specify the file. **R** uses unix conventions for path names regardless of the operating system. So, for example, in Windows a file located on the C-drive in folder StatData named Data1.txt would be scanned by

```
x = scan("c:/StatData/Data1.txt")
```

The file name argument also can be a web address.

Data Frames and read.table(). Tabular data contained in a file can be read by **R** using the *read.table()* function. Each column in the table is treated as a separate variable and variables can be numeric, logical, or character (strings). That is, different columns can be different types, but each column must be the same type. An example of such a file is

`http://www.utdallas.edu/~ammann/stat3355scripts/Temperature.data`

Note that the first few lines begin with the character #. This is the comment character. **R** ignores that character and the remainder of the line. The first non-comment line contains names for the columns. In that case we must include the optional argument `header=TRUE` as follows:

```
Temp = read.table("http://www.utdallas.edu/~ammann/stat3355scripts/Temperature.data",
  header=TRUE)
```

The first column in this file is not really data, but just gives the name of each city in the data set. These can be used as row names:

```
Temp = read.table("http://www.utdallas.edu/~ammann/stat3355scripts/Temperature.data",
  header=TRUE, row.names=1)
```

The value returned by *read.table()* is a *data.frame*. This type of object can be thought of as an enhanced matrix. It has a *dimension* just like a matrix, the value of which is a vector containing the number of rows and number of columns. However, a data frame is intended to represent a data set in which each row is the set of variables obtained for each subject in the sample and each column contains the observations for each variable being measured. In the case of the Temperature data, these variables are:

JanTemp, Lat, Long

Unlike a matrix, a data frame can have different types of variables, but each variable (column) must contain the same type.

Individual variables in a data frame can be accessed several ways.

(a) Using $

```
Latitude = Temp$Lat
```

(b) Name:

```
Latitude = Temp[["Lat"]]
```

(c) Number:

```
Latitude = Temp[[2]]
```

Note that the object named `Latitude` is a vector. If you want to extract a subset of the variables with all rows included, then use []. The result is a data frame. If the original data frame has names, these are carried over to the new data frame. If you only want some of the rows, then specify these the way it is done with matrices:

```
LatLong = Temp[2:3] #extract variables 2 through 3
LatLong = Temp[c("Lat","Long")] #extract Lat and Long
LatLong1 = Temp[1:20,c("Lat","Long")] #extract first 20 rows for Lat and Long
```

Although it may seem like more work to use names, the advantage is that one does not need to know the index of the desired column, just its name.

Additional variables can be added to a data frame as follows.

```
#create new variable named Region with same length as other variables in Temp
Region = rep("NE",dim(Temp)[1])
# NE is defined to be Lat >= 39.75 and Long < 90
# SE is defined to be Lat < 39.75 and Long < 90
# SW is defined to be Lat < 39.75 and Long >= 90
# NW is defined to be Lat >= 39.75 and Long >= 90
Region[Temp$Lat < 39.75 & Temp$Long < 90] = "SE"
Region[Temp$Lat < 39.75 & Temp$Long >= 90] = "SW"
Region[Temp$Lat >= 39.75 & Temp$Long >= 90] = "NW"
#give Region the same row names as Temp
names(Region) = dimnames(Temp)[[1]]
#make Region a factor
Region = factor(Region)
#add Region to Temp
Temp1 = data.frame(Temp,Region)
#plot January Temperature vs Region
#since Region is a factor, this gives a boxplot
plot(JanTemp ~ Region,data=Temp1)
```

17. The `plot()` function is a top-level function that generates different types of plots depending on the types of its arguments. The formula interface is the recommended way to use this function, especially if the variables you wish to plot are contained within a data frame. When a `plot()` command (or any other top-level graphics function) is entered, then **R** closes any graphic device that currently is open and begins a new graphics window or file. Optional arguments include:

    a. *xlim*= A vector of length 2 that specifies x-axis limits. If not specified, then **R** computes limits from the range of the x-data.

    b. *ylim*= A vector of length 2 that specifies y-axis limits. If not specified, then **R** computes limits from the range of the y-data.

    c. *xlab*= A string that specifies a label for the x-axis, default is name of x argument.

    d. *ylab*= A string that specifies a label for the y-axis, default is name of y argument.

    e. *col*= A color name or vector of names for the colors of plot elements.

    f. *type*= A 1-character string that specifies the type of plot: *type="p"* plots points (default); *type="l"* plots lines; *type="n"* sets up plot region and adds axes but does not plot anything.

    g. *main*= Adds a main title. This also can be done separately using the `title()` function.

    h. *sub*= Adds a subtitle. This also can be done separately using the `title()` function.

18. Other functions add components to an existing graphic. These functions include

    a. `title()` Add a main title to the top of an existing graphic. Optional argument *sub*= adds a subtitle to the bottom.

    b. `points(x,y)` Add points at locations specified by the x,y coordinates. Optional arguments include *pch*= to use different plotting symbols, *col*= to use different colors for the points.

    c. `lines(x,y)` Add lines that join the points specified by x,y arguments. Optional arguments include *lty*= to use different line types, *col*= to use different colors for the points.

    d. `text(x,y,labels=)` Add strings at the locations specified by x,y arguments.

    e. `mtext()` Add text to margins of a plot.

19. **Accessing data in a spreadsheet**. If a table of data is contained in a spreadsheet like *Excel*, then the easiest way to import it into **R** is to save the table as a *comma-separated-values* file. Then use `read.table()` to read the file with separator argument `sep=","`. The file
    `http://www.utdallas.edu/~ammann/SmokeCancer.csv`
    can be read into **R** by

```
Smoke =  read.table("http://www.utdallas.edu/~ammann/SmokeCancer.csv",
  header=TRUE,sep=",",row.names=1)
```

Note that 2 of the entries in this table are `NA`. These denote types of cancer that were not reported in that state during the time period covered by the data. We can change those entries to 0 as follows.

```
Smoke[is.na(Smoke)] = 0
```

There is a companion function, `write.table()`, that can be used to write a matrix or data frame to a file that then can be imported into a spreadsheet.

20. **Saving graphics**. By default **R** uses a separate graphical window for the display of graphic commands. A graphic can be saved to a file using any of several different graphical file types. The most commonly used are *pdf()* and *png()* since these types can be imported into documents created by **Word** or LaTeX. The first argument for these functions is the filename. Arguments `width=,height=` give the dimensions of the graphic. For *pdf()* the dimension units are inches, for *png()* the units are pixels. *pdf()* supports multi-page graphics, but *png()* only allows one page per file unless the file name has the form `Myplot%d.png`. For example,

```
pdf("TempPlot.pdf",width=6,height=6)
plot(JanTemp ~ Lat,data=Temp)
plot(JanTemp ~ Region,data=Temp1)
graphics.off()
#creates a 2-page pdf document
png("TempPlot%d.png",width=480,height=480)
plot(JanTemp ~ Lat,data=Temp)
plot(JanTemp ~ Region,data=Temp1)
graphics.off()
#creates two files: TempPlot1.png and TempPlot2.png
```

The function *graphics.off()* writes any closing material required by the graphic file type and then closes the graphics file.

21. **RStudio** includes a plot tab where plots are displayed. After creating a plot, it can be exported to a graphic file that can be added to a Word document. This is done via the *Export* link on the plot tab using the *Save as image* selecion. The most widely used image file type is *png*.

There are a number of datasets included in the **R** distribution along with examples of their use in the help pages. One example is given below.

```
# load cars data frame
data(cars)
# plot braking distance vs speed with custom x-labels and y-labels,
# and axis numbers horizontal
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1)
# add plot title
title(main = "Cars data")
# new plot of same variables on a log scale for both axes
plot(cars, xlab = "Speed (mph)", ylab = "Stopping distance (ft)",
     las = 1, log = "xy")
# add plot title
title(main = "Cars data (logarithmic scales)")
# fit a regression model using log(speed) to predict log(dist) and
# print a summary of the fit
summary(fm1 = lm(log(dist) ~ log(speed), data = cars))
# save the current plotting parameters and then setup a new plot
# region that puts 4 plots on the same page, 2 rows and 2 columns.
# use custom margins for the plot region.
opar = par(mfrow = c(2, 2), oma = c(0, 0, 1.1, 0),
           mar = c(4.1, 4.1, 2.1, 1.1))
# plot the diagnostic residual plots associated with a regression fit.
plot(fm1)
# restore the original plotting parameters
par(opar)
```

Class Notes

Graphical tools

The computer tools that we have available today give us access to a wide array of graphical techniques and tools that can be used for effective presentation of complex data. However, we must first understand what type of data we wish to present, since the presentation tool that should be used for a set of data depends on the questions we wish to answer and the type of data we are using to answer those questions.

Categorical (qualitative) data

Categorical data is derived from populations that consist of some number of subpopulations and we record only the subpopulation membership of selected individuals. In such cases the basic data summary is a frequency table that counts the number of individuals within each category. If there is more than one set of categories, then we can summarize the data using a multi-dimensional frequency table. For example, here is part of a dataset that records the hair color, eye color, and sex of a group of 592 students.

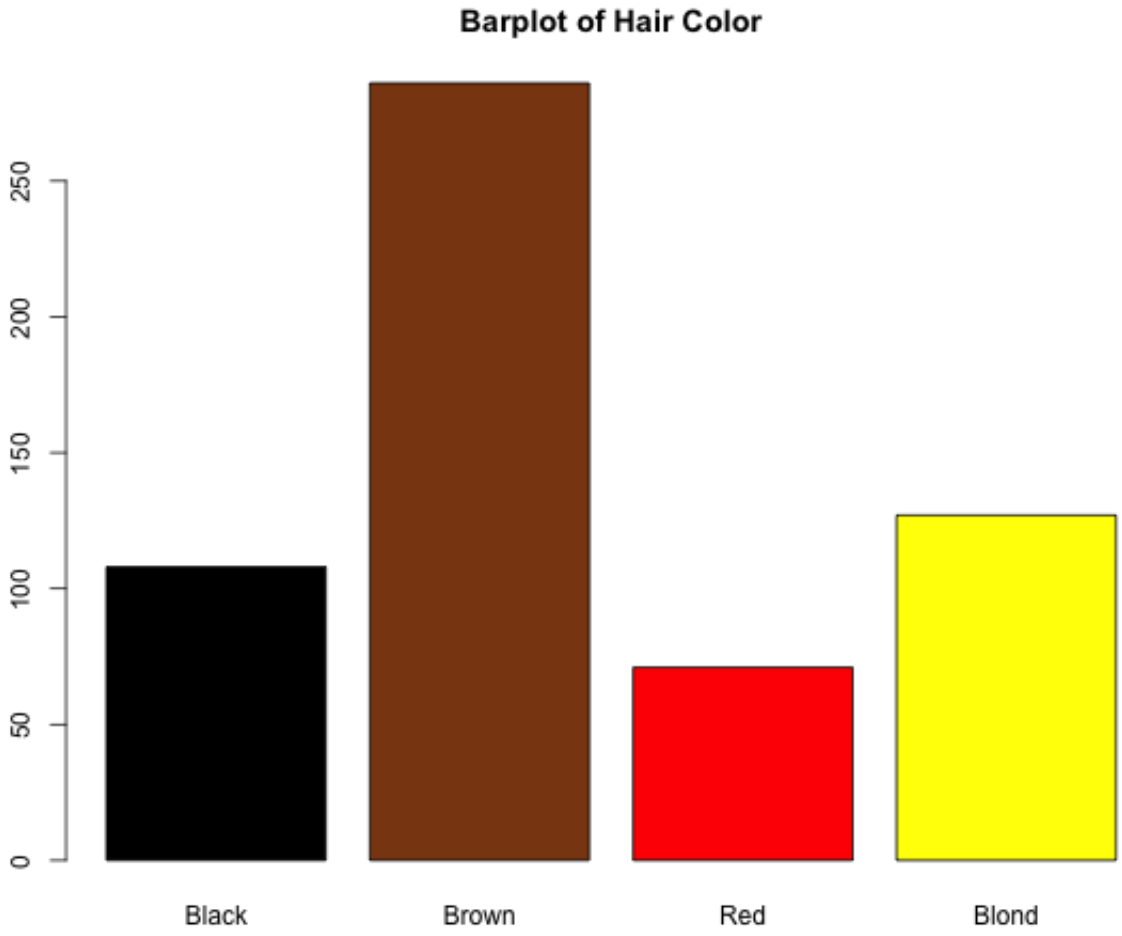
```
Hair   Eye    Sex
Black  Brown  Female
Red    Green  Male
Blond  Blue   Male
Brown  Hazel  Female
...
```

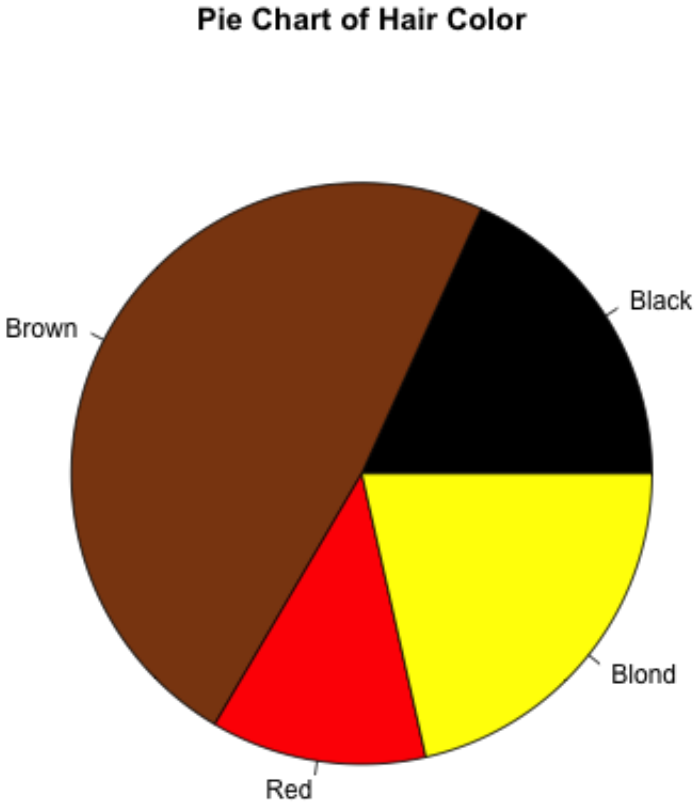
In the past numerical codes were used in place of names because of memory limitations, but in that case it is important to remember that codes are just labels. **R** does that internally by representing categorical data as a **factor**. This is a special type of vector that has an attribute names *levels* which represent the unique set of categories of the variable.

The frequency table for hair color in this dataset is:

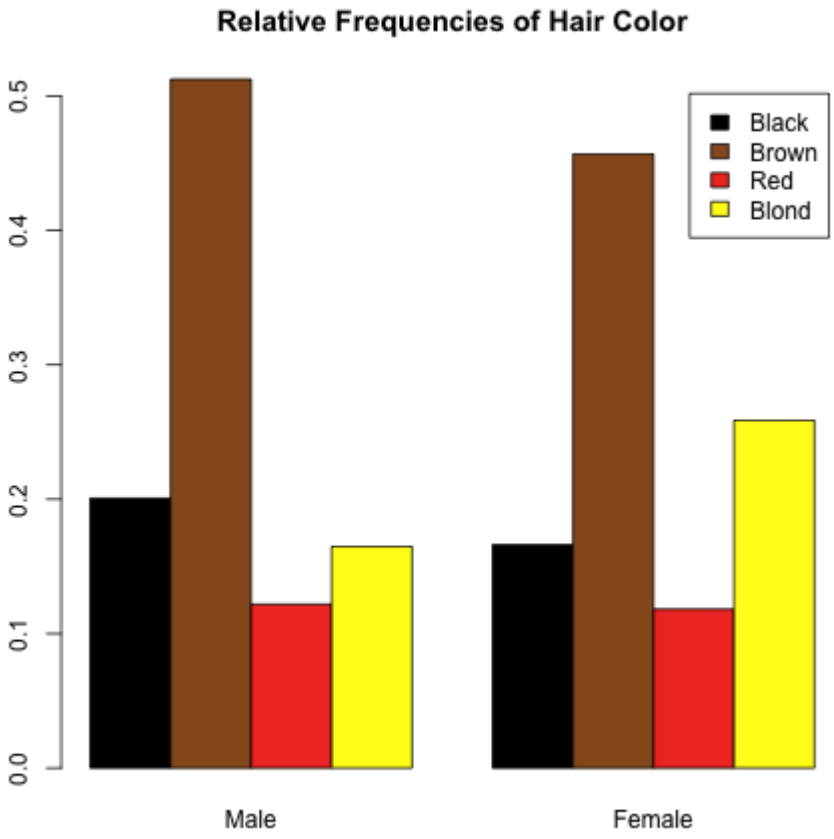
```
Black Brown   Red Blond
  108   286    71   127
```

The basic graphical tool for categorical data is the barplot. This plots bars for each category, the height of which is the frequency or relative frequency of that category. Barplots are more effective than pie charts because we can more readily make a visual comparison of heights of rectangles than angles in a pie.


Barplot of Hair Color
0
50
100
150
200
250
Black
Brown
Red
Blond

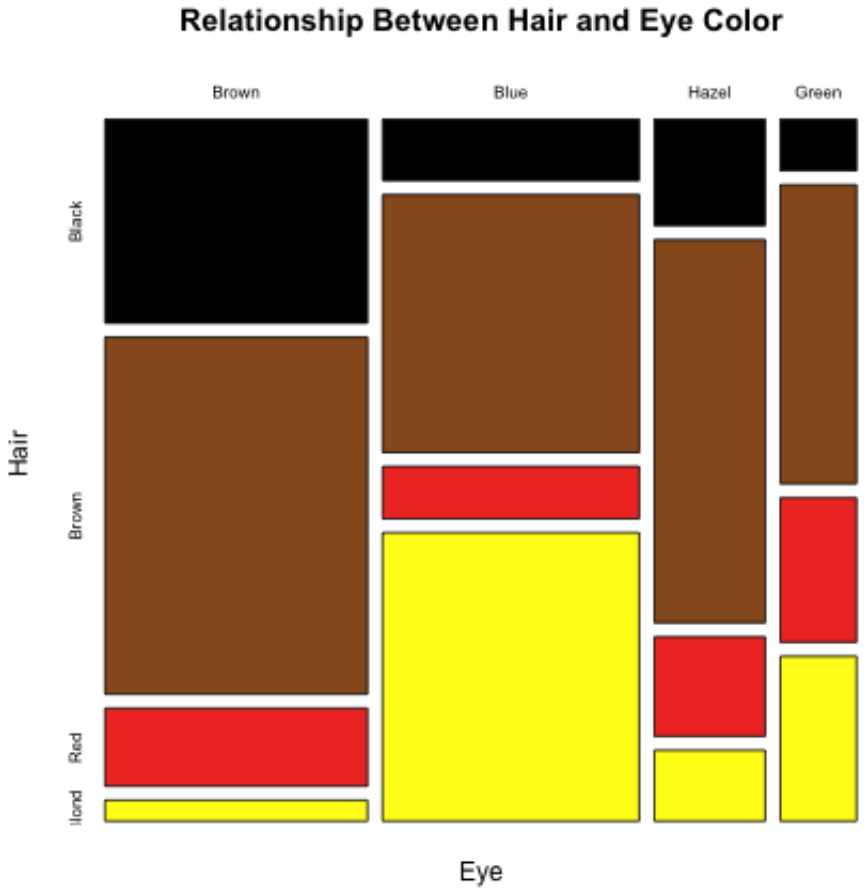

Pie Chart of Hair Color
Brown
Black
Blond
Red

If a second categorical variable also is observed, for example hair color and sex, a barplot with side-by-side bars for each level of the first variable plotted contiguously, and each such group plotted with space between groups, is most effective to compare each level of the first variable across levels of the second. For example, the following plot shows how hair color is distributed for a sample of males and females. A comparison of the relative frequencies for males and females shows that a relatively higher proportion of females have blond hair and somewhat lower proportion of females have black or brown hair.



We can also display the relationship between hair and eye color using a 2-dimensional frequency table and barplot. The areas of the rectangles in this plot represent the relative frequency of the corresponding category combination.

Hair	Eye Brown	Blue	Hazel	Green
Black	68	20	15	5
Brown	119	84	54	29
Red	26	17	14	14
Blond	7	94	10	16



Are hair color and eye color related? Although we will consider this question in detail later, we can think about how to interpret this question here. First note that a total of 108 people have black hair 68 of whom also have brown eyes. That is, 63% (68/108) of those with black hair also have brown eyes. In the language of probability this ratio is referred to

as a conditional probability and would be expressed as

$$P(\text{Brown eyes} \mid \text{Black hair}) = \frac{68}{108} = 0.63.$$

First note the correspondence between the structure of the sentence, *63% of those with black hair also have brown eyes*, and the arithmetic that goes with it. The reference group for this percentage is defined by the prepositional phrase, *of those with black hair*, and the count for this group is the denominator. The verb plus object in this sentence is *have brown eyes*. The count of people who have brown eyes within the reference group (*those with black hair*) is the numerator of this percentage. So those who are counted for the numerator must satisfy both requirements, *have brown eyes* and *have black hair*. The corresponding probability statement is

$$\begin{aligned} P(\text{Brown eyes} \mid \text{Black hair}) &= \frac{P(\{\text{Brown eyes}\} \cap \{\text{Black hair}\})}{P(\{\text{Black hair}\})} \\ &= \frac{68/592}{108/592} \\ &= \frac{68}{108} = 0.63. \end{aligned}$$

It is important to remember that the reference group for ordinary probability such as

$$P(\{\text{Black hair}\})$$

is the total group, whereas the reference group for conditional probability is the subgroup specified after the `|` symbol.

The total counts for eye color are:

```
Brown  Blue Hazel Green
  220   215    93    64
```

so 220 of the 592 people in this data have brown eyes. That is, 220/592 = 37% of all people in this data set have brown eyes, but brown eyes occur much more frequently among people with black hair, 63%. The corresponding probabiity statements are

$$\begin{aligned} P(\{\text{Brown eyes}\}) &= \frac{220}{592} = 0.37 \\ P(\text{Brown eyes} \mid \text{Black hair}) &= \frac{68/592}{108/592} = 0.63 \end{aligned}$$

This shows that the percentage of people who have brown eyes depends on whether or not they have black hair. If the two percentages had been equal, that is, if 37% of people with black hair also had brown eyes, then we would say that having brown eyes does not depend on whether or not a person has black hair since those percentages would have been the same. Therefore, for those two outcomes to be independent, there should have been 40 people (37% of 108) with black hair and brown eyes. This is the expected count under the assumption of independence between brown eyes and black hair. We can do the same for each combination of categories in this table to give the expected frequencies:

```
       Brown   Blue Hazel Green
Black  40.14  39.22 16.97 11.68
Brown 106.28 103.87 44.93 30.92
Red    26.39  25.79 11.15  7.68
Blond  47.20  46.12 19.95 13.73
```

If all of the observed counts had been equal to these expected counts, then hair and eye color would be completely independent. Obviously that is not the case. We can define a measure of distance between the observed counts and the expected counts under the assumption of independence by

$$D = \sum \frac{(O-E)^2}{E},$$

where the sum is over all combinations of hair and eye categories. Note that the expected count for a cell can be expressed as

$$E = \frac{R * C}{N},$$

where R denotes the row total, C denotes the column total, and N denotes the grand total. For this data, $D = 138.3$. Later in the course we will discuss how to interpret this distance statistically and determine whether or not it is large. The contribution to this distance from each cell is:

```
       Eye
Hair    Brown  Blue Hazel Green
  Black 19.35  9.42  0.23  3.82
  Brown  1.52  3.80  1.83  0.12
  Red    0.01  2.99  0.73  5.21
  Blond 34.23 49.70  4.96  0.38
```

Note that blond hair with brown or blue eyes are the greatest contributors to the distance from independence of these counts.

```
data(HairEyeColor) #load HairEyeColor data set
HairEyeColor #this is a 3-d array
HairEye = apply(HairEyeColor,c(1,2),sum) #sum over sex, keep dimensions 1,2
HairEye
Hair = apply(HairEye,1,sum) #get totals for hair color
Eye = apply(HairEye,2,sum) #get totals for eye color
Gender = apply(HairEyeColor,3,sum) #get totals for sex
# graphics
Hair.color = c("black","saddlebrown","red","yellow")
Eye.color = c("saddlebrown","blue","yellow4","green")
barplot(Hair,col=Hair.color)
```

```
title("Barplot of Hair Color")
#barplot is better than pie chart
par(mfrow=c(2,1))
barplot(Hair,col=Hair.color)
title("Barplot of Hair Color")
pie(Hair,col=Hair.color)
title("Pie Chart of Hair Color")
par(mfrow=c(1,1))
#compare males and females
HairGender = margin.table(HairEyeColor, c(1, 3))
print(HairGender)
barplot(HairGender,col=Hair.color,main="Hair Color")
barplot(HairGender,col=Hair.color,legend.text=TRUE,xlim=c(0,3),main="Hair Color")
#relative frequency
HairGenderP = scale(HairGender,scale=Gender,center=FALSE)
print(HairGenderP)
barplot(HairGenderP,col=Hair.color,legend.text=TRUE,xlim=c(0,3),main="Relative Frequenci
barplot(HairGenderP,beside=TRUE,col=Hair.color,legend.text=TRUE,main="Relative Frequenci
# find distances from independence
# there are several ways to compute R*C. The easiest way is to use the
# function outer() which is a generalized outer product
# this function takes two vectors as arguments and generates a matrix
# with number of rows = length of first argument and
# number of columns = length of second argument.
# Elements of the matrix are obtained by multiplying each element of the first
# vector by each element of the second vector.
N = sum(HairEyeColor)
ExpHairEye = outer(Hair,Eye)/N
round(ExpHairEye,2) #note that outer preserves names of Hair and Eye
# now get distance from independence
D = ((HairEye - ExpHairEye)^2)/ExpHairEye
round(D,2) # gives contribution from each cell
sum(D) # print total distance
# now use R function paste to combine text and value
paste("Distance of Hair-Eye data from Independence =",round(sum(D),2))
# if round is not used then lots of decimal places will be printed!
paste("Distance of Hair-Eye data from Independence =",sum(D))
```

We will see later that this data is very far from independence!

R has several ways to save the graphics into files so they can be added to a document. After a graphic is created in **Rstudio**, use the `Export` menu to interactively save the graphic as an image file. The default file type is `PNG` which is the recommended image format to use. Be sure to change the name of the image file from the default name `Rplot.png`. Another

way is to use the graphics function `png()` to specify the file name along with options that specify the size in pixels of image. After all comnmands for a particular graphic have been entered, finish the graphic by entering

```
graphics.off()
```

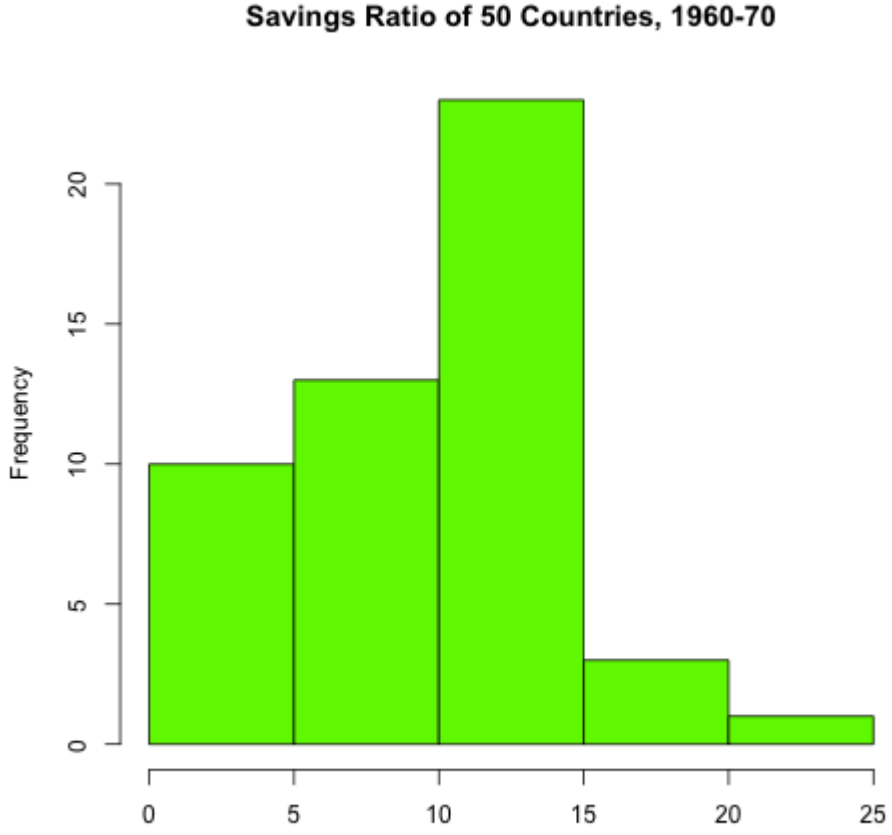
Try to use informative file names for saved graphics. The folowing script creates text output and image files for the hair-eye color example. These can be imported into a document processor such as Word.
`http://www.utdallas.edu/~ammann/stat3355scripts/HairEye1.r`

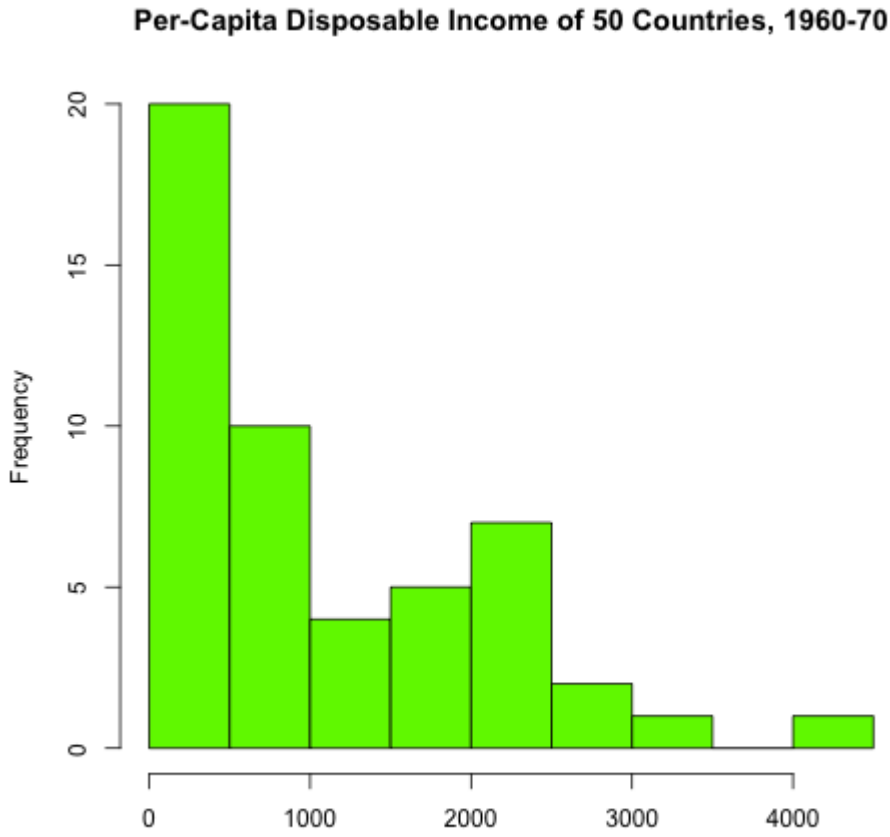
Quantitative data

Data in which the values represent some numerical quantity are referred to as quantitative data. For example, here is a portion of a dataset that contains savings rates along with other demographic variables for 50 countries during 1960-70.

```
                sr pop15 pop75     dpi  ddpi
Australia    11.43 29.35  2.87 2329.68  2.87
Austria      12.07 23.32  4.41 1507.99  3.93
Belgium      13.17 23.80  4.43 2108.47  3.82
...
```

In this dataset *sr* represents savings ratio, *pop15* represents the percent of population under age 15, *pop75* is the percent of population over age 75, *dpi* is the real per-capita disposable income, and *ddpi* is the percent growth rate of *dpi*. The most commonly used graphical method for summarizing quantitative data is the **histogram**. To construct a histogram, we first partition the data values into a set of non-overlapping intervals and then obtain a frequency table. A histogram is the barplot of the corresponding frequency data but with contiguous bars. Here are histograms for savings ratio and disposable income.

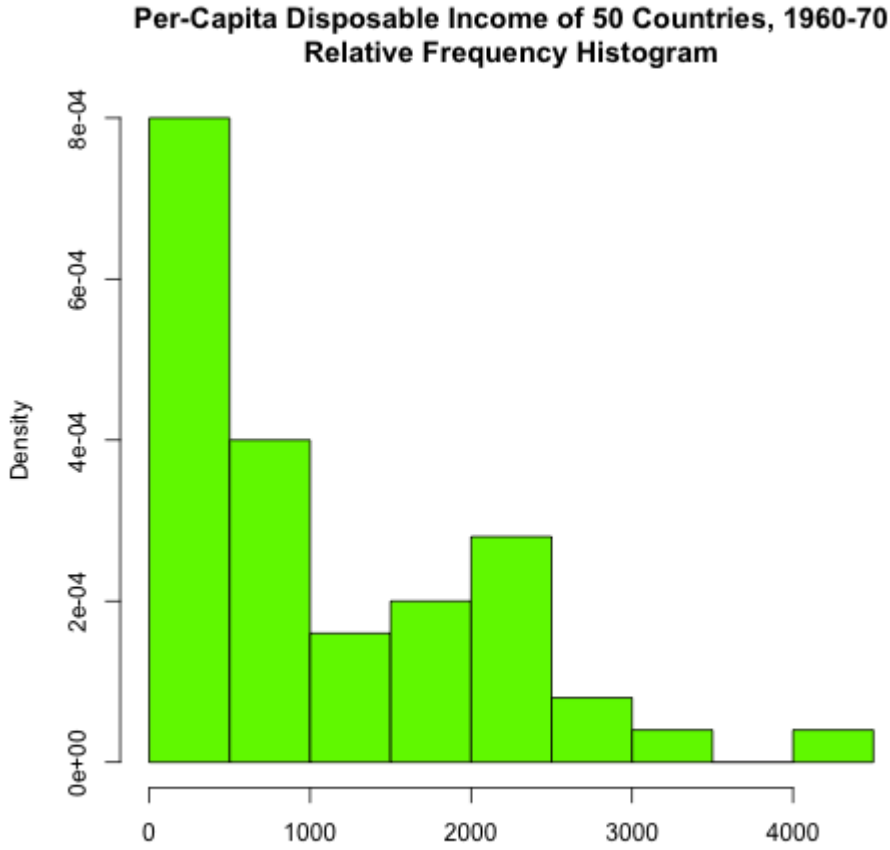



Per-Capita Disposable Income of 50 Countries, 1960-70
Frequency
0
5
10
15
20
0
1000
2000
3000
4000

These histograms were generated by:

```
data(LifeCycleSavings)
png("LifeCycleSavings1.png",width=600,height=500, bg="transparent")
hist(LifeCycleSavings$sr,xlab="",main="Savings Ratio of 50 Countries, 1960-70",col="gree
#
png("LifeCycleSavings2.png",width=600,height=500, bg="transparent")
hist(LifeCycleSavings$dpi,xlab="",main="Per-Capita Disposable Income of 50 Countries, 19
graphics.off()
```

In some applications, proportions within the sub-intervals are of greater interest than the frequencies. In such cases a relative frequency histogram can be used instead. In this case the y-axis is re-scaled by dividing the frequencies by the total number of observations. The shape of a relative frequency histogram is unchanged; the only quantity that changes is the scale of the y-axis. **R** can generate probability histograms in which the y-axis is scaled to make total area of the histogram equal to 1. Changing the scale of the y-axis to represent proportions takes a little extra work.

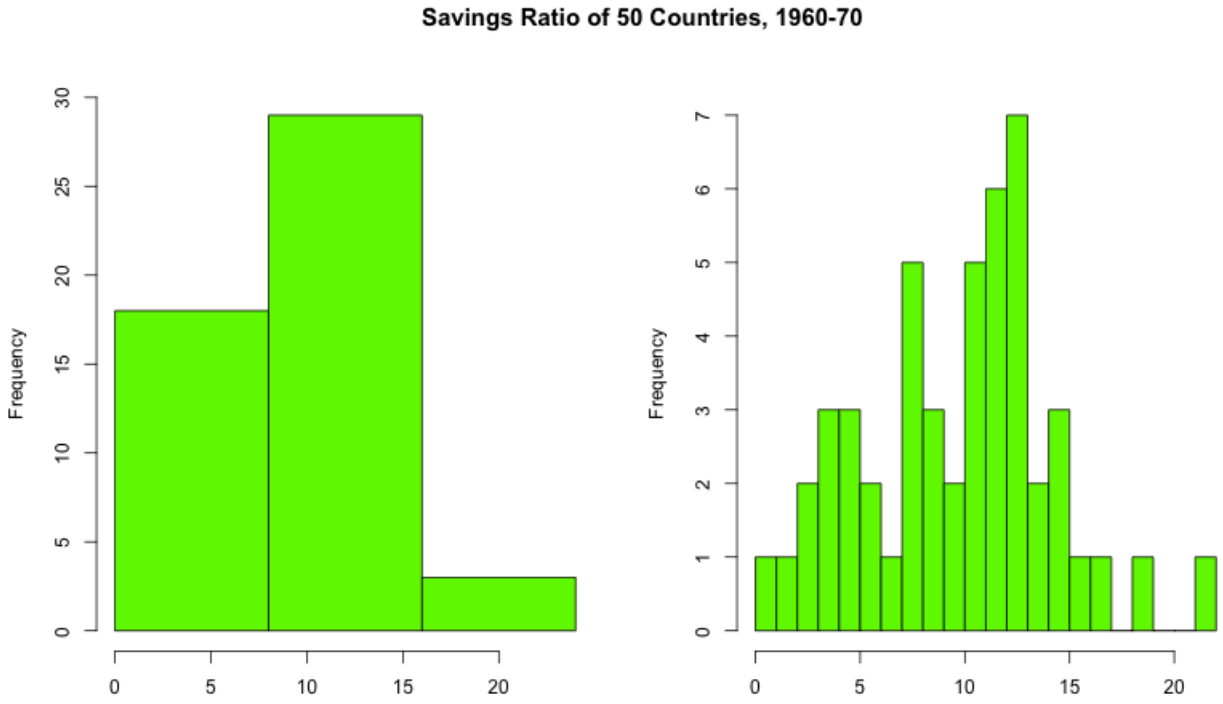


These histograms were generated by the following code:

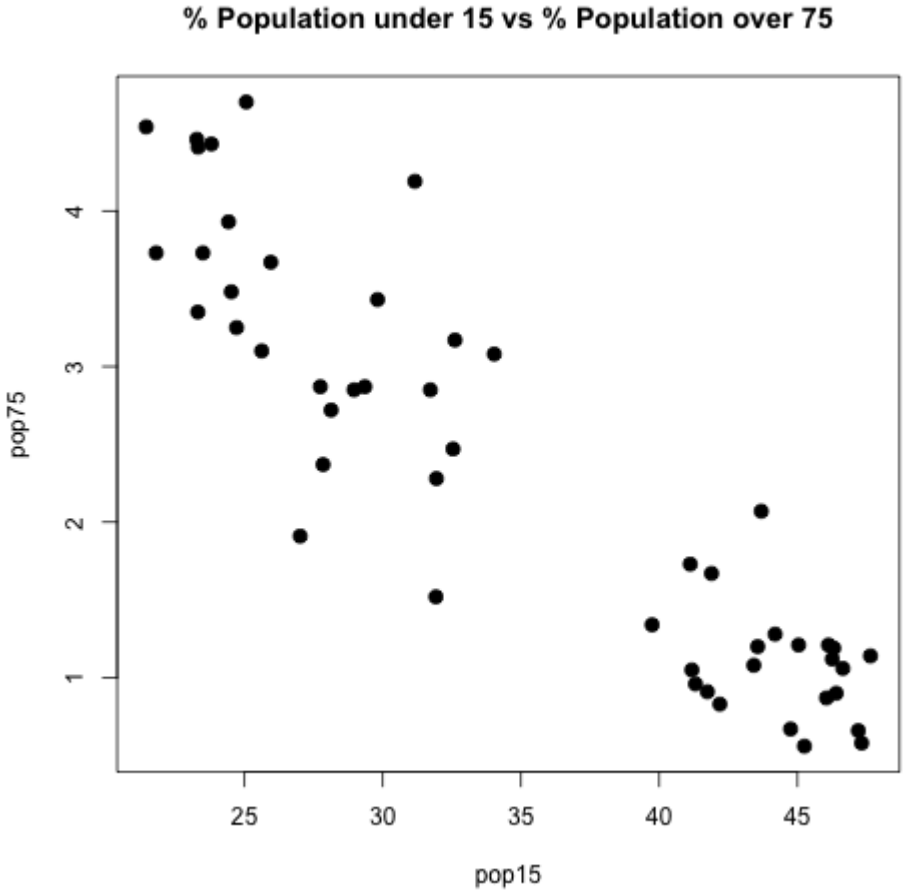
```
data(LifeCycleSavings)
png("LifeCycleSavings2a.png",width=650,height=500, bg="transparent")
hist(LifeCycleSavings$dpi,xlab="",main="",freq=F,col="green") # don't use default title
mtext("Probability Histogram of Per-Capita Disposable Income\n50 Countries, 1960-70",
    outer=T,line=-3,cex=1.25,font=2)
### advanced: change y-axis tick marks to represent relative frequencies
# don't use default title
# don't use default y-axis tick marks
# capture output of hist()
png("LifeCycleSavings2b.png",width=650,height=500, bg="transparent")
savhist = hist(LifeCycleSavings$dpi,xlab="",ylab="Proportion",main="",yaxt="n",col="gree
mtext("Relative Frequency Histogram of Per-Capita Disposable Income\n50 Countries, 1960-
    outer=T,line=-3,cex=1.25,font=2)
ycnt = savhist$counts # heights of histogram bars
n = sum(ycnt) # number of observations
yrelf = pretty(range(ycnt/n)) # obtain new labels for tick marks
# y-axis scale in hist represents counts
# locations of new tick labels need to correspond to counts so they are located at yrelf
axis(side=2,at=yrelf*n,labels=yrelf) # put new labels where marks f
graphics.off()
```

There is no fixed number of sub-intervals that should be used. A large number of sub-intervals corresponds to less summarization of the data, and a small number of sub-intervals corresponds to more summarization.

When two or more variables are measured for each individual in the dataset, then we may be interested in the relationship between these variables. The type of graphical display we use depends on the types of the variables. We have already seen an example of a 2-dimensional barplot for the case in which both variables are categorical. If both variables are quantitative, then the basic graphical tool is the **scatterplot**. For example, here is a scatterplot of *pop15* versus *pop75*.


Savings Ratio of 50 Countries, 1960-70
Frequency
0
5
10
15
20
25
30
Frequency
0
1
2
3
4
5
6
7
0
5
10
15
20

The relationships among all 5 of the variables in this dataset can be displayed simultaneously by constructing pairwise scatterplots on the same graphic.



Note: we will defer until later in the course a discussion of numerical descriptions of these relationships.

R functions

See help pages for detailed descriptions of the functions used in this section.
`barplot()`: construct bar plots for categorical variables.
`pie()`: construct pie charts for categorical variables.
`title()`: add titles to an existing plot.
`par()`: set graphical parameters.
`scale()`: center and scale each column of a numeric matrix.
`margin.table()`: obtain margin totals for an array of frequency counts.
`mosaicplot()`: mosaic plot for 2-d frequency table.
`assocplot()`: plot deviations from independence for 2-d frequency table.
`hist()`: histogram for continuous variables.
`pairs()`: plot on one page all pairwise scatter plots of multivariate data matrix.
`mtext()`: add text to margins of an existing plot.
`plot()`: generic function for plotting. The type of plot produced depends on the type of data specified by its arguments. `names()`: returns or sets the names of a vector or data frame. The names of a data frame correspond to the column names of the matrix.

Examples

Some of the functions used in this section are described below.

read.table(). If the data set for a project is not small, it is most convenient to enter the data into **R** from a tabular data file in which each row corresponds to an individual and columns contain various measurements associated with each individual. These files must be plain text (not created by a document processor such as **Word**). If the data comes from a database or spreadsheet, the simplest way to have **R** read the data is to have the database or spreadsheet export the data into a comma-separated values file (*csv*). An example is given by the file
`http://www.utdallas.edu/~ammann/stat3355scripts/crabs.csv`

a. The first argument is the name of the data file. This must be a string that contains the full path to the file if it is not in the startup directory, or it may be an internet address if the file is on a remote server.

b. The first row of the *crabs.csv* file contains names for the columns. This row is referred to as a header and requires use of the

   `header=TRUE`

   argument.

c. The values in each row are separated by a comma. The default separator is white space, so the argument

   ```
   sep=","
   ```

   is needed for the crabs data file. The following **R** code performs this task.

   ```
   Crabs = read.table("http://www.utdallas.edu/~ammann/stat3355scripts/crabs.csv",head
   ```

Note: `read.table()` will return an error message if it finds that the rows don't all contain the same number of values. This can occur, for example, if a csv file was created from an *Excel* file that had some extraneous blank cells. Otherwise, *read.table()* returns a data frame that is assigned to the name *Crabs*.

Note that the first two columns, named *Species* and *Sex*, respectively, contain strings, not numeric values. In such cases, **read.table()** assumes these are categorical variables and then converts each of them automatically to a **factor**. The unique values of a factor are referred to as its *levels*. The levels of *Species* are B,O (for blue and orange), and the levels of *Sex* are M,F.

A particular column of a data frame can be accessed by name of the data frame followed by a dollar sign followed by the name of the column. So, for example,

```
Crabs$FL
```

refers to the column named `FL` within the `Crabs` data frame. We can obtain a histogram of that column by

```
hist(Crabs$FL)
```

Smoke-Cancer data

The file
`http://www.utdallas.edu/~ammann/SmokeCancer.csv`
contains data related to smoking and cancer rates by state for 2010.

1. Import this data into **R** using the first column as row names. This requires adding the argument,

   ```
   row.names=1
   ```

   within **read.table()**.

2. Create a new data frame that contains the following variables:
   CigSalesRate = FY2010 Sales per 100,000 population,
   CigYouthRate, CigAdultRate, LungCancerRate, EsophCancerRate

3. Create all pairwise plots of the variables in this data frame. Add an informative main title and note on the plot that the data includes all states and D.C. Used filled circles for the plot character.

4. Create a new plot of LungCancerRate vs CigSalesRate with informative title. Note on the plot that CigSalesRate is cigarette sales per 100,000 population.

5. Repeat this plot but now use red for Texas, black for others, and add the text `TX` next to the point corresponding to Texas in this plot.

6. Repeat previous plot but use CigYouthRate instead of CigSalesRate.

7. Repeat but use CigAdultRate instead of CigYouthRate.

8. Once you are happy with how these plots look, save them in a pdf document.

Crabs data

Some of the graphical tools available in **R** are illustrated in the script file
`http://www.utdallas.edu/~ammann/stat3355scripts/crabsGraph.r`

Example script

`http://www.utdallas.edu/~ammann/stat3355scripts/CarsExample.r`

Numerical summaries of data

Although graphical techniques are useful visualization tools, they are not very good for making decisions or inferences based on data. For those situations we need to consider numerical measures. Numerical measures describe various attributes of a dataset, the most common of which are measures of location and measures of dispersion.

Note: graphics for this section are generated by the script file
`http://www.utdallas.edu/~ammann/stat3355scripts/NumericGraphics.r`

Measures of Location

We used a histogram to describe the distribution of savings rate and per capita disposable income. Now suppose instead we would like to know where the middle of the savings rate and disposable income is located. This requires that we first define what we mean by the **middle** of a dataset. There are three such measures in common use: the **mean**, **median**, and **mode**.

The **mean** usually refers to the arithmetic mean or average. This is just the sum of the measurements divided by the number of measurements. We make a notational distinction between the mean of a population and the mean of a sample. The general rule is that

greek letters are used for population characteristics and latin letters are used for sample characteristics. Therefore,

$$\mu = \frac{1}{N}\sum_{i=1}^{N} X_i,$$

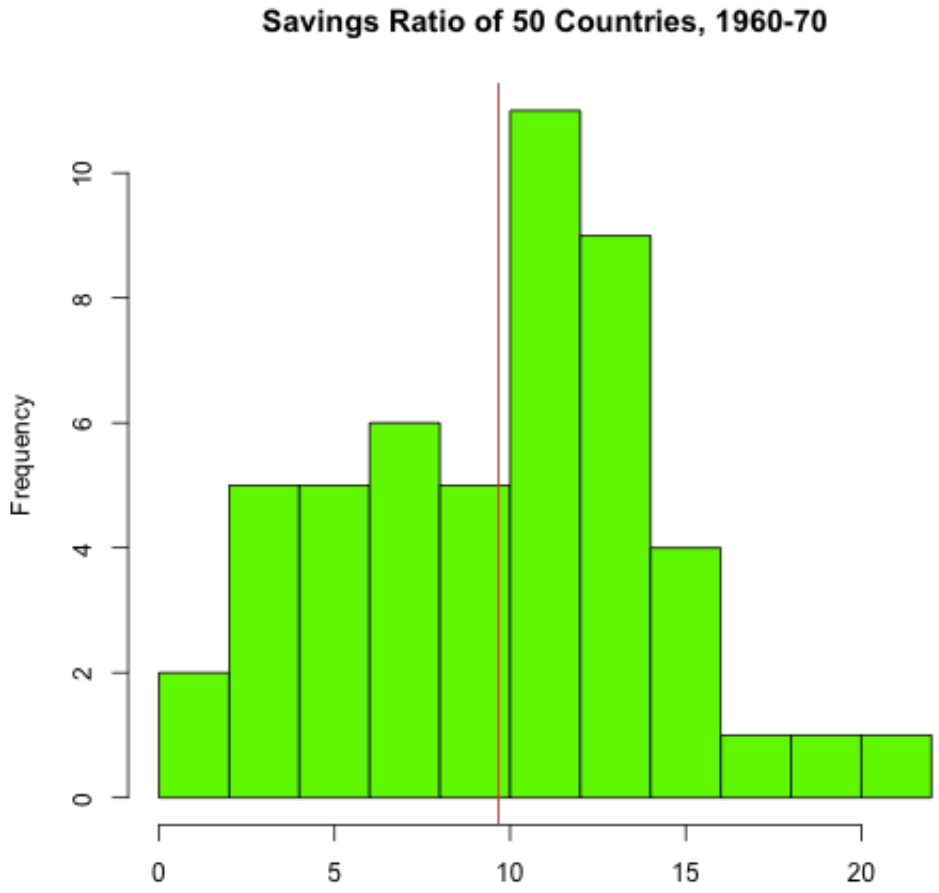
denotes the (arithmetic) mean of a population of N observations, and

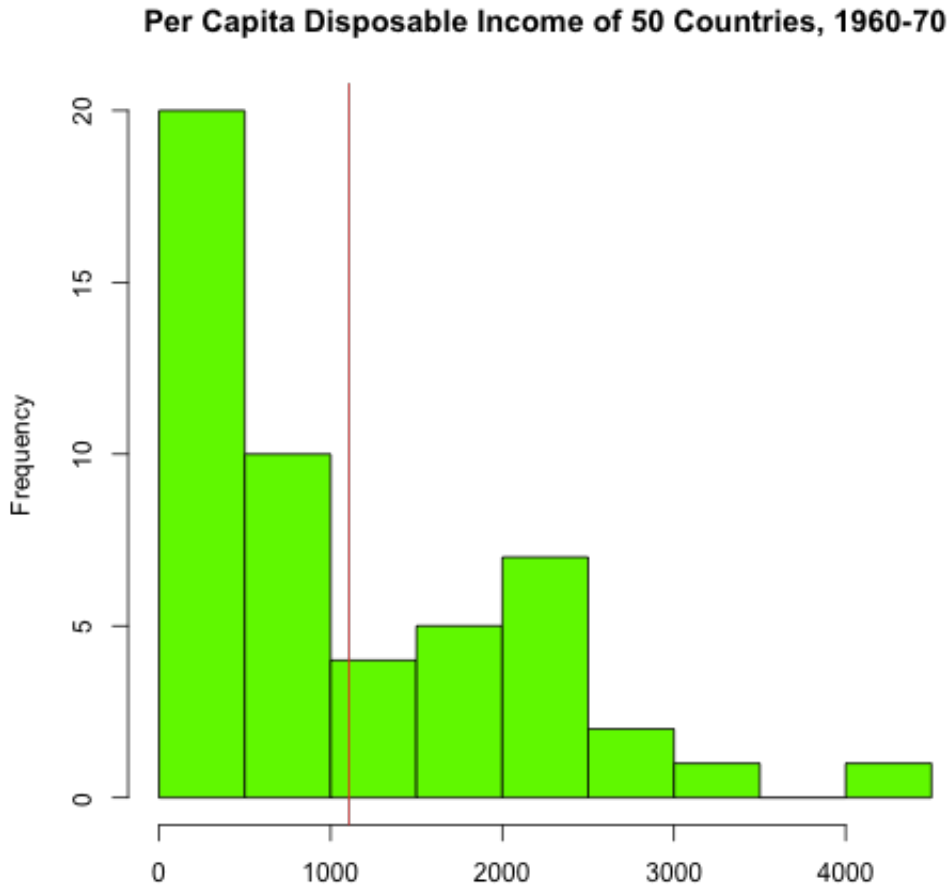
$$\overline{X} = \frac{1}{n}\sum_{i=1}^{n} X_i,$$

denotes the mean of a sample of size n selected from a population. The mean can be thought of as a center of gravity of the data values. That is, the histogram of the data would balance at the location defined by the mean. We can express this property mathematically by noting that the mean is the solution to the equation,

$$\sum_{i=1}^{n}(X_i - c) = 0.$$

This property of the mean has advantages and disadvantages. The mean is a natural measure of location for data that have a well-defined middle of high concentration with the frequency decreasing more or less evenly as we move away from the middle in either direction. The mean is not as useful when the data is heavily skewed. This is illustrated in the following two histograms. The first is the histogram of savings ratio with its mean superimposed, and the second is the histogram of disposable income.


Savings Ratio of 50 Countries, 1960-70
Frequency
0
2
4
6
8
10
0
5
10
15
20



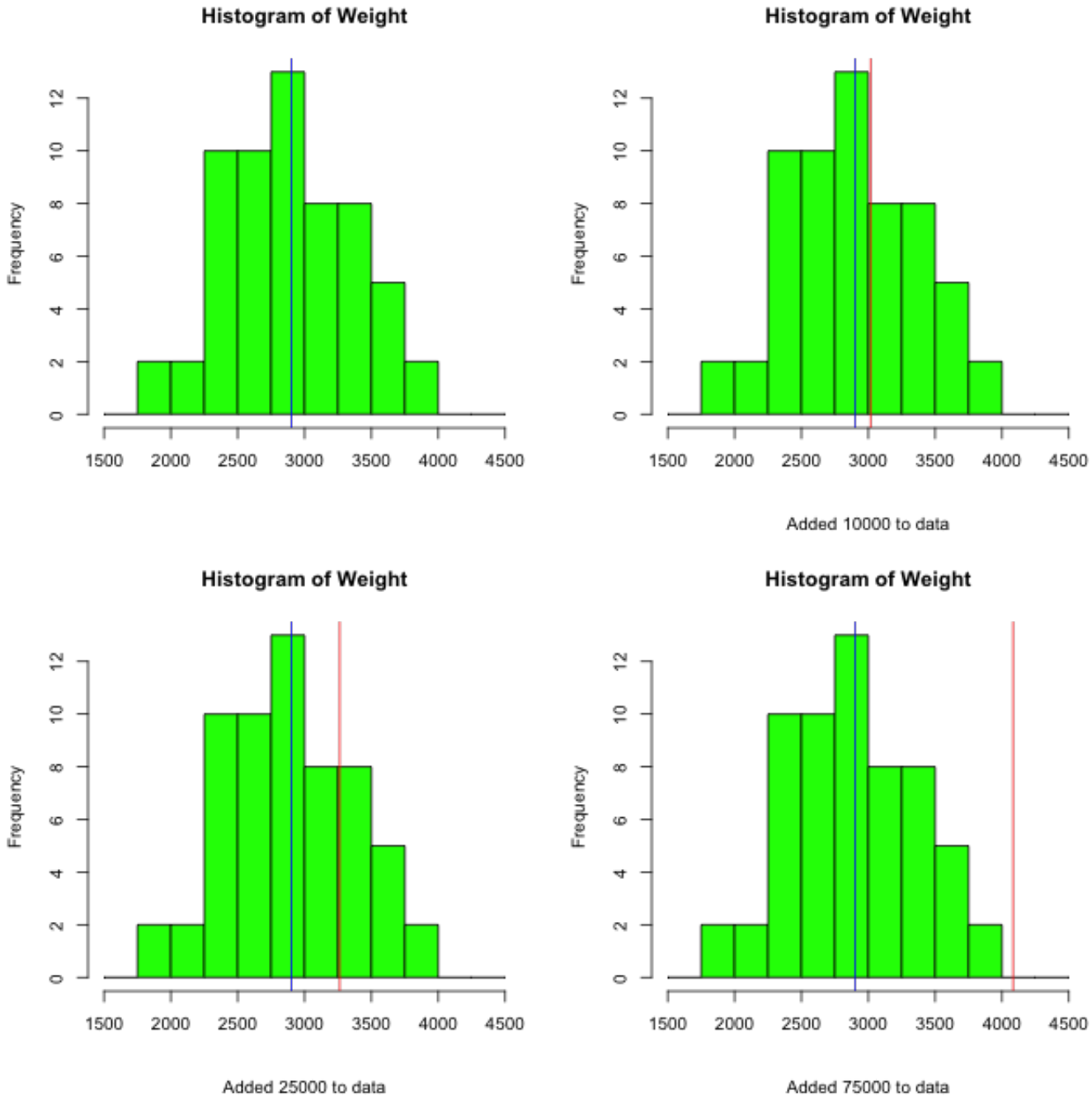
Another disadvantage of this measure is that it is very sensitive to the presence of a relatively few extreme observations. For example, the following data gives some quantities associated with 60 automobiles.

	Weight	Disp.	Mileage	Fuel	Type
Eagle Summit 4	2560	97	33	3.030303	Small
Ford Escort 4	2345	114	33	3.030303	Small
Ford Festiva 4	1845	81	37	2.702703	Small
Honda Civic 4	2260	91	32	3.125000	Small
Mazda Protege 4	2440	113	32	3.125000	Small
Mercury Tracer 4	2285	97	26	3.846154	Small
Nissan Sentra 4	2275	97	33	3.030303	Small
Pontiac LeMans 4	2350	98	28	3.571429	Small
Subaru Loyale 4	2295	109	25	4.000000	Small
Subaru Justy 3	1900	73	34	2.941176	Small
Toyota Corolla 4	2390	97	29	3.448276	Small

Toyota Tercel 4	2075	89	35	2.857143	Small
Volkswagen Jetta 4	2330	109	26	3.846154	Small
Chevrolet Camaro V8	3320	305	20	5.000000	Sporty
Dodge Daytona	2885	153	27	3.703704	Sporty
Ford Mustang V8	3310	302	19	5.263158	Sporty
Ford Probe	2695	133	30	3.333333	Sporty
Honda Civic CRX Si 4	2170	97	33	3.030303	Sporty
Honda Prelude Si 4WS 4	2710	125	27	3.703704	Sporty
Nissan 240SX 4	2775	146	24	4.166667	Sporty
Plymouth Laser	2840	107	26	3.846154	Sporty
Subaru XT 4	2485	109	28	3.571429	Sporty
Audi 80 4	2670	121	27	3.703704	Compact
Buick Skylark 4	2640	151	23	4.347826	Compact
Chevrolet Beretta 4	2655	133	26	3.846154	Compact
Chrysler Le Baron V6	3065	181	25	4.000000	Compact
Ford Tempo 4	2750	141	24	4.166667	Compact
Honda Accord 4	2920	132	26	3.846154	Compact
Mazda 626 4	2780	133	24	4.166667	Compact
Mitsubishi Galant 4	2745	122	25	4.000000	Compact
Mitsubishi Sigma V6	3110	181	21	4.761905	Compact
Nissan Stanza 4	2920	146	21	4.761905	Compact
Oldsmobile Calais 4	2645	151	23	4.347826	Compact
Peugeot 405 4	2575	116	24	4.166667	Compact
Subaru Legacy 4	2935	135	23	4.347826	Compact
Toyota Camry 4	2920	122	27	3.703704	Compact
Volvo 240 4	2985	141	23	4.347826	Compact
Acura Legend V6	3265	163	20	5.000000	Medium
Buick Century 4	2880	151	21	4.761905	Medium
Chrysler Le Baron Coupe	2975	153	22	4.545455	Medium
Chrysler New Yorker V6	3450	202	22	4.545455	Medium
Eagle Premier V6	3145	180	22	4.545455	Medium
Ford Taurus V6	3190	182	22	4.545455	Medium
Ford Thunderbird V6	3610	232	23	4.347826	Medium
Hyundai Sonata 4	2885	143	23	4.347826	Medium
Mazda 929 V6	3480	180	21	4.761905	Medium
Nissan Maxima V6	3200	180	22	4.545455	Medium
Oldsmobile Cutlass Ciera 4	2765	151	21	4.761905	Medium
Oldsmobile Cutlass Supreme V6	3220	189	21	4.761905	Medium
Toyota Cressida 6	3480	180	23	4.347826	Medium
Buick Le Sabre V6	3325	231	23	4.347826	Large
Chevrolet Caprice V8	3855	305	18	5.555556	Large
Ford LTD Crown Victoria V8	3850	302	20	5.000000	Large
Chevrolet Lumina APV V6	3195	151	18	5.555556	Van
Dodge Grand Caravan V6	3735	202	18	5.555556	Van
Ford Aerostar V6	3665	182	18	5.555556	Van

Mazda MPV V6	3735	181	19	5.263158	Van
Mitsubishi Wagon 4	3415	143	20	5.000000	Van
Nissan Axxess 4	3185	146	20	5.000000	Van
Nissan Van 4	3690	146	19	5.263158	Van

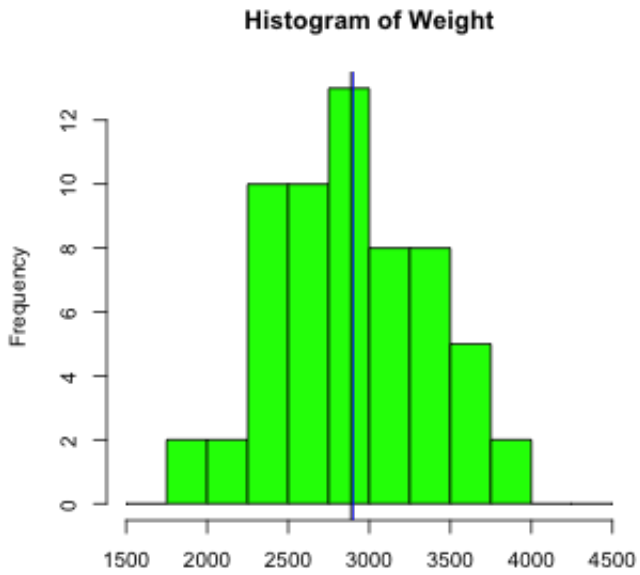
The 4 plots given below represent histograms of **Weight** with the mean of Weight superimposed. The second, third, and fourth plots are histograms of Weight with the values 10000, 25000, and 70000, respectively, added to the dataset. The *blue* line is the original mean and the red lines are the means of the modified data.

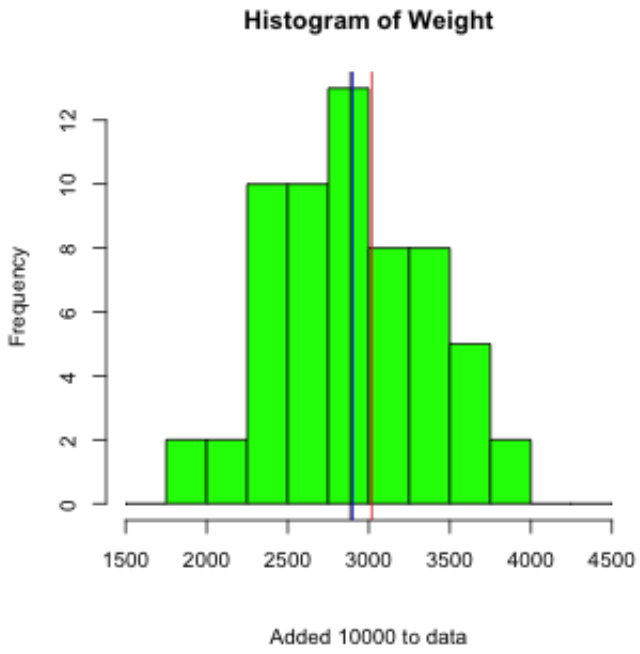


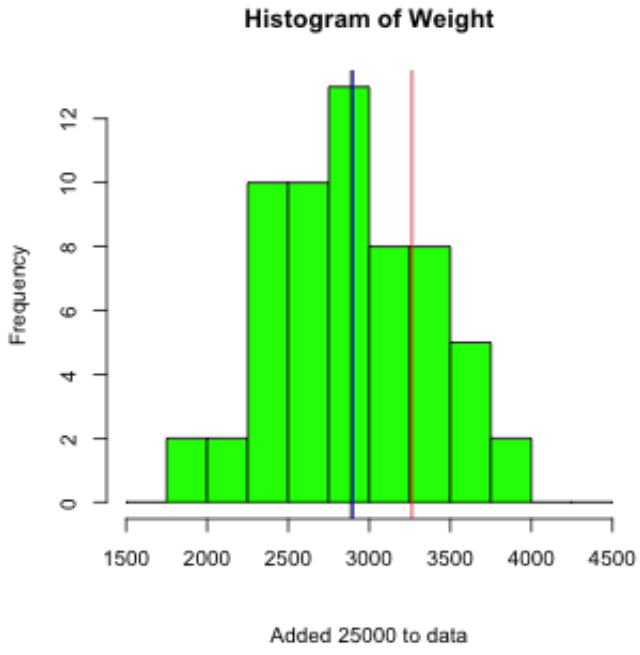
An alternative measure of location is the **median**. This measure is defined to be a number such that half of the measurements are below this number and half are above. The advantage of this measure is that it is not sensitive to the presence of a few outliers. Also, it gives an intuitive description of location regardless of the shape of the histogram. The median is obtained by first ordering the data values from smallest to largest. it the number

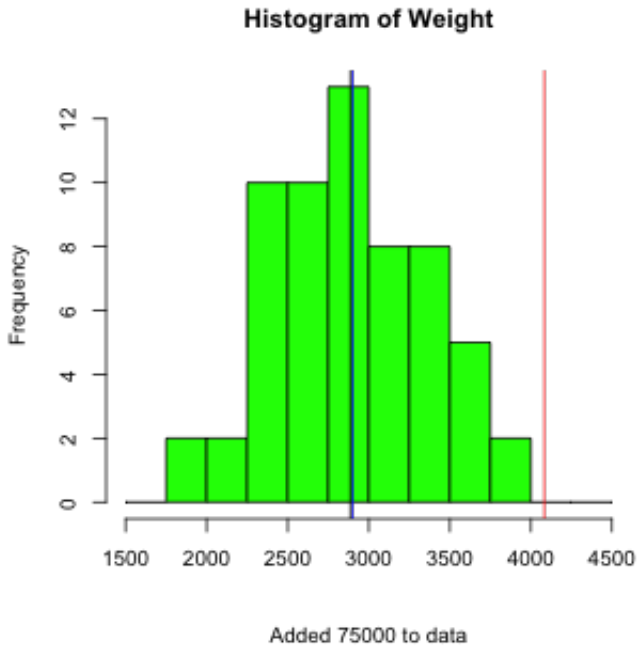
of observations n is odd, then the median is the ordered value in position *(n+1)/2*. If n is even, then the median is half-way between the *n/2* and *n/2 + 1* ordered values.

The plots below are identical to the previous plots except that the median is superimposed in black on each histogram. Note that the location of the median is much more stable than the mean. For that reason the median is used to describe the middle of data such as real estate prices and wages.









The **mode** is simply the most frequently occurring measurement or category. It is not used much except for some very specialized applications.

R notes:

There is a dataset named *state.x77* in **R** that is a matrix with 50 rows and 8 columns. We can obtain the means for each column using the function *colMeans*:

```
state.means = colMeans(state.x77)
```

This function is a shortcut for:

```
state.means = apply(state.x77,2,mean)
```

There also is a vector named *state.region* giving the geographic region (Northeast, South, North Central, West) for each state. We can use this to extract data for states belonging to a particular region as follows.

```
NorthEast.x77 = state.x77[state.region == "Northeast",]
South.x77 = state.x77[state.region == "South",]
NorthCentral.x77 = state.x77[state.region == "North Central",]
West.x77 = state.x77[state.region == "West",]
```

Suppose we wanted to build a matrix that contains the means for each variable within each region so that rows correspond to region and columns correspond to variables. We could accomplish that as follows.

```
#construct blank matrix with dimnames
Region.means = matrix(0,4,dim(state.x77)[2],
              dimnames=list(levels(state.region),dimnames(state.x77)[[2]]))
Region.means["Northeast",] = colMeans(NorthEast.x77)
Region.means["South",] = colMeans(South.x77)
Region.means["North Central",] = colMeans(NorthCentral.x77)
Region.means["West",] = colMeans(West.x77)
Region.means
round(Region.means,2)
```

Now suppose we wanted to categorize states by region and by whether or not they are above average in Illiteracy.

```
table(state.region,state.x77[,"Illiteracy"] > state.means["Illiteracy"])
```

We can make this frequency table look better by giving more informative names to the Illiteracy columns.

```
Region = state.region #give state.region a better name
# create logical vector that indicates above average or not
Illiteracy = state.x77[,"Illiteracy"] > state.means["Illiteracy"]
# assign the name Region.Illiteracy to freq table
Region.Illiteracy = table(Region,Illiteracy)
# change col names of this table
dimnames(Region.Illiteracy)[[2]] = c("Below Average","Above Average")
Region.Illiteracy
```

Another way to do this that gives access so R's object-oriented behavior is to convert the Illiteracy vector to a factor.

```
Region = state.region #give state.region a better name
# create logical factor that indicates above average or not
Illiteracy = factor(state.x77[,"Illiteracy"] > state.means["Illiteracy"])
# factor function automatically orders the levels alphabetically, so in this case
# levels are FALSE, TRUE
levels(Illiteracy)
# assign new names for these levels
levels(Illiteracy) = c("Below Average","Above Average")
# assign the name Region.Illiteracy to freq table
Region.Illiteracy = table(Region,Illiteracy)
# now we don't need to change col names of this table
Region.Illiteracy
# Plot income vs Illiteracy as a factor instead of a numeric variable
plot(state.x77[,"Income"] ~ Illiteracy,ylab="Income",col=c("cyan")
# add a horizontal line at the overall mean income
abline(h=state.means["Income"])
# add title and sub-title
title("Per Capita Income vs Illiteracy")
title(sub="Horizontal line is at overall mean income")
```

Note that *state.x77* is a matrix, not a data frame.

```
is.data.frame(state.x77)
# make a data frame from this matrix
State77 = data.frame(state.x77)
# compare the following two plot commands:
plot(Income ~ Illiteracy, data=State77)
plot(State77$Income ~ Illiteracy,ylab="Income")
```

Measures of Dispersion

It is possible to have two very different datasets with the same means and medians. For that reason, measures of the middle are useful but limited. Another important attribute

of a dataset is its dispersion or variability about its middle. The most useful measures of dispersion are the **range**, **percentiles**, and the **standard deviation**. The **range** is the difference between the largest and the smallest data values. Therefore, the more spread out the data values are, the larger the range will be. However, if a few observations are relatively far from the middle but the rest are relatively close to the middle, the range can give a distorted measure of dispersion.

Percentiles are positional measures for a dataset that enable one to determine the relative standing of a single measurement within the dataset. In particular, the p^{th} $\%ile$ is defined to be a number such that $p\%$ of the observations are less than or equal to that number and $(100-p)\%$ are greater than that number. So, for example, an observation that is at the 75^{th} $\%ile$ is less than only *25%* of the data. In practice, we often cannot satisfy the definition exactly. However, the steps outlined below at least satisfies the spirit of the definition.

1. Order the data values from smallest to largest; include ties.

2. Determine the position,

$$k.ddd = 1 + \frac{p(n-1)}{100}.$$

3. The p^{th} $\%ile$ is located between the k^{th} and the $(k+1)^{th}$ ordered value. Use the fractional part of the position, $.ddd$ as an interpolation factor between these values. If $k = 0$, then take the smallest observation as the percentile and if $k = n$, then take the largest observation as the percentile. For example, if $n = 75$ and we wish to find the 35^{th} percentile, then the position is $1 + 35 * 74/100 = 26.9$. The percentile is then located between the 26^{th} and 27^{th} ordered values. Suppose that these are 57.8 and 61.3, respectively. Then the percentile would be

$$57.8 + .9 * (61.3 - 57.8) = 60.95.$$

Note. Quantiles are equivalent to percentiles with the percentile expressed as a proportion (70^{th} $\%ile$ is the .70 quantile).

The 50^{th} percentile is the median and partitions the data into a lower half (below median) and upper half (above median). The 25^{th}, 50^{th}, 75^{th} percentiles are referred to as *quartiles*. They partition the data into 4 groups with 25% of the values below the 25^{th} percentile (lower quartile), 25% between the lower quartile and the median, 25% between the median and the 75^{th} percentile (upper quartile), and 25% above the upper quartile. The difference between the upper and lower quartiles is referred to as the *inter-quartile range*. This is the range of the middle 50% of the data.

The third measure of dispersion we will consider here is associated with the concept of distance between a number and a set of data. Suppose we are interested in a particular dataset and would like to summarize the information in that data with a single value that

represents the *closest* number to the data. To accomplish this requires that we first define a measure of distance between a number and a dataset. One such measure can be defined as the *total distance between the number and the values in the dataset*. That is, the distance between a number **c** and a set of data values, $X_i,\ 1 \leq i \leq n$, would be

$$D(c) = \sum_{i=1}^{n} |X_i - c|.$$

It can be shown that the value that minimizes *D(c)* is the median. However, this measure of distance is not widely used for several reasons, one of which is that this minimization problem does not always have a unique solution.

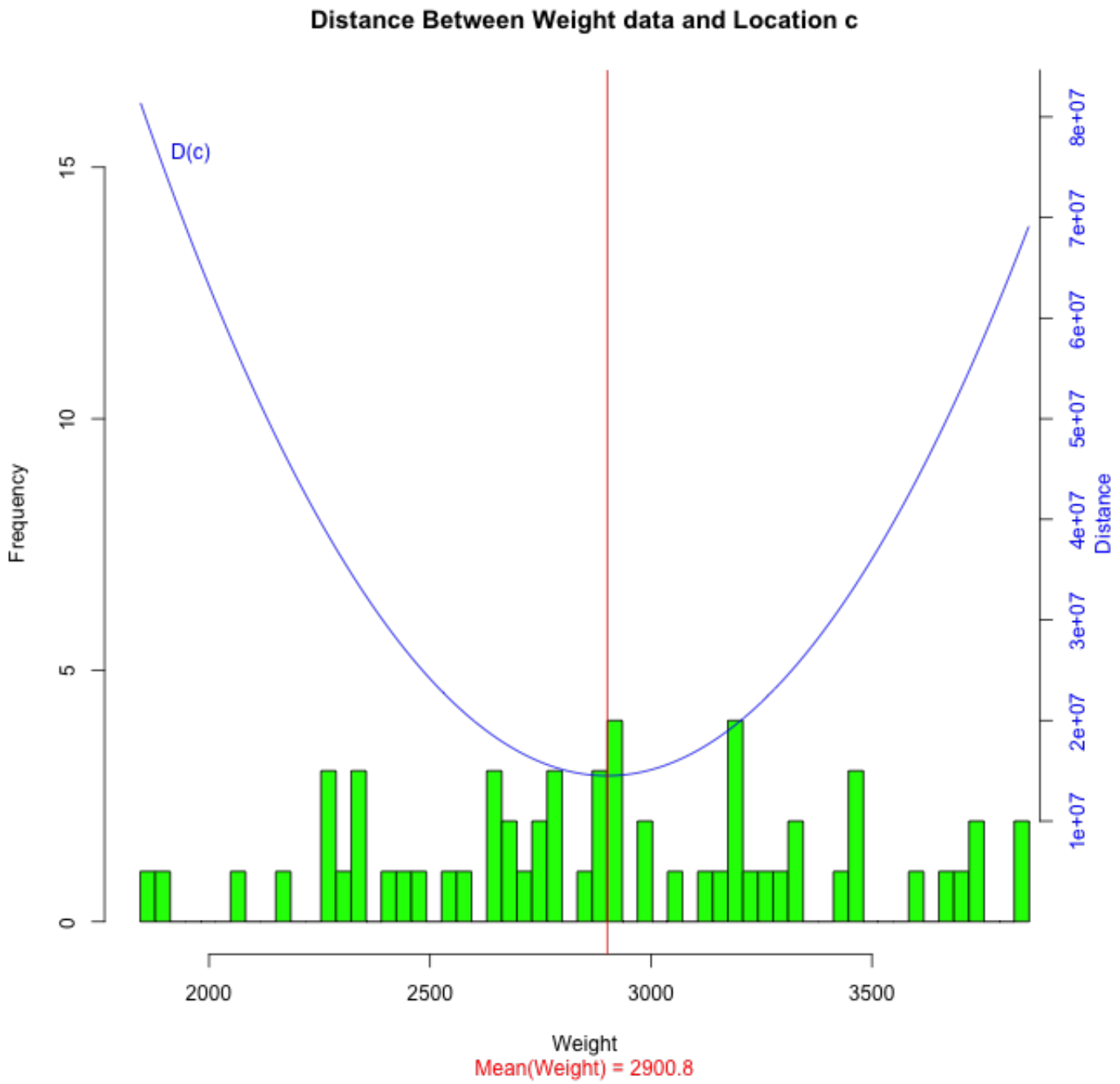
An alternative measure of distance between a number and a set of data that is widely used and does have a unique solution is defined by,

$$D(c) = \sum_{i=1}^{n} (X_i - c)^2.$$

That is, the distance between a number c and the data is the sum of the *squared* distances between c and each data value. We can take as our single number summary the value of c that is closest to the dataset, i.e., the value of c which minimizes $D(c)$. It can be shown that the value that minimizes this distance is $c = \overline{X}$. This is accomplished by differentiating *D(c)* with respect to c and setting the derivative equal to 0.

$$0 = \frac{\partial}{\partial c} D(c) = \sum_{i=1}^{n} -2(X_i - c) = -2 \sum_{i=1}^{n} (X_i - c).$$

As we have already seen, the solution to this equation is $c = \overline{X}$. The graphic below gives a histogram of the Weight data with the distance function *D(c)* superimposed. This graph shows that the minimum distance occurs at the mean of Weight.



The mean is the closest single number to the data when we define distance by the square of the deviation between the number and a data value. The *average distance* between the data and the mean is referred to as the **variance** of the data. We make a notational distinction and a minor arithmetic distinction between variance defined for populations and variance

defined for samples. We use

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2,$$

for population variances, and

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})^2,$$

for sample variances. Note that the unit of measure for the variance is the square of the unit of measure for the data. For that reason (and others), the square root of the variance, called the **standard deviation**, is more commonly used as a measure of dispersion,

$$\sigma = \sqrt{\sum_{i=1}^{N}(X_i - \mu)^2/N},$$

$$s = \sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2/(n-1)}.$$

Note that datasets in which the values tend to be far away from the middle have a large variance (and hence large standard deviation), and datasets in which the values cluster closely around the middle have small variance. Unfortunately, it is also the case that a data set with one value very far from the middle and the rest very close to the middle also will have a large variance.

The standard deviation of a dataset can be interpreted by **Chebychev's Theorem**:

> for any $k > 1$, the proportion of observations within the interval $\mu \pm k\sigma$ is at least $(1 - 1/k^2)$.

For example, the mean of the *Mileage* data is 24.583 and the standard deviation is 4.79. Therefore, at least 75% of the cars in this dataset have weights between $24.583 - 2 * 4.79 = 15.003$ and $24.583 + 2 * 4.79 = 34.163$. Chebychev's theorem is very conservative since it is applicable to every dataset. The actual number of cars whose Mileage falls in the interval (15.003,34.163) is 58, corresponding to 96.7%. Nevertheless, knowing just the mean and standard deviation of a dataset allows us to obtain a rough picture of the distribution of the data values. Note that the smaller the standard deviation, the smaller is the interval that is guaranteed to contain at least 75% of the observations. Conversely, the larger the standard deviation, the more likely it is that an observation will not be close to the mean. From the point of view of a manufacturer, reduction in variability of some product characteristic would correspond to an increase of consistency of the product. From the point of view of a financial manager, variability of a portfolio's return is referred to as volatility.

Note that **Chebychev's Theorem** applies to all data and therefore must be conservative. In many situations the actual percentages contained within these intervals are much higher

than the minimums specified by this theorem. If the shape of the data histogram is known, then better results can be given. In particular, if it is known that the data histogram is approximately bell-shaped, then we can say
$\mu \pm \sigma$ contains approximately 68%,
$\mu \pm 2\sigma$ contains approximately 95%,
$\mu \pm 3\sigma$ contains essentially all
of the data values. This set of results is called the **empirical rule**. Later in the course we will study the bell-shaped curve (known as the normal distribution) in more detail.

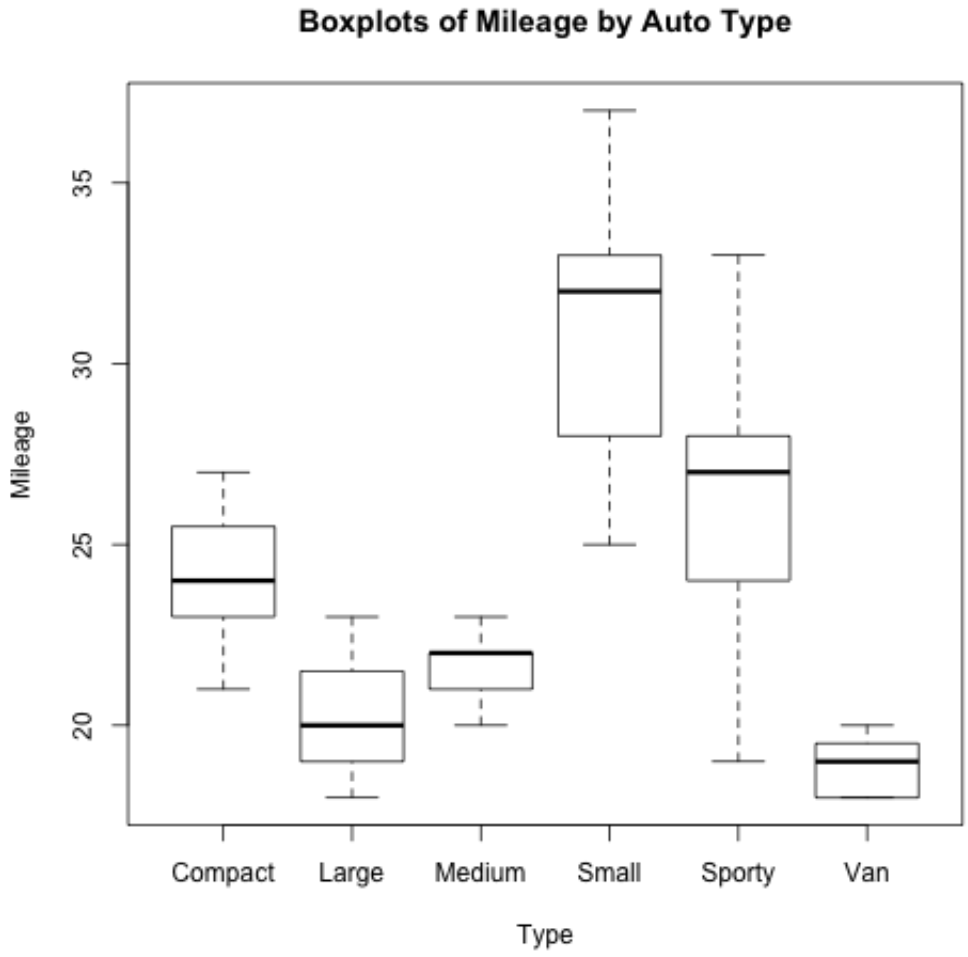
The relative position of an observation in a data set can be represented by its distance from the mean expressed in terms of the s.d. That is,

$$z = \frac{x - \mu}{\sigma},$$

and is referred to as the z-score of the observation. NPositive z-scores are above the mean, negative z-scores are below the mean. Z-scores greater than 2 are more than 2 s.d.'s above the mean. From Chebychev's theorem, at least 75% of observations in any dataset will have z-scores between -2 and 2

Since z-scores are dimension-less, then we can compare the relative positions of observations from different populations or samples by comparing their respective z-scores. For example, directly comparing the heights of a husband and wife would not be appropriate since males tend to be taller than females. However, if we knew the means and s.d.'s of males and females, then we could compare their z-scores. This comparison would be more meaningful than a direct comparison of their heights.

If the data histogram is approximately bell-shaped, then essentially all values should be within 3 s.d.'s of the mean, which is an interval of width 6 s.d.'s. A small number of observations that are unusually large or small can greatly inflate the s.d. Such observations are referred to as outliers. Identification of outliers is important, but this can be difficult since they will distort the mean and the s.d. For that reason, we can't simply use $\overline{X} \pm 2s$ or $\overline{X} \pm 3s$ for this purpose. We instead make use of some relationships between quartiles and the s.d. of bell-shaped data. In particular, if the data histogram is approximately bell-shaped, then $IQR \approx 1.35s$. This relationship can be used to define a robust estimate of the s.d. which is then used to identify outliers. Observations that are more than $1.5(IQR) \approx 2s$ from the nearest quartile are considered to be outliers. Boxplots in **R** are constructed so that the box edges are at the quartiles, the median is marked by a line within the box, and this the box is extended by whiskers indicating the range of observations that are no more than 1.5(IQR) from the nearest quartile. Any observations falling outside this range are plotted with a circle. For example, the following plot shows boxplots of mileage for each automobile type.



R Notes. The data set
`http://www.utdallas.edu/~ammann/stat3355scripts/BirthwtSmoke.csv`
is used to illustrate Chebychev's Theorem and the empirical rule. This is a csv file that contains two columns: *BirthWeight* gives weight of babies born to 1226 mothers and *Smoker* indicates whether or not the mother was a smoker.

```
# import data into R
BW = read.table("http://www.utdallas.edu/~ammann/stat3355scripts/BirthwtSmoke.csv",heade
# note that Smoker is automatically converted to a factor
# obtain mean and s.d. for all babies
allBirthWeights = BW[,"BirthWeight"]
meanAllWeights = mean(allBirthWeights)
sdAllWeights = sd(allBirthWeights)
# construct histogram of all weights
hist(allBirthWeights, main="Histogram of Birth Weights\nAll Mothers included",col="cyan"
```

```
# now report application of Chebychev's Theorem
# print line that gives the interval +- 2 s.d.'s from mean using paste function
cheb.int = meanAllWeights + 2*c(-1,1)*sdAllWeights
cat("At least 3/4 of birth weights are in the interval\n")
cat(paste("[",round(cheb.int[1],1),", ",
    round(cheb.int[2],1),"]",sep=""),"\n")
cat("Since histograph is approximately bell-shaped,\n")
cat("we can say that approximately 95% will be in this interval.\n")
# now count how many are in the interval
allprop = mean(allBirthWeights > cheb.int[1] & allBirthWeights < cheb.int[2])
cat(paste("Actual proportion in this interval is",round(allprop,3)),"\n")
```

Next repeat this separately for mothers who smoke and mothers who don't smoke.

```
# extract weights for mothers who smoked
smokeBirthWeights = allBirthWeights[BW$Smoker == "Yes"]
meanSmokeWeights = mean(smokeBirthWeights)
sdSmokeWeights = sd(smokeBirthWeights)
# construct histogram of smoke weights
hist(smokeBirthWeights, main="Histogram of Birth Weights: Smoking Mothers",col="cyan")
# now report application of Chebychev's Theorem
# print line that gives the interval +- 2 s.d.'s from mean using paste function
cheb.int = meanSmokeWeights + 2*c(-1,1)*sdSmokeWeights
cat("At least 3/4 of birth weights from mothers who smoked are in the interval\n")
cat(paste("[",round(cheb.int[1],1),", ",
    round(cheb.int[2],1),"]",sep=""),"\n")
cat("Since histograph is approximately bell-shaped,\n")
cat("we can say that approximately 95% will be in this interval.\n")
# now count how many are in the interval
smokeprop = mean(smokeBirthWeights > cheb.int[1] & smokeBirthWeights < cheb.int[2])
cat(paste("Actual proportion in this interval is",round(smokeprop,3)),"\n")
# extract weights for mothers who did not smoke
nonSmokeBirthWeights = allBirthWeights[BW$Smoker == "No"]
meannonSmokeWeights = mean(nonSmokeBirthWeights)
sdnonSmokeWeights = sd(nonSmokeBirthWeights)
# construct histogram of non smoker weights
hist(nonSmokeBirthWeights, main="Histogram of Birth Weights: Non-smoking Mothers",col="c
# now report application of Chebychev's Theorem
# print line that gives the interval +- 2 s.d.'s from mean using paste function
cheb.int = meannonSmokeWeights + 2*c(-1,1)*sdnonSmokeWeights
cat("\nAt least 3/4 of birth weights from mothers who did not smoke are in the interval\
cat(paste("[",round(cheb.int[1],1),", ",
    round(cheb.int[2],1),"]",sep=""),"\n")
```

```
cat("Since histograph is approximately bell-shaped,\n")
cat("we can say that approximately 95% will be in this interval.\n")
# now count how many are in the interval
nonsmokeprop = mean(nonSmokeBirthWeights > cheb.int[1] & nonSmokeBirthWeights < cheb.int
cat(paste("Actual proportion in this interval is",round(nonsmokeprop,3)),"\n")
# now create graphic with both histograms aligned vertically
# use same x-axis limits to make them comparable
png("WeightHists.png",width=600,height=960)
par(mfrow=c(2,1),oma=c(1,0,0,0))
Smoke.tab = table(BW$Smoker)
hist(smokeBirthWeights, main="",col="cyan",xlab="Birth weight",xlim=range(allBirthWeight
title(sub=paste("Smoking Mothers: n =",Smoke.tab["Yes"]))
mtext("Histogram of Birth Weights",outer=TRUE,cex=1.2,font=2,line=-2)
hist(nonSmokeBirthWeights, main="",col="cyan",xlab="Birth weight",xlim=range(allBirthWei
title(sub=paste("Non-smoking Mothers: n =",Smoke.tab["No"]))
graphics.off()
```

A more effective way to visualize the differences in birth weights between mothers who smoke and those who do not is to use boxplots. These can be obtained through the **plot()** function. This function is what is referred to in **R** as a generic function. For this data what we would like to show is how birth weights depend on smoking status of mothers. We can do this using the formula interface of **plot()** as follows.

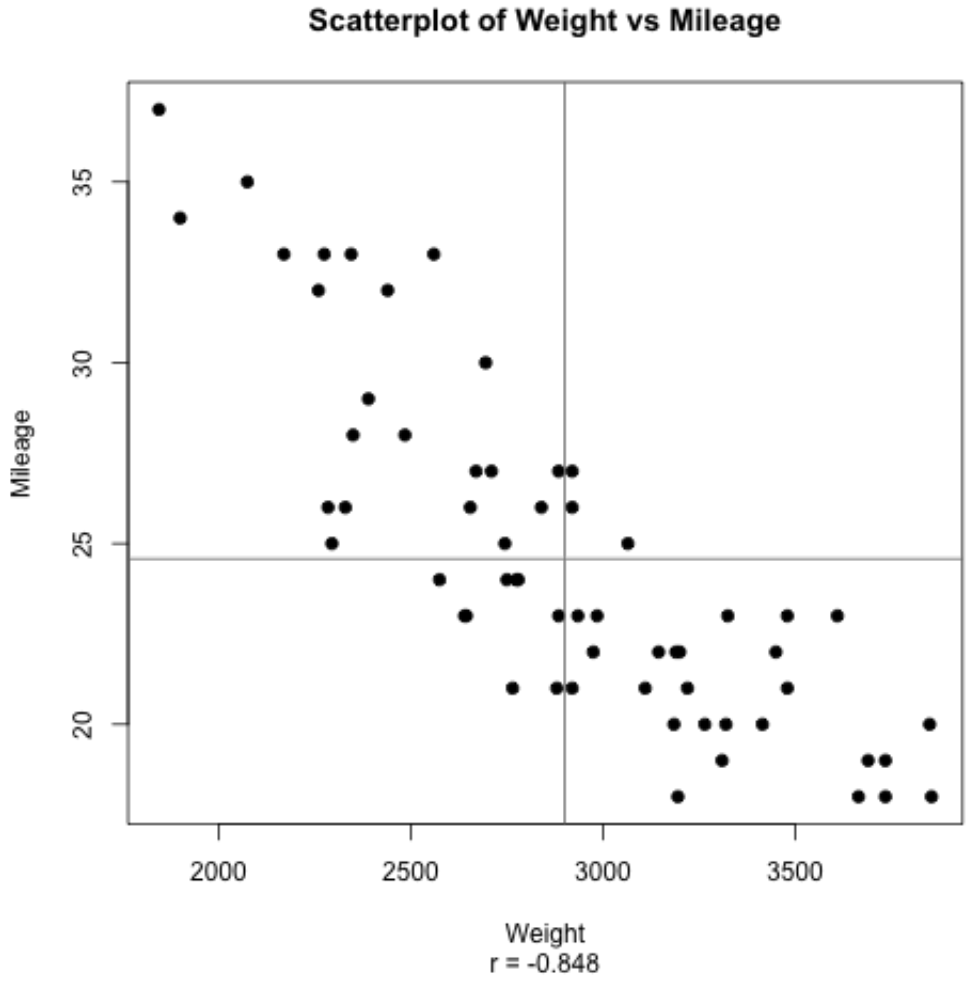
```
plot(BirthWeight ~ Smoker, data=BW)
```

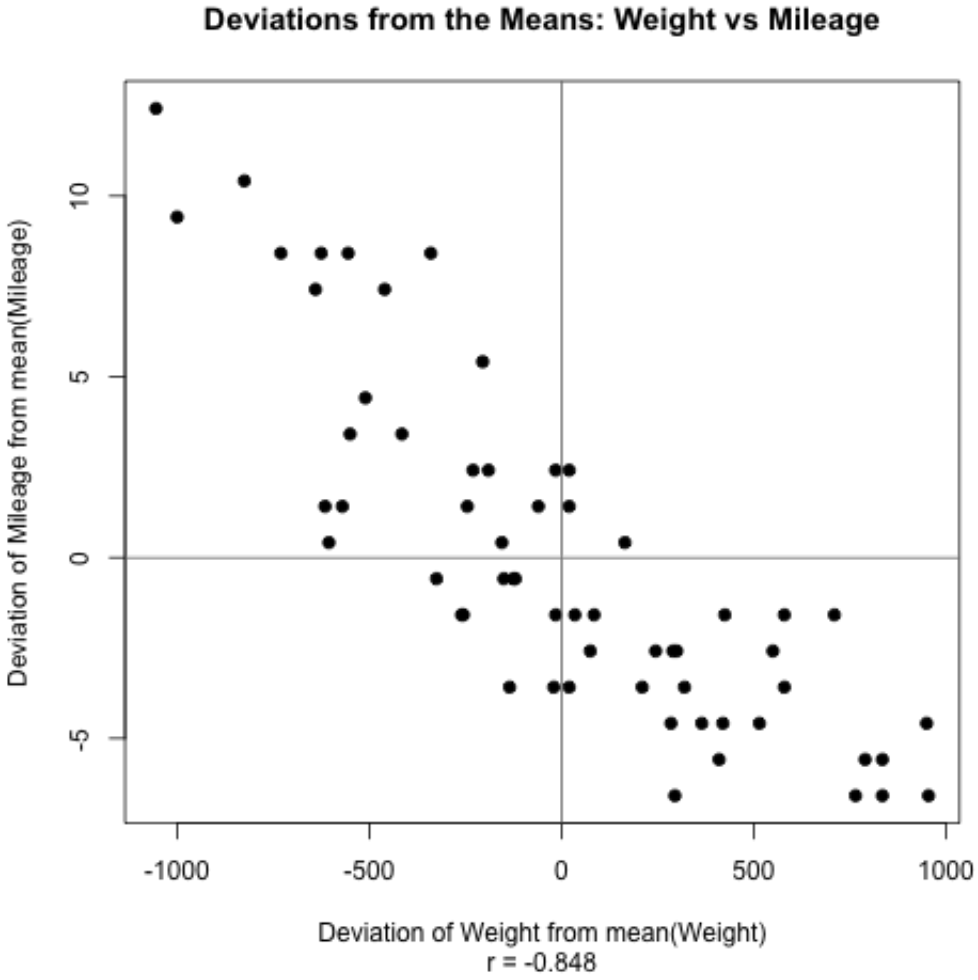
The first argument is the formula which can be read as: *BirthWeight depends on Smoker*. The *data=BW* argument tells **R** that the names used in the formula are variables in a data frame named *BW*. In this case the response variable *BirthWeight* is a numeric variable and the independent variable *Smoker* is a factor. For this type of formula **plot()** generates separate boxplots for each level of the factor. The box contains the middle 50% of the responses for a group (lower quartile - upper quartile) and the middle line within the box represents the group mean. The dashed lines and whisker represent a robust estimate of a 95% coverage interval derived from the median and inter-quartile range instead of the mean and s.d. Now let's create a stand-alone script that makes this plot look better by adding color, a title, and group sizes.

```
BW = read.table("http://www.utdallas.edu/~ammann/stat3355scripts/BirthwtSmoke.csv",heade
bw.col = c("SkyBlue","orange")
png("BirthWeightBox.png",width=600,height=600)
plot(BirthWeight ~ Smoker, data=BW, col=bw.col,ylab="Birth Weight")
title("Birth Weights vs Smoking Status of Mothers")
Smoke.tab = table(BW$Smoker)
axis(side=1, at=seq(2), labels=paste("n=",Smoke.tab, sep=""), tick=FALSE, line=1)
graphics.off()
```

Measures of Association

The automobile dataset given above includes both Weight and Mileage of 60 automobiles. In addition to describing location and dispersion for each variable separately, we also may be interested in what kind of relationship exists between these variables. The following figure represents a scatterplot of these variables with the respective means superimposed. This shows that for a high percentage of cars, those with above average Weight tend to have below average Mileage, and those with below average Weight have above average Mileage. This is an example of a decreasing relationship, and most of the data points in the plot fall in the upper left/lower right quadrants. In an increasing relationship, most of the points will fall in the lower left/upper right quadrants.





We can derive a measure of association for two variables by considering the deviations of the data values from their respective means. Note that the product of deviations for a data point in the lower left or upper right quadrants is positive and the product of deviations for a data point in the upper left or lower right quadrants is negative. Therefore, most of these products for variables with a strong increasing relationship will be positive, and most of these products for variables with a strong decreasing relationship will be negative. This implies that the sum of these products will be a large positive number for variables that have a strong increasing relationship, and the sum will be a large negative number for variables that have a strong decreasing relationship. This is the motivation for using

$$r = \frac{\frac{1}{N}\sum_{i=1}^{N}(X_i - \mu_x)(Y_i - \mu_y)}{\sigma_x \sigma_y}.$$

as a measure of association between two variables. This quantity is called the **correlation coefficient**. The denominator of r is a scale factor that makes the correlation coefficient

dimension-less and scales so that $0 \leq |r| \leq 1$. Note that this can be expressed equivalently as

$$r = \frac{\frac{1}{n-1}\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{s_x s_y}.$$

If the correlation coefficient is close to 1, then the variables have a strong increasing relationship and if the correlation coefficient is close to -1, then the variables have a strong decreasing relationship. If the correlation is exactly 1 or -1, then the data must fall exactly on a straight line. The correlation coefficient is limited in that it is only valid for *linear* relationships. A correlation coefficient close to 0 indicates that there is no *linear* relationship. There may be a strong relationship in this case, just not linear. Furthermore, the correlation may understate the strength of the relationship even when r is large, if the relationship is non-linear.

The correlation coefficient between Weight and Mileage is -0.848. This is a fairly large negative number, and so there is a fairly strong linear, decreasing relationship between Weight and Mileage. This is confirmed by the scatterplot. Since these variables are so strongly related, we can ask how well can we predict Mileage just by knowing the Weight of a vehicle. To answer this question, we first define a measure of distance between a dataset and a line.

Suppose we have measured two variables for each individual in a sample, denoted by $\{(X_1, Y_1), \cdots, (X_n, Y_n)\}$, and we wish to predict the value of Y given the value of X for a particular individual using a straight line for the prediction. A reasonable approach would be to use the line that comes closest to the data for this prediction. Let $Y{=}a{+}bX$ denote the equation of a prediction line, and let $\hat{Y}_i = a + bX_i$ denote the predicted value of Y for X_i. The difference between an actual and predicted Y-value represents the error of prediction for that data point. We define the *distance* between a prediction line and a point in the dataset to be the square of the prediction error for that observation. The total distance between the actual and predicted Y-values is then the sum of the squared errors, which is the variance of the prediction errors multiplied by n. Since the predicted values, and hence the errors, depend on the slope and intercept of the prediction line, we can express this total distance by

$$D(a, b) = \sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 = \sum_{i=1}^{n}(Y_i - a - bX_i)^2.$$

Our goal now is to find the line that is closest to the data using this definition of distance. This line has slope and intercept that minimize $D(a, b)$. We can use differential calculus to find the minimum.

$$\frac{\partial}{\partial a}D(a, b) = -2\sum_{i=1}^{n}(Y_i - a - bX_i),$$

$$\frac{\partial}{\partial b}D(a, b) = -2\sum_{i=1}^{n}X_i(Y_i - a - bX_i).$$

Setting these equal to 0 gives the system of equations

$$0 = \sum_{i=1}^{n}(Y_i - a - bX_i) = n(\overline{Y} - b\overline{X} - a),$$

$$0 = \sum_{i=1}^{n} X_i Y_i - na\overline{X} - b\sum_{i=1}^{n} X_i^2.$$

Therefore,

$$a = \overline{Y} - b\overline{X},$$

and, after substituting for a in the second equation and solving for b,

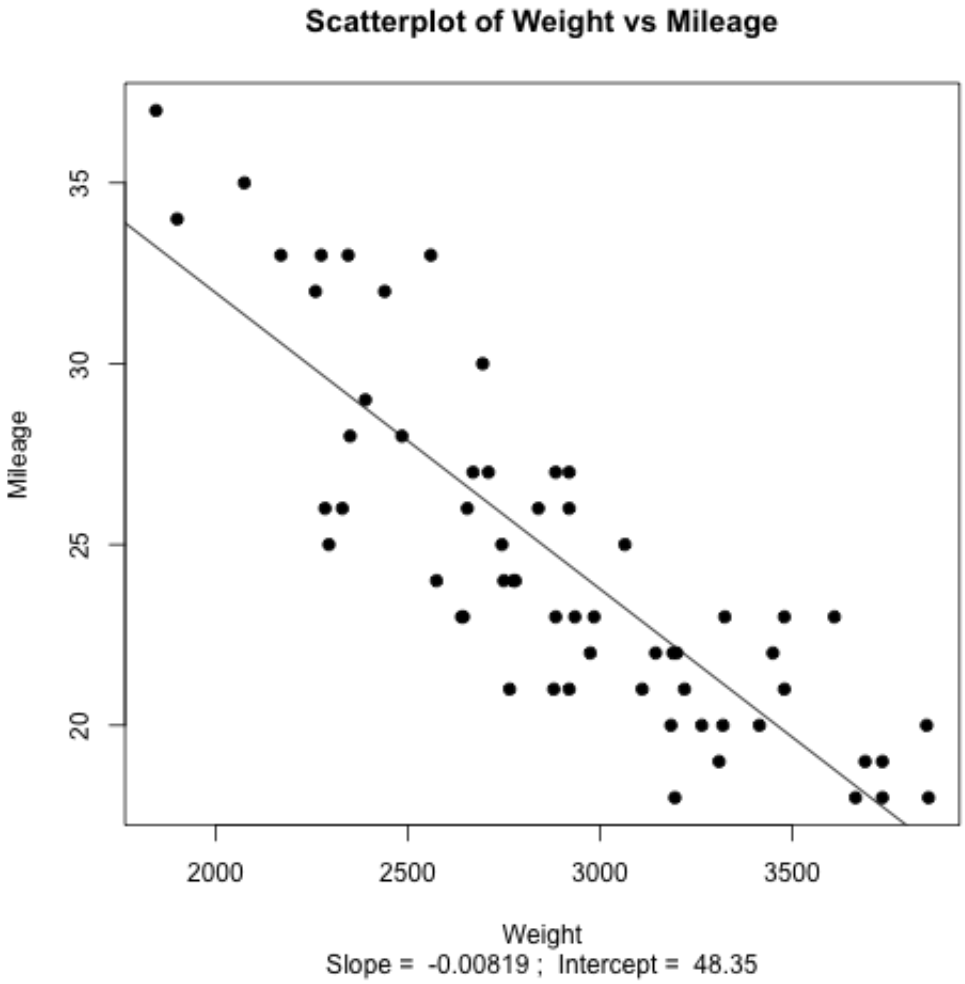
$$b = \frac{\sum_{i=1}^{n} X_i Y_i - n\overline{XY}}{\sum_{i=1}^{n} X_i^2 - n\overline{X}^2}.$$

It can be shown that the numerator equals $(n-1)rs_x s_y$ and the denominator equals $(n-1)s_x^2$. Hence,

$$b = r\frac{s_y}{s_x}, \quad a = \overline{Y} - b\overline{X}.$$

The prediction line, referred to as the **least squares regression line**, is then

$$\hat{Y} = a + bX.$$



The next question that can be asked related to this prediction problem is how well does the prediction line predict? We can't answer that question completely yet because the full answer requires inference tools that we have not yet covered, but we can give a descriptive answer to this question. The distance measure, *D(a,b)*, represents the variance of the prediction errors. One way of describing how well the prediction line performs is to compare it to the best prediction we could obtain without using the X values to predict. In that case, our predictor would be a single number. We have already seen that the closest single number to a dataset is the mean of the data, so in this case, the best predictor based only on the Y values is $\overline{Y}$. This corresponds to a horizontal line with intercept $\overline{Y}$, and so the distance between this line and the data is $D(\overline{Y}, 0)$. This quantity represents the error variance for the best predictor that does not make use of the X values, and so the difference,

$$D(\overline{Y}, 0) - D(a, b),$$

represents the reduction in error variance (improvement in prediction) that results from use

of the X values to predict. If we express this as a percent,

$$100\frac{D(\overline{Y},0) - D(a,b)}{D(\overline{Y},0)},$$

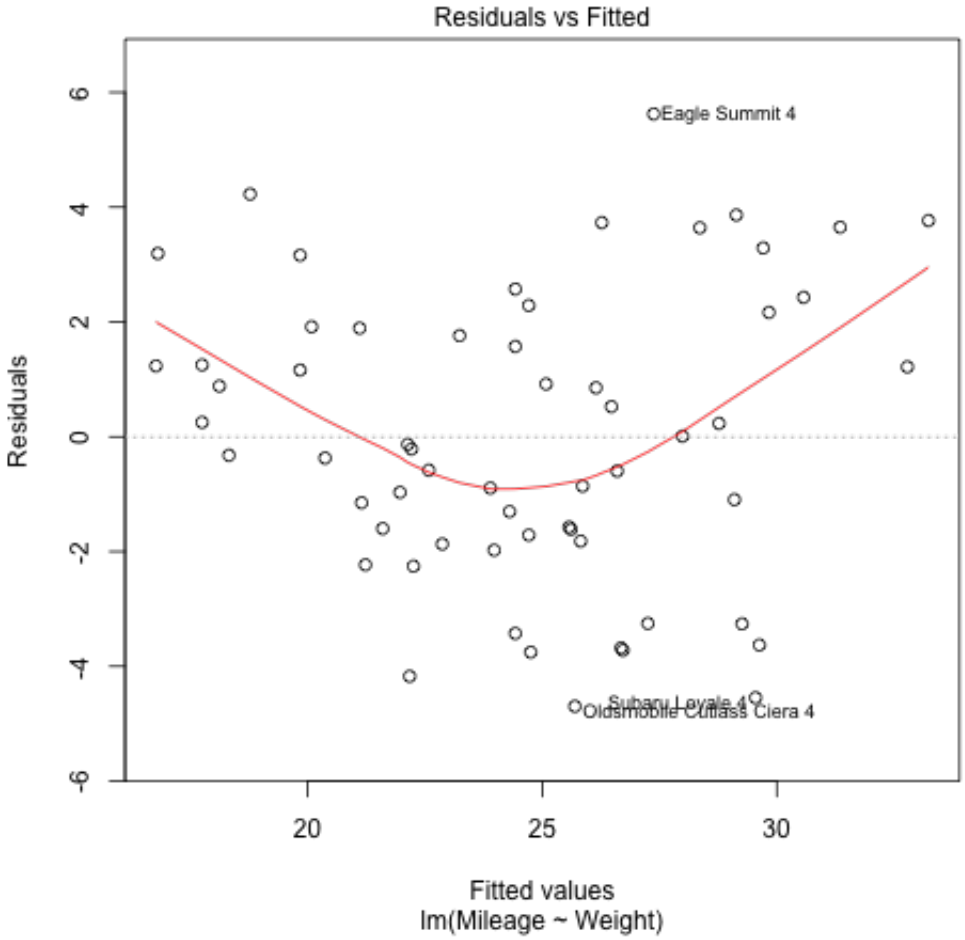
then this is the percent of the error variance that can be removed if we use the least squares regression line to predict as opposed to simply using the mean of the Y's. It can be shown that this quantity is equal to the square of the correlation coefficient,

$$r^2 = \frac{D(\overline{Y},0) - D(a,b)}{D(\overline{Y},0)}.$$

R-squared also can be interpreted as the proportion of variability in the Y-variable that can be explained by the presence of a linear relationship between X and Y.

In the automobile example, the correlation between Weight and Mileage was r = *-0.848*, and so $r^2 = 0.719$. If we use the regression line to predict Mileage based on Weight, we can remove 71.9% of the variance of the Mileage data by using Weight to predict Mileage. Another way of expressing this is to ask: Why don't all cars have the same mileage. Part of the answer to that question is that cars don't all weigh the same and there is a fairly strong linear relationship between weight and mileage that accounts for 71.9% of the variability in mileage. This leaves 28.1% of this variability that is related to other factors, including the possibility of a non-linear relationship between Mileage and Weight.

To help judge the adequacy of a linear regression fit, we can plot the residuals vs the predictor variable X. The residuals are the prediction errors, $e_i = Y_i - \hat{Y}_i$, $1 \leq i \leq n$. If a linear fit is reasonable, then the residuals should have no discernable relationship with X and should be essentially noise. This plot for a linear fit to predict Mileage based on Weight is shown below.



This shows that the residuals are still related to Weight, so a linear fit is not adequate. Note that removal of the linear component of the relationship between weight and mileage, as represented by the residuals from a linear fit, does a better job of revealing this non-linearity than a scatterplot of these variables. This will be discussed in greater detail later.

Now suppose we only wish to consider cars whose engine displacements are no more than 225. We can define a logical expression that represents such cars and use that to subset the fuel data frame:

```
ndx = fuel.frame$Disp < 225
fuel1 = fuel.frame[ndx,]
```

Then we can use the *fuel1* data frame to plot Mileage versus Weight and to fit a linear regression model.

```
plot(Mileage ~ Weight,data=fuel1,pch=19)
title("Scatterplot of Weight vs Mileage")
```

```
Disp.lm = lm(Mileage~Weight,data=fuel1)
Disp.coef = coef(Disp.lm)
abline(Disp.coef,col="red")
plot(residuals(Disp.lm) ~ Weight,data=fuel1,pch=19,ylab="Residuals")
abline(h=0,col="red")
title("Residuals vs Weight\nData = fuel1")
```

The ideal situation is that the only thing left after we remove the linear relationship from the response variable, Mileage, is noise.

```
# qqnorm plot
qqnorm(residuals(Disp.lm),pch=19)
qqline(residuals(Disp.lm),col="red")
```

It is important to remember that correlation is a mathematical concept that says nothing about causation. The presence of a strong correlation between two variables indicates that there *may* be a causal relationship, but does not prove that one exists, nor does it indicate the direction of any causality.

The **R** code to generate the graphics in this section can be found at:
`http://www.utdallas.edu/~ammann/stat3355scripts/NumericGraphics.r`

An example using the crabs data can be found at:
`http://www.utdallas.edu/~ammann/stat3355scripts/crabs02042016.r`

Introduction to Probability Models

Probability is a mathematical description of a process whose outcome is uncertain. We call such a process an **experiment**. This could be something as simple as tossing a coin or as complicated as a large-scale clinical trial consisting of three phases involving hundreds of patients and a variety of treatments. The **sample space** of an experiment is the set of all possible outcomes of the experiment, and an **event** is a set of possible outcomes, that is, a subset of the sample space.

For example, the sample space of an experiment in which three coins are tossed consists of the outcomes

$$\{HHH, HHT, HTH, THH, HTT, THT, TTH, TTT\}$$

while the sample space of an experiment in which a disk drive is selected and the time until first failure is observed would consist of the positive real numbers. In the first case, the event that exactly one head is observed is the set $\{HHT, HTH, THH\}$. In the second case, the event that the time to first failure of the drive exceeds 1000 hours is the interval $(1000, \infty)$.

Probability arose originally from descriptions of games of chance – gambling – that have their origins far back in human history. It is usually interpreted as the proportion or percentage of times a particular outcome is observed if the experiment is repeated a large number of times. We can think of this proportion as representing the *likelihood* of that outcome

occurring whenever the experiment is performed. **Probability** is formally defined to be a function that assigns a real number to each event associated with an experiment according to a set of basic rules. These rules are designed to coincide with our intuitive notions of likelihood, but they must also be mathematically consistent.

This mathematical representation is simplest when the sample space contains a finite or countably infinite number of elements. However, our mathematics and our intuition collide when working with an experiment that has an uncountable sample space, for example an interval of real numbers. Consider for example the following experiment. You purchase a spring driven clock, set it at 12:00 (ignore AM and PM), wind the clock and let it run until it stops. We can represent the sample space of this experiment as the interval, $[0, 12)$, and we can ask questions such as

1. What is the probability the clock stops between 1:00 and 2:00?
2. What is the probability the clock stops between 4:00 and 4:30?
3. What is the probability the clock stops between 7:05 and 7:06?

We can answer each of these questions using our intuitive ideas of likelihood. For the first question, since we know nothing about the clock, we can assume that there is no preference of one interval of time over any other interval of time for the clock to stop. Therefore, we would expect that each of the 12 intervals of length one hour are equally likely to contain the stopping time of the clock, and so the likelihood that it stops between 1:00 and 2:00 would be $1/12$. Similarly, the likelihood that it stops between 4:00 and 4:30 would be $1/24$ since there are 24 intervals of length $1/2$ hour, and the likelihood that it stops between 7:05 and 7:06 would be $1/720$ since there are 720 intervals of length one minute. In each case our intuition tells us that the likelihood of an event for this experiment is the reciprocal of the number of non-overlapping intervals of the same length, since each such interval is assumed to be equally likely to contain the stopping point of the clock. Note also that the interval $[1, 2)$, corresponding to the times between 1:00 and 2:00, contains the non-overlapping intervals, $[1, 1.5)$ and $[1.5, 2)$. Each of these intervals would have likelihoods $1/24$ and the sum of these two likelihoods equals the likelihood of the entire interval. This illustrates the additive nature of likelihood that we have for this concept.

A problem occurs if we ask a question such as what is the probability that the clock stops at precisely $\sqrt{2}$ minutes past 1? In this case there is an uncountably infinite number of such times in the interval $[0, 12)$, so that the likelihood we would assign to such an event would be $1/\infty = 0$. However, the sum of the likelihoods for all such events between 1:00 and 2:00 would be 0, not $1/12$ as we have derived above. This inconsistency requires that we modify the rules somewhat. In the case of uncountably infinite sample spaces, we only require that probability be defined for an *interesting* set of events. In the case of the clock experiment, this *interesting* set of events would consist of all interval subsets of the sample space with positive length along with events that can be formed from countable unions and intersections of such intervals. This collection of events is referred to as the **probability space** for the experiment. In the case of finite or countably infinite sample spaces, the probability space

can be the set of all possible subsets of the sample space. Unless specified otherwise, all events used here are assumed to be in the probability space.

The basic rules or axioms of probability are then:

1. Probability is a function $P : \mathcal{F} \to [0, 1]$, where $\mathcal{F}$ is the probability space. That is, the probability function assigns a number between 0 and 1 to each event in the probability space.

2. $P(S) = 1$, where S is the sample space. That is, the probability that an outcome in the sample space occurs is 1.

3. For any countable collection of mutually exclusive events in $\mathcal{F}$, $A_i,\ i \geq 1$, we have

$$P(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i).$$

   That is, the probability of the union of non-overlapping events equals the sum of the individual probabilities.

All other properties of probability derive from these basic axioms along with any additional definitions we construct.

As noted previously, when working with experiments that have equally likely outcomes, it is only necessary to count the number of outcomes contained in events to determine their probabilities. Events in many such experiments involve the selection of objects from a population. There are different methods used for counting outcomes for such situations depending on whether or not the selection order of the selected objects is recognized and whether a selected object is returned (**selection with replacement**) to the population before the next selection is made or not returned (**selection without replacement**). We use the term **permutation** to refer to selection of objects in which selection order is distinguished and use the term **combinations** to refer to the case in which selection order is not distinguished. We will consider here three of these methods, permutations with and without replacement, and combinations without replacement.

Permutations without replacement

If the object selected is not returned to the population before the next object is selected, then an object can appear in the selected subset no more than once. There are n choices in the population to fill the first position, but then that leaves *n-1* choices in the population to fill the second position. Therefore, there are $n(n-1)$ ways to fill the first two positions. Continuing this argument, we can see that there are $n(n-1)\dots(n+1-k)$ ways to select k objects without replacement from a population of n objects when selection order is distinguished. This number is commonly expressed using factorial notation,

$$n(n-1)\dots(n+1-k) = \frac{n!}{(n-k)!}$$

Permutations with replacement

This case occurs when we wish to select k objects with replacement from a population of n objects and selection order is distinguished. Replacement implies that the same object could be selected multiple times. What is required is to count the number of distinct sets of k objects could be selected in this way. We can view this selection process by considering the ways in which each of the positions, $1, \ldots, n$, of the set are filled. Note that there are n choices in the population to fill the first position, and since the object selected for this first position is then returned to the population, there are n choices available for the second selection as well. Therefore, there are n^2 ways to fill the first two positions. Continuing this argument, we can see that there are n^k ways to select k objects with replacement from a population of n distinguishable objects.

Combinations without replacement

The only difference between this case and the case of permutations without replacement is that the selection order of the k selected objects is not distinguished here. This implies that a different arrangement of the same objects is not counted for this case and so this case involves simply selecting a subset of size k from the population. Therefore, we can view the number of permutations without replacement as a two-stage process: first select a subset (combinations without replacement) and then generate every possible rearrangement of each of these subsets. Note that the number of ways to generate every possible rearrangement of k objects is equivalent to counting the number of permutations without replacement of k objects selected from a population of size k, and so is equal to $k!$. Denote by $C(n,k)$ the number of combinations without replacement. Then we have,

$$\frac{n!}{(n-k)!} = C(n,k)k!.$$

Hence,

$$C(n,k) = \frac{n!}{k!(n-k)!}.$$

This quantity is usually denoted by

$$\binom{n}{k}$$

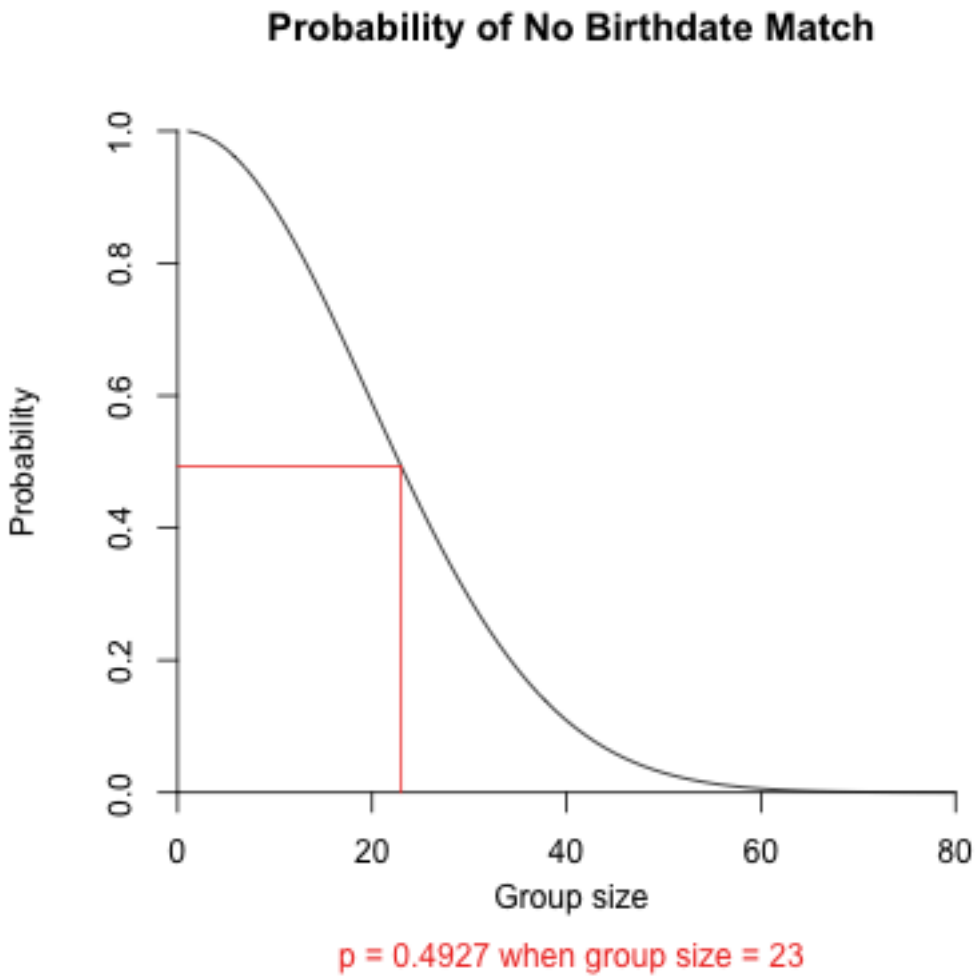
Examples

Birthday Problem. This is a classic example of how probability in some applications does not coincide with our intuition. Suppose we have a class of n individuals and want to determine the probability that there is at least one pair of individuals who have the same birthday. To simplify this problem, we will ignore birthdays that occurred on Feb. 29 during a leap year and count those as occurring on March 1. The model we will assume

for this problem treats an individual's birthdate as if it was randomly selected from the set of 365 possible birthdays. Therefore, the experiment in which each individual selects a birthdate is an experiment with equally likely outcomes. Therefore, we must count the number of ways to select n birthdays from the population of 365 possible birthdays, and then count the number of ways to select n birthdays with at least one matching pair. It turns out to be easier to count the number of ways to select n birthdays with no matches. This is equivalent to counting the number of permutations without replacement of k objects selected from a population of 365 objects. This number is therefore *365!/(365-n)!*. The total number of possible birthdates for this group is equivalent to the number of permutations with replacement of n birthdates from the population of 365 possible birthdates. This gives,

$$P(\textit{no birthdate matches}) = \frac{365!/(365-n)!}{365^n}.$$

A plot of this probability as a function of n is given below. Note that when n=23, there is about a 50% probability of no matches in the group, and when n=50, there is about a 3% chance of no matches in the group.


Probability of No Birthdate Match
Probability
0.0
0.2
0.4
0.6
0.8
1.0
0
20
40
60
80
Group size
p = 0.4927 when group size = 23

Binomial coefficients. Note that the number of combinations without replacement occurs in the binomial series,

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}.$$

Now consider an experiment in which two gamblers play a series of 10 games, the results of which are independent. That is, the event that the first gambler wins (or loses) on game r is independent of the event that he wins (or loses) any other game. Suppose that the probability that the first gambler wins a particular game is p, and his probability of winning any other game is the same. Find the probability that the first gambler wins exactly 4 games. To solve this problem, first note that an arbitrary outcome of this experiment can be represented by a string of 10 characters, each of which is either W or L, denoting the outcomes of each game. The event that the first gambler wins 4 games consists of all possible strings in which W occurs 4 times and L occurs 6 times. Each such string can be specified by the 4 positions of W in this string. For example, the outcome WWWWLLLLLL could be specified by the positions, 1234 of W. Since the games are independent, the probability of observing this outcome would be

$$P(WWWWLLLLLL) = pppp(1-p)(1-p)(1-p)(1-p)(1-p)(1-p) = p^4(1-p)^6.$$

Any other outcome with exactly 4 wins would just be a rearrangement of the 10 characters in the string, and so would have the same probability. Therefore, the probability that the first gambler wins exactly 4 games is this probability times the number of such outcomes. We can obtain this number by counting the number of combinations of 4 positions taken from the possible 10 positions for W in the string. Hence,

$$P(4\ wins) = \binom{10}{4} p^4(1-p)^6.$$

Using the same arguments, we can see that

$$P(k\ wins\ in\ 10\ games) = \binom{10}{k} p^k(1-p)^{10-k},\ 0 \le k \le 10.$$

We can easily extend this to n games to obtain

$$P(k\ wins\ in\ n\ games) = \binom{n}{k} p^k(1-p)^{n-k},\ 0 \le k \le n.$$

Finally, note that these probabilities are terms in a binomial series, and that

$$\sum_{k=0}^{n} P(k\ wins\ in\ n\ games) = \sum_{k=0}^{n} \binom{n}{k} p^k(1-p)^{n-k} = (p+1-p)^n = 1.$$

Subpopulation selection. Suppose a committee consists of 40 males and 20 females and must select a subcommittee of 5 members. It decides to make this selection randomly.

What is the probability that all 5 members of the subcommittee will be female? What is the probability that at least 2 member of the subcommittee will be male? First note that an outcome of this experiment is a set of 5 members selected without replacement from the committee, and this experiment has equally likely outcomes. To answer the questions, we will first obtain the probability that exactly k members of the subcommittee will be female for $0 \leq k \leq 5$.. Note that if k members are female, then 5-k members will be male. Hence, the number of outcomes contained in the event that exactly k members are female can be obtained by counting the number of ways to select a subset of size k from the 20 females and multiplying that times the number of ways to select a subset of size 5-k from the 40 males. Since order of selection does not count, this number is then

$$\binom{20}{k}\binom{40}{5-k}.$$

The number of outcomes in the sample space is the total number of ways to select a subset of size 5 from the 60 committee members, so the probability that exactly k are female is,

$$P(k) = \frac{\binom{20}{k}\binom{40}{5-k}}{\binom{60}{5}}.$$

We can now answer the questions.

$$\begin{aligned} P(5\ females) &= P(5) = \frac{\binom{20}{5}\binom{40}{0}}{\binom{60}{5}} \\ &= \frac{20!5!55!}{5!15!60!} \\ &= \frac{20 \cdot 19 \cdot 18 \cdot 17 \cdot 16}{60 \cdot 59 \cdot 58 \cdot 57 \cdot 56} \\ &= 0.0028. \end{aligned}$$

$$\begin{aligned} P(at\ least\ 2\ males) &= P(no\ more\ than\ 3\ females) = P(0) + P(1) + P(2) + P(3) \\ &= 1 - P(4) - P(5). \end{aligned}$$

$$\begin{aligned} P(4) &= \frac{\binom{20}{4}\binom{40}{1}}{\binom{60}{5}} \\ &= \frac{20 \cdot 19 \cdot 18 \cdot 17 \cdot 40 \cdot 5}{60 \cdot 59 \cdot 58 \cdot 57 \cdot 56} \\ &= 0.0355. \end{aligned}$$

So, $P(at\ least\ 2\ males) = 1 - 0.0355 - 0.0028 = 1 - 0.0383 = .9617$.

Additional Properties of Probability

The **complement** of an event is defined to be the set of all outcomes contained in the sample space that are not contained in the event. It is denoted by A^c. Note that the complement of the sample space is defined to be the empty set, $\emptyset$, the set with no elements. Also, $A \bigcup \emptyset = A$ and $A \bigcap \emptyset = \emptyset$ for any event A. Therefore, if we set $A_1 = A,\ A_i = \emptyset,\ i \geq 2$, then $\{A_i\}$ is a countable collection of mutually exclusive events. Hence, from axiom 3 we have,

$$P(A) = P(\bigcup_{i=1}^{\infty} A_i) = P(A) + \sum_{i=2}^{\infty} P(\emptyset).$$

Since $P(\emptyset) \geq 0$ from Axiom 1, then this equation implies that $P(\emptyset) = 0$.

Note: mathematical equations are sentences with the same syntax as English and can be read as such. The set operations, *intersection*, *union*, and *complement* are often read as the English equivalents, *and*, *or*, and *not*, respectively. Also, the word *or* used in this context is assumed to mean the *inclusive or*.

Now let $A_i,\ 1 \leq i \leq n$ be a finite collection of mutually exclusive events and set $A_i = \emptyset$ for $i > n$. Then from Axiom 3, we have

$$\begin{aligned} P(\bigcup_{i=1}^{n} A_i) &= P(\bigcup_{i=1}^{\infty} A_i) \\ &= \sum_{i=1}^{\infty} P(A_i) = \sum_{i=1}^{n} P(A_i) + \sum_{i=n+1}^{\infty} P(\emptyset) \\ &= \sum_{i=1}^{n} P(A_i). \end{aligned}$$

That is, the probability of a finite union of mutually exclusive events equals the sum of the probabilities.

Suppose we are interested in an experiment in which the sample space consists of a finite collection of n outcomes, $O_i,\ 1 \leq i \leq n$, and that each outcome is equally likely with probability p. Then the previous result implies that

$$1 = \sum_{i=1}^{n} P(O_i) = np.$$

Therefore, we must have $p = 1/n$. Furthermore, since an event for such an experiment may be written as the union of the individual outcomes contained in the event, then

$$P(A) = \frac{\#\{A\}}{n},$$

where $\#\{A\}$ represents the number of elements in the set A. For experiments with equally likely outcomes, the probability of an event is just the number of outcomes in the event divided by the total number of outcomes.

Next note that A and A^c are mutually exclusive and $A \bigcup A^c = S$. Therefore, from the previous result we have,

$$1 = P(A \bigcup A^c) = P(A) + P(A^c).$$

So, the probability of the complement of an event is one minus the probability of the event. This result is useful for situations in which an event of interest is very complicated and its probability is difficult to obtain directly, but the complement of the event is simple with an easily obtainable probability.

The axioms of probability tell us how to find the probability of the union of mutually exclusive events, but not how to find the probability of the union of arbitrary, not necessarily mutually exclusive, events. We can use the results derived thus far to solve this problem. Suppose we are interested in two events, A and B. We need to write the union of these two events as the union of two mutually exclusive events. This can be done by noting that $A \bigcup B = A \bigcup \{B \bigcap A^c\}$. Since A and $B \bigcap A^c$ are mutually exclusive, then

$$P(A \bigcup B) = P(A) + P(B \bigcap A^c).$$

Next note that $B = \{A \bigcap B\} \bigcup \{B \bigcap A^c\}$, which is a disjoint union. Therefore,

$$P(B) = P(A \bigcap B) + P(B \bigcap A^c)$$

and so,

$$P(B \bigcap A^c) = P(B) - P(A \bigcap B).$$

Combining this with the previous result gives,

$$P(A \bigcup B) = P(A) + P(B) - P(A \bigcap B),$$

the probability of A union B equals the sum of the probabilities minus the probability of the intersection.

In a similar way, we can show that probability is a monotone function. Suppose that $A \subset B$. Then we may express B as a disjoint union, $B = A \bigcup (B \bigcap A^c)$ and apply the additivity property of probability,

$$\begin{aligned} P(B) &= P(A \bigcup (B \bigcap A^c)) \\ &= P(A) + P(B \bigcap A^c) \\ &\geq P(A), \end{aligned}$$

since $P(B \bigcap A^c) \geq 0$. Hence, if $A \subset B$, then $P(A) \leq P(B)$.

Another extension that can be derived directly from the axioms is an extremely useful result called the **Law of Total Probability**. A **partition** of the sample space is defined to be a collection, finite or countably infinite, of mutually exclusive events in the probability space whose union is the sample space. Suppose that $\{B_i\}$ is partition and A is an arbitrary

event. Then $A = \bigcup\{A \bigcap B_i\}$, and the events, $A \bigcap B_i$ are mutually exclusive. The Law of Total Probability is just the application of Axiom 3 to this expression,

$$P(A) = \sum P(A \bigcap B_i).$$

This property allows us to breakdown a complicated event A into more manageable pieces, $A \bigcap B_i$.

Example. Suppose a standard card deck (13 denominations in 4 suits) is well-shuffled and then the top card is discarded. What is the probability that the 2^{nd} card (the new top card) is an ace? Let A enote the event that the 2^{nd} card is an ace. The partitioning events we will use are the events

$$B_1 = \{1^{st} \text{ card is Ace}\}, \quad B_2 = \{1^{st} \text{ card is not Ace}\}$$

Then,

$$\begin{aligned} P(A) &= P(A \bigcap B_1) + P(A \bigcap B_2) \\ &= P(1^{st} \text{ card is Ace} \bigcap 2^{nd} \text{ card is Ace}) + P(1^{st} \text{ card is not Ace} \bigcap 2^{nd} \text{ card is Ace}). \end{aligned}$$

The first term has numerator which is the number of ways the first card is an ace and the second card is an ace, and has denominator which is the total number of different outcomes for the first two cards. We can use permutations to count the number of outcomes for both numerator and denominator. The numerator is $4 \cdot 3$ and the denominator is $52 \cdot 51$. Hence,

$$P(1^{st} \text{ card is Ace} \bigcap 2^{nd} \text{ card is Ace}) = \frac{(4)(3)}{(52)(51)}.$$

Similarly, the second term is

$$P(1^{st} \text{ card is not Ace} \bigcap 2^{nd} \text{ card is Ace}) = \frac{(48)(4)}{(52)(51)}.$$

These give

$$\begin{aligned} P(A) &= \frac{(4)(3)}{(52)(51)} + \frac{(48)(4)}{(52)(51)} \\ &= \frac{(4)(3+48)}{(52)(51)} \\ &= \frac{(4)(51)}{(52)(51)} \\ &= \frac{4}{52} = \frac{1}{13}. \end{aligned}$$

Note that this probability is the same as the probability that the first card is an ace.

Continuous Random Variables

Continuous random variables are variables that take values that could be any real number within some interval. One common example of such variables is *time*, for example, the time to failure of a system or the time to complete some task. Other examples include physical measurements such as length or diameter. As will be seen, continuous random variables also can be used to approximate discrete random variables.

To develop probability models for continuous r.v.'s, it is necessary to make one important restriction: we only consider events associated with these r.v.'s that are defined in terms of intervals of real numbers, including intersections and unions of intervals. Probability models are constructed by representing the probability that a r.v. is contained within an interval as the area under a curve over that interval. That curve is called the *density function* of the r.v. To satisfy the laws of probability, density functions must satisfy the following two conditions:

1. $f(t) \geq 0, \ \forall \ t,$
2. $\int_{-\infty}^{\infty} f(t)dt = 1.$

The second condition corresponds to the requirement that the probability of the entire sample space must be 1. Any function that satisfies these two conditions is the density function of some r.v.

The probability that the r.v. is contained within an interval is then

$$P(a < X \leq b) = \int_a^b f(t)dt.$$

Note that in the case of continuous r.v.'s,

$$P(a < X \leq b) = P(a < X < b) = P(a \leq X < b) = P(a \leq X \leq b),$$

since the area under a curve at a point is 0. The distribution function of a continuous r.v. is given by

$$F(x) = P(X \leq x) = \int_{-\infty}^{x} f(t)dt.$$

Note that the Fundamental Theorem of Calculus implies that

$$f(x) = \frac{d}{dx}F(x).$$

Also note that the value of a density function is not a probability; nor is a density necessarily bounded by 1. It can be thought of as the concentration of likelihood at a point.

The expected value of a continuous r.v. is defined analogously to the expected value of a discrete r.v. with the p.m.f. replaced by the density function and the sum replaced by an integral:

$$E(X) = \int_{-\infty}^{\infty} xf(x)dx.$$

Also, the variance of a continuous r.v. is defined by

$$Var(X) = \int_{-\infty}^{\infty} (x-\mu)^2 f(x)dx,$$

where $\mu = E(X)$. Note that the additive property of integrals gives

$$\begin{aligned} Var(X) &= \int_{-\infty}^{\infty} (x^2 - 2\mu x + \mu^2) f(x)dx \\ &= \int_{-\infty}^{\infty} x^2 f(x)dx - 2\mu \int_{-\infty}^{\infty} x f(x)dx + \mu^2 \\ &= \int_{-\infty}^{\infty} x^2 f(x)dx - \mu^2 \\ &= E(X^2) - \mu^2, \end{aligned}$$

where $\mu = E(X)$.

To construct probability models for continuous r.v.'s, it is only necessary to find a density function that models appropriately the concentration of likelihood.

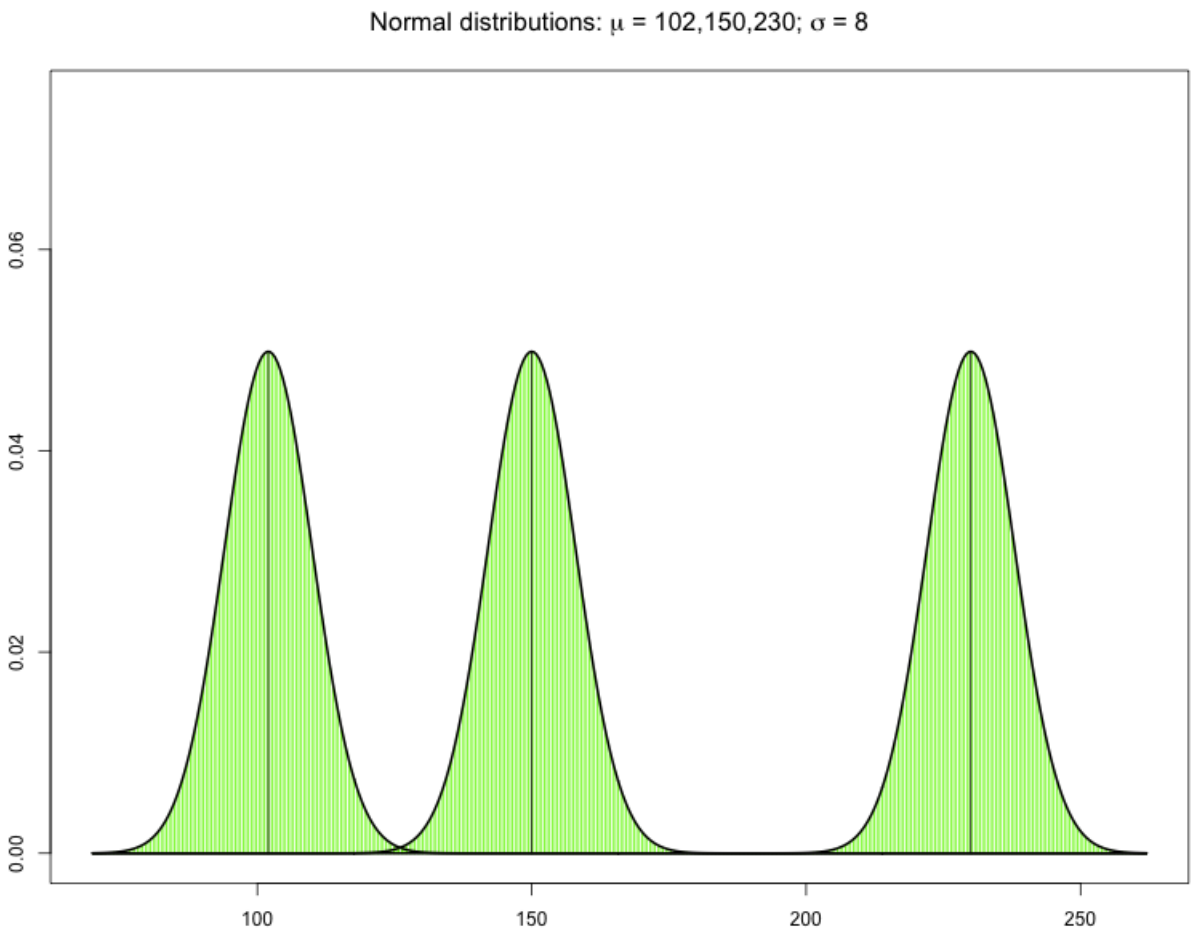
Normal Distribution

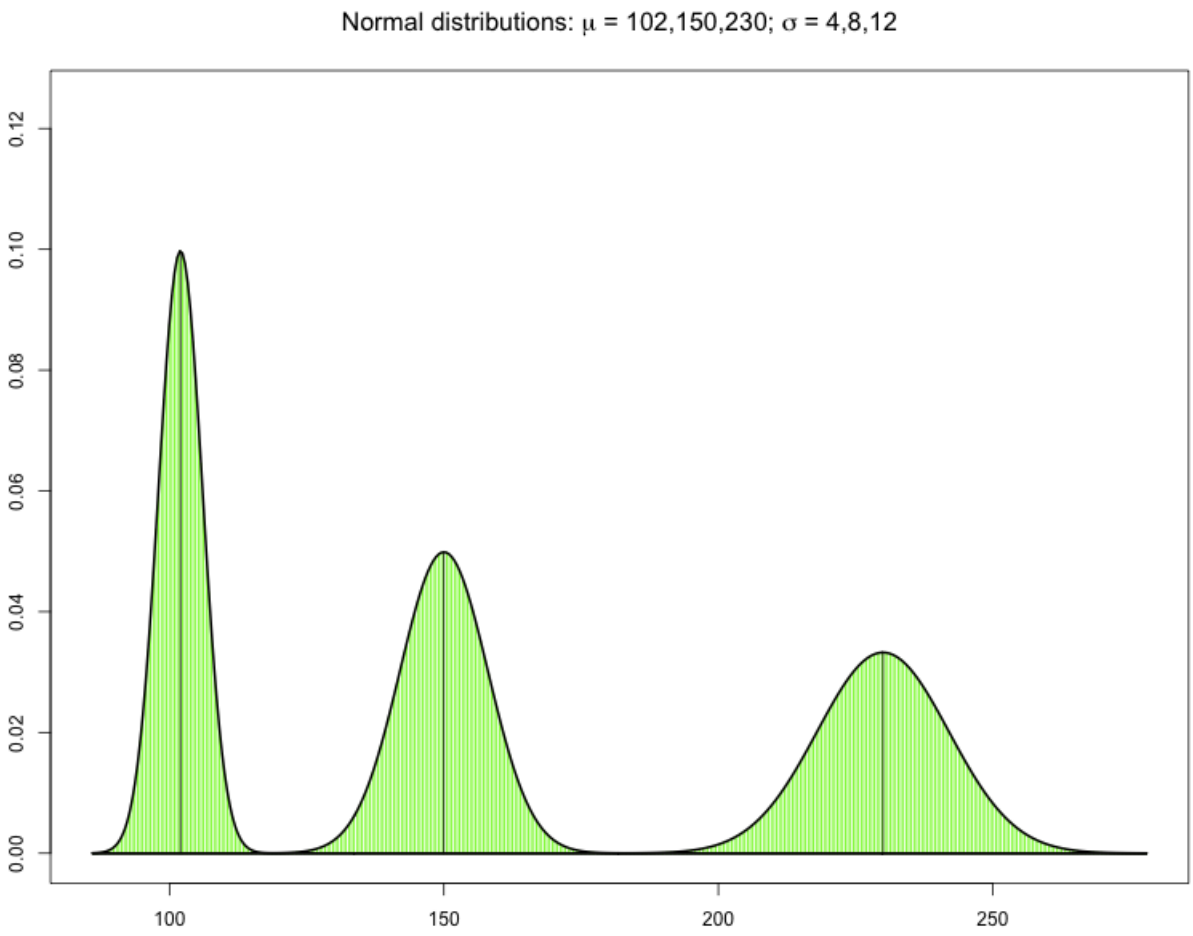
The **normal distribution**, also known as the *Bell Curve*, has been used (and abused) as a model for a wide variety of phenomena to the point that some have the impression that any data that does not fit this model is in some way *abnormal*. That is not the case. The name *normal distribution* comes from the title of the paper Carl Friedrich Gauss wrote that first described the mathematical properties of the bell curve, "On the Normal Distribution of Errors". For this reason, the distribution is sometimes referred to as the **gaussian distribution**. Perhaps that name would be less misleading. The main importance of this model comes from the central role it plays in the behavior of many statistics that are derived from large samples.

The normal distribution represents a family of distribution functions, parametrized by the mean and standard deviation, denoted by $N(\mu, \sigma)$. The density function for this distribution is

$$f(x; \mu, \sigma) = (2\pi\sigma^2)^{-1/2} \exp\{-(x-\mu)^2/(2\sigma^2)\}.$$

The mean is referred to as a location parameter since it determines the location of the peak of the curve. The standard deviation is referred to as a scale parameter since it determines how spread out or concentrated the curve is. The plots below illustrate these properties. In the first plot, the means differ but the standard deviations are all the same. In the second plot, both the means and the standard deviations differ.


Normal distributions: μ = 102,150,230; σ = 8
0.00
0.02
0.04
0.06
100
150
200
250


Normal distributions: μ = 102,150,230; σ = 4,8,12
0.00
0.02
0.04
0.06
0.08
0.10
0.12
100
150
200
250

Probability that a continuous random variable is contained within an interval is modeled by the area under the curve corresponding to the interval. Suppose for example we have a random variable that has a $N(50, 5)$ distribution and we are interested in the probability that this r.v. takes a value between 45 and 60. The problem now is to determine this area. Unfortunately (or perhaps fortunately from the point of view of students) the normal density function does not have an explicit integral. This implies that we must either use a set of tabulated values to obtain areas under the curve or use a computer routine to determine the areas. One property satisfied by the family of normal distributions is *closure under linear transformations*. That is, if $X \sim N(\mu, \sigma)$, and if $Y = a + bX$, then $Y \sim N(a + b\mu, |b|\sigma)$. We can make use of this property by noting that

$$Z = \frac{X - \mu}{\sigma} = -\frac{\mu}{\sigma} + \frac{1}{\sigma}X$$

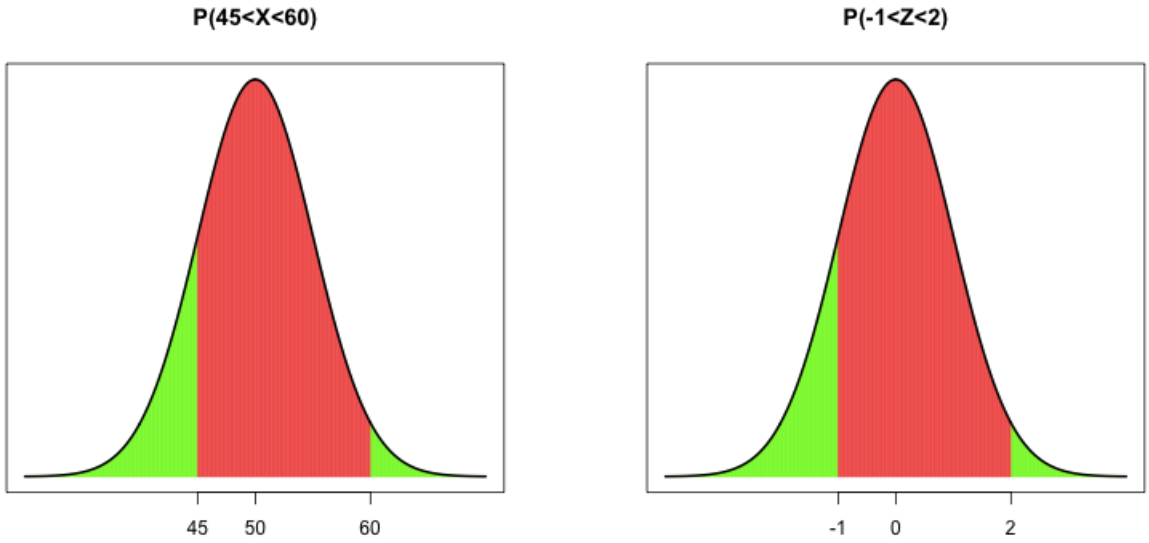
has a $N(0, 1)$ distribution. This distribution is referred to as the **standard normal distribution**, and the value of Z corresponding to X is referred to as the **standardized score** or **Z-score** for X. This property implies that the probability of any interval can be transformed into a probability involving the standard normal distribution. The interpretation of the *Z-score* can seen by expressing X in terms of Z,

$$X = \mu + Z\sigma.$$

This shows that the *z-score* represents the number of standard deviations X is from its mean.

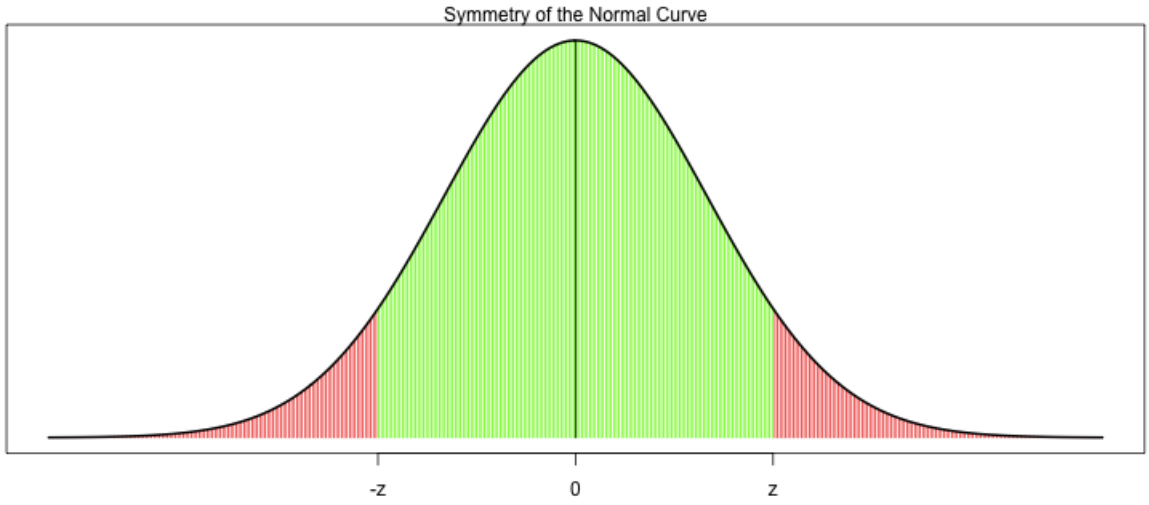
For example, if $X \sim N(50, 5)$, then

$$\begin{aligned} P(45 < X < 60) &= P(\frac{45-50}{5} < \frac{X-50}{5} < \frac{60-50}{5}) \\ &= P(-1 < Z < 2). \end{aligned}$$



As can be seen by comparing these two plots, the areas for $P(45 < X < 60)$ and $P(-1 < Z < 2)$ are the same. Therefore, it is only necessary to tabulate areas for the standard normal distribution. The textbook contains such a table on page 789. This table gives areas under the standard normal curve below z for $z > 0$. This table requires an additional property of normal distributions called symmetry:

$$P(Z < -z) = P(Z > z), \quad P(0 < Z < z) = P(-z < Z < 0).$$


Symmetry of the Normal Curve
-z
0
z

Example. Suppose a questionnaire designed to assess employee satisfaction with working conditions is given to the employees of a large corporation, and that the scores on this questionnaire are approximately normally distributed with mean 120 and standard deviation 18.

a) Find the proportion of employees who scored below 150.

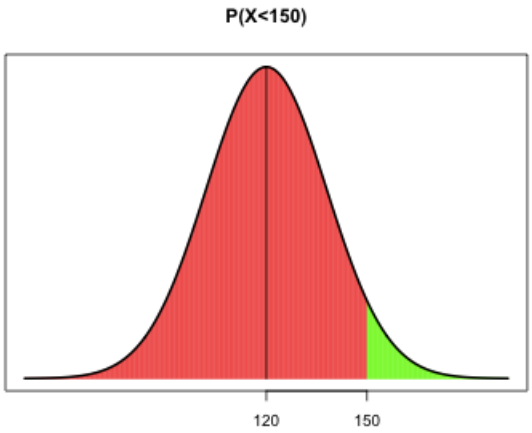
b) Find the proportion of employees who scored between 140 and 160.

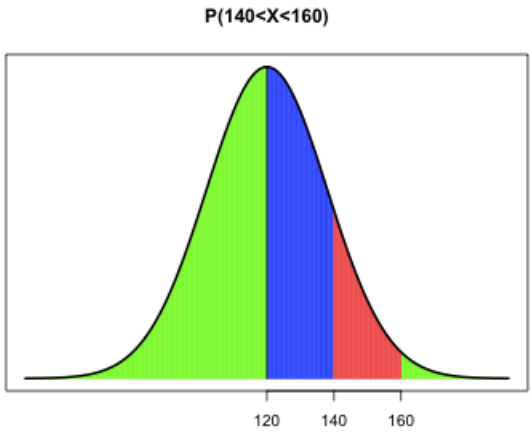
c) What proportion scored above 105?

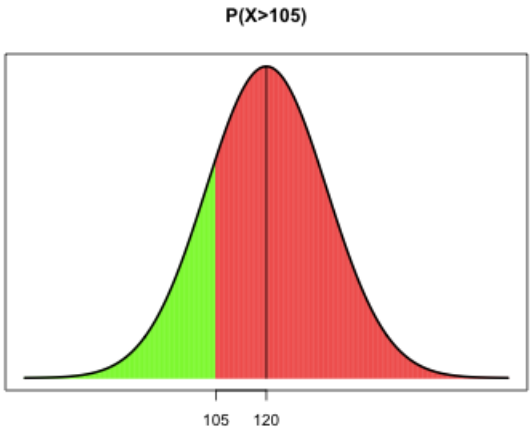
d) What proportion scored between 90 and 145?

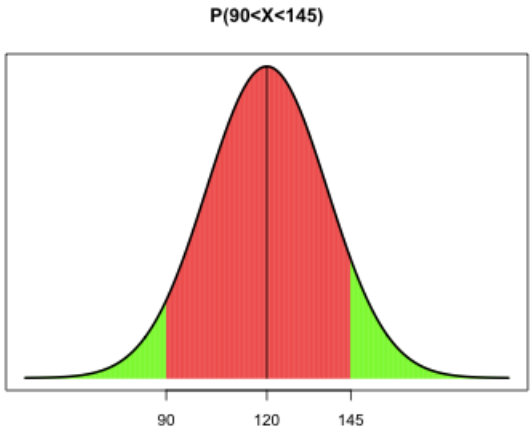
These areas are represented in the plots given below.

e) 15% of employees scored below what value?









Solutions

a) First transform to $N(0,1)$.

$$z = \frac{150 - 120}{18} = 1.67,$$

$$P(X < 150) = P(Z < 1.67).$$

From the table on the inside back cover of the text, the area below 1.67 is 0.9525. Therefore,

$$P(X < 150) = P(Z < 1.67) = 0.9525.$$

b) Transform to $N(0,1)$.

$$\begin{aligned} z_1 &= \frac{140 - 120}{18} = 1.11 \\ z_2 &= \frac{160 - 120}{18} = 2.22. \end{aligned}$$

In this case we must subtract the area below 1.11 from the area below 2.22. From the table these areas are, respectively, .8665 and .9868. This gives

$$P(140 < X < 160) = P(1.11 < Z < 2.22) = 0.9868 - 0.8665 = 0.1203.$$

c) Transform to $N(0,1)$.

$$z = \frac{105 - 120}{18} = -0.83.$$

The symmetry property of the normal distribution implies that the area above -0.83 is the same as the area below 0.83, which we get from the table.

$$P(X > 105) = P(Z > -0.83) = P(Z < 0.83) = 0.7967.$$

d) Transform to $N(0,1)$.

$$\begin{aligned} z_1 &= \frac{90 - 120}{18} = -1.67 \\ z_2 &= \frac{145 - 120}{18} = 1.39 \end{aligned}$$

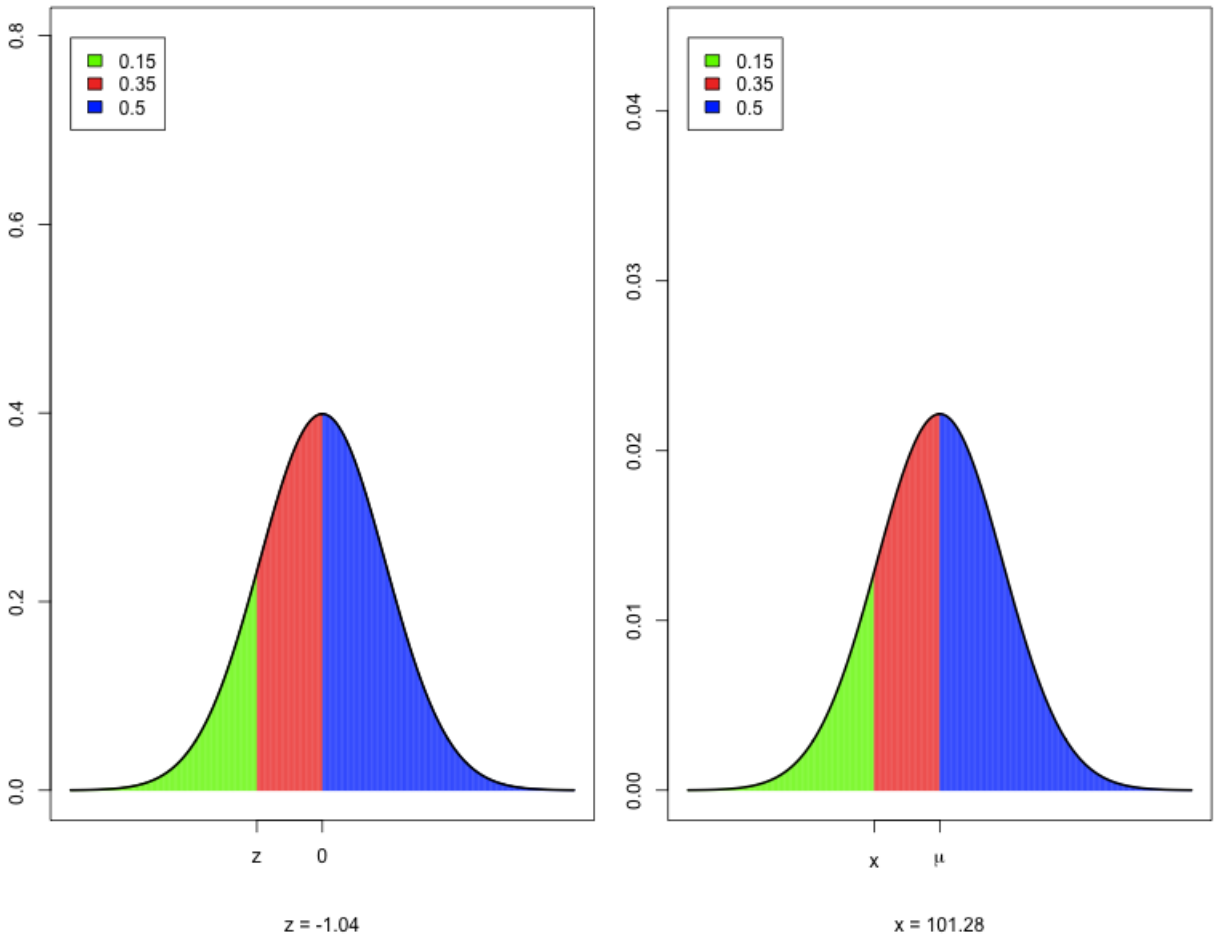
The area we require is the difference between the area below 1.39 and the area below -1.67. By symmetry, the area below -1.67 is the same as the area above 1.67.

$$\begin{aligned} P(90 < X < 145) &= P(Z < 1.39) - P(Z < -1.67) \\ &= 0.9177 - [1 - P(Z < 1.67)] \\ &= 0.9177 - [1 - 0.9525] \\ &= 0.8702. \end{aligned}$$

e) This problem is different than the others because we are given an area and must use this to determine the appropriate value. The first step is to determine on which side of the mean the required value is located. This is determined by two quantities: whether the area is less than 0.5 or greater than 0.5, and the direction relative to the required value occupied by the specified area. In this case, the area (**15%=0.15**) is less than 0.5 and the direction is specified by *scored* **below** *what value*. These imply that the required value must be less than the mean. A picture of this area is given below. To answer this question, we first answer the corresponding question for the standard normal distribution. What *z-value* has an area of **0.15 below** it? This *z-value* must be negative since the area is less than **0.15** and the direction is **below** (or to the left of) the required value. Since the table gives areas below **z**, the area we must find in the table is $1 - 0.15 = 0.85$. The closest area in the table to 0.85 is 0.8508 which corresponds to a z-score of 1.04. Since the z-score for this problem is negative, then the answer to this question for the standard normal distribution is $z = -1.04$. Finally, we must convert this z-score to the x-value,

$$x = \mu + z\sigma = 120 + (-1.04)(18) = 101.28.$$

If you check this answer by finding the area below 101.28, you will see that the steps we just followed are the same steps we used to find areas but applied in reverse order. Also note that the value of 101.28 represents the 15^{th} percentile of this normal distribution. Other percentiles can be obtained similarly.



Since z-scores represent the number of standard deviations from the mean, and since they are directly associated with percentiles, they can be used to determine the relative standing of an observation from a normally distributed population. In particular, consider the following three intervals: $\mu \pm \sigma$, $\mu \pm 2\sigma$, and $\mu \pm 3\sigma$. After converting these intervals to z-scores, they become, respectively, (-1,1), (-2,2), and (-3,3). Because of the symmetry property, the probabilities for these intervals are,

$$\begin{aligned} P(\mu - \sigma < X < \mu + \sigma) &= P(-1 < Z < 1) = 2P(0 < Z < 1) = 2(.3413) = .6826 \\ P(\mu - 2\sigma < X < \mu + 2\sigma) &= P(-2 < Z < 2) = 2P(0 < Z < 2) = 2(.4772) = .9544 \\ P(\mu - 3\sigma < X < \mu + 3\sigma) &= P(-3 < Z < 3) = 2P(0 < Z < 3) = 2(.4987) = .9974 \end{aligned}$$

This is the basis for the **empirical rule**: if a set of data has a histogram that is approximately bell-shaped, then approximately 68% of the measurements are within 1 standard deviation of the mean, approximately 95% are within 2 standard deviations of the mean, and essentially

all (makes more sense than approximately 99.74%) are within 3 standard deviations of the mean.

Suppose that in the previous example an employee scored 82 on the employee satisfaction survey. The z-score for 82 is (82-120)/18 = -2.11. So this score is more than 2 standard deviations below the mean. Since 95% of the scores are within 2 standard deviations of the mean, this is a relatively low score. We could be more specific by determining the percentile rank for this score. From the table of normal curve areas, the area below 2.11 is 0.9826, so the area below $z = -2.11$ is $1 - 0.9826 = 0.0174$. That is, only 1.74% of those who took this questionnaire scored this low or lower.

Large Sample Approximations

The main importance of the normal distribution is associated with the **Central Limit Theorem**. This theoem was originally derived as a large sample approximation for the binomial distribution when n is large and p is not extreme. In this case we may approximate the binomial distribution function by the normal distribution with mean np and standard deviation $\sqrt{np(1-p)}$.

Suppose for example that in a very large population of voters, 48% favor Candidate A for president, and that a sample of 500 is randomly selected from this population. What is the probability that more than 250 in the sample will favor Candidate A? We can model the number in the sample who favor Candidate A with a binomial distribution with *n=500* and *p=0.48*. Since n is large, we can approximate this distribution with a normal distribution with mean $\mu = 500(.48) = 240$ and standard deviation $\sigma = \sqrt{500(.48)(.52)} = 11.2$. Since the binomial is a discrete distribution, we can improve this approximation slightly by extending the interval of values whose probability we wish obtain by 0.5 at each end of the interval. For example, if we want to find $P(N = 230)$, then we approximate it by $P(229.5 < X < 230.5$, where X has the appropriate approximate normal distribution. Similarly,

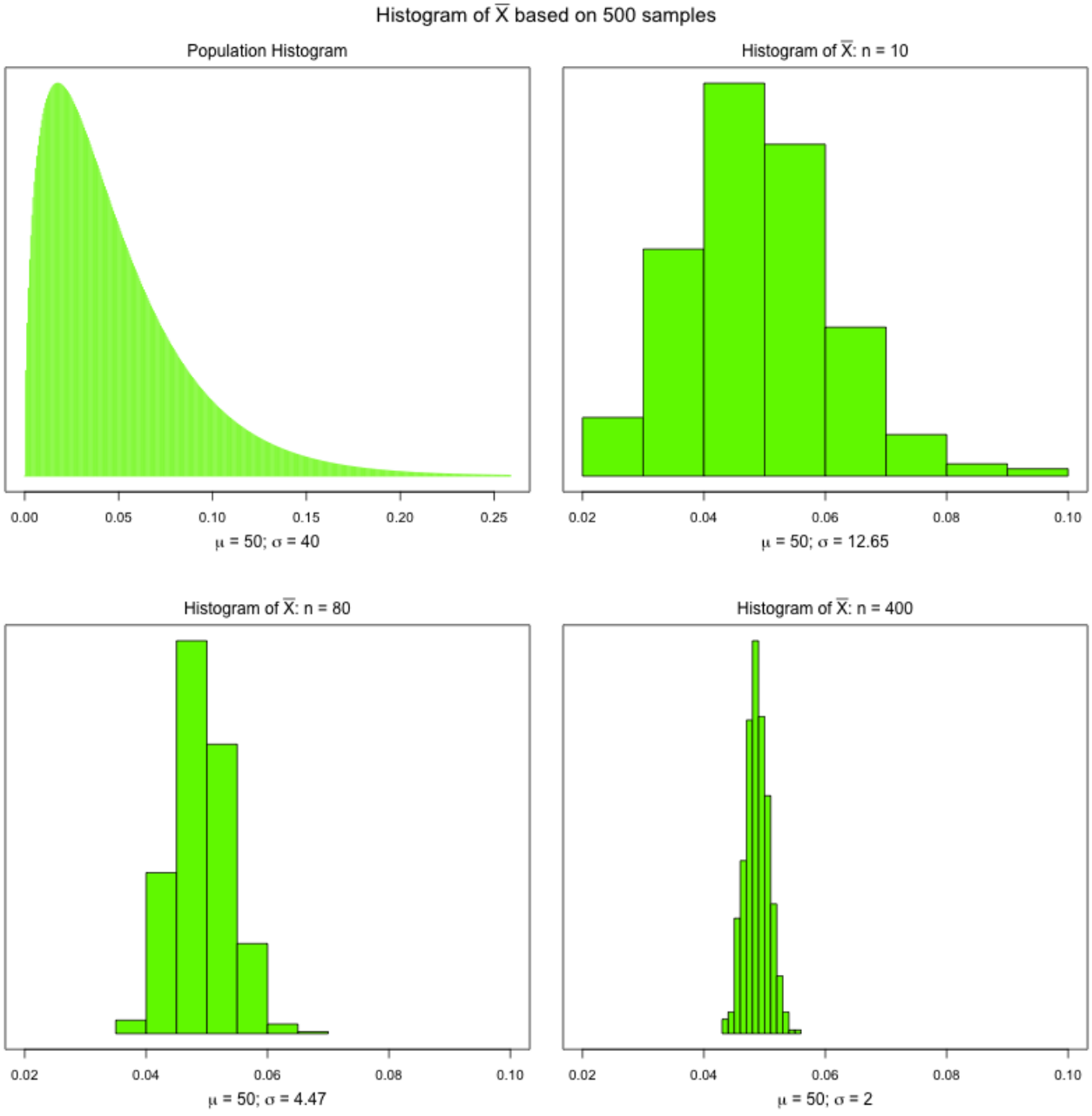
$$\begin{aligned}
P(N < a) &\approx P(X < a - .5) \\
P(N \leq a) &\approx P(X < a + .5) \\
P(N > a) &\approx P(X > a + .5) \\
P(N \geq a) &\approx P(X > a - .5) \\
P(a < N < b) &\approx P(a + .5 < X < b - .5) \\
P(a \leq N \leq b) &\approx P(a - .5 < X < b + .5)
\end{aligned}$$

Therefore, from the table of areas under the normal curve, we obtain

$$\begin{aligned}
P(N > 250) &\approx P(X > 250.5) \\
&= P(Z > (250.5 - 240)/11.2) \\
&= P(Z > 0.94) \\
&= = 1 - 0.8264 = 0.1736.
\end{aligned}$$

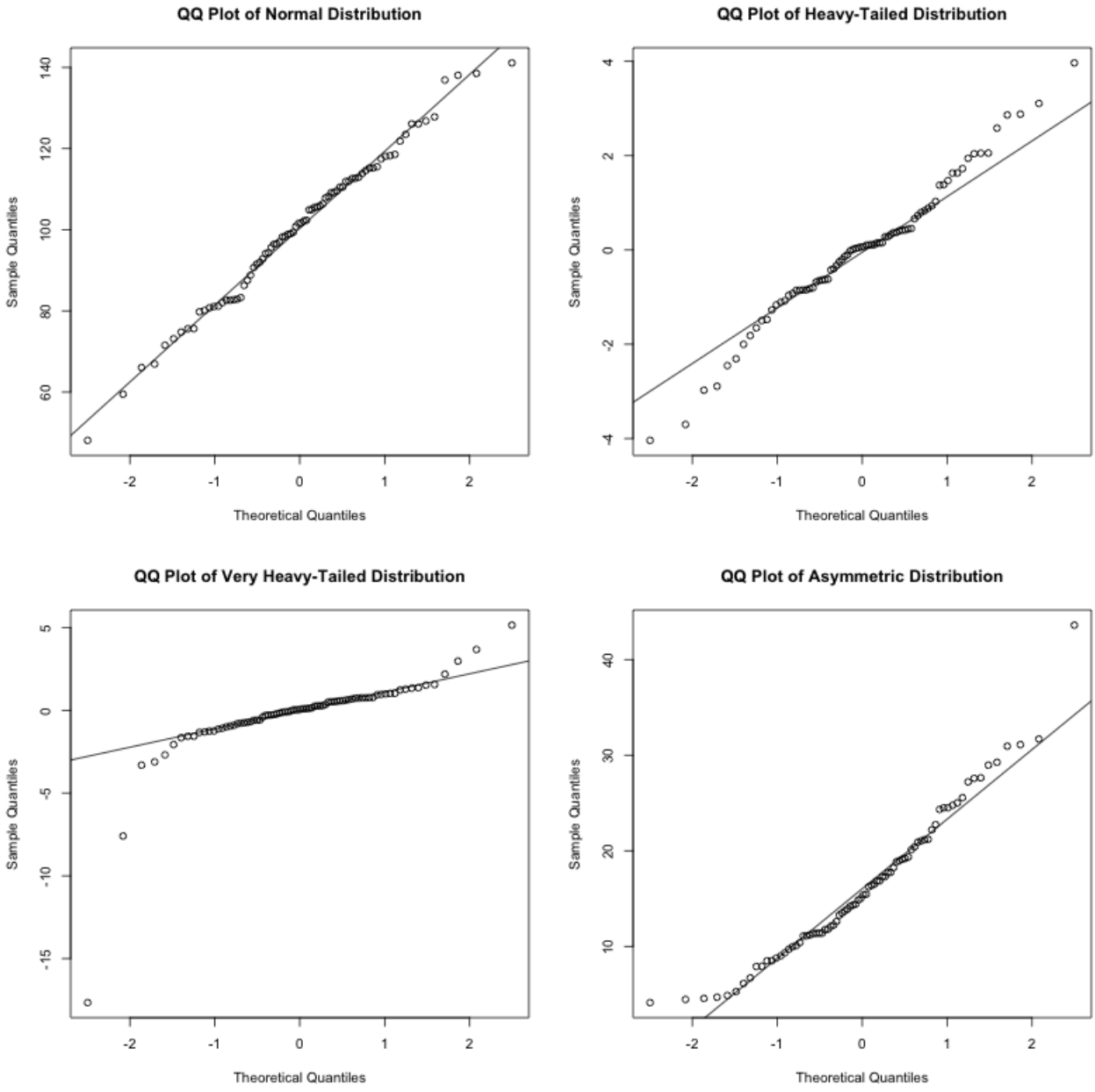
Note that we could also express this event in terms of the sample proportion who favor Candidate A. Let $\hat{p} = N/500$ denote the sample proportion. Then the probability we obtained above could be expressed as $P(\hat{p} > 0.5)$. Since $\hat{p}$ is a linear function of N, then we can use the normal distribution with mean $\mu = 240/500 = 0.48$ and standard deviation $\sigma = 11.2/500 = 0.022$ to approximate the distribution of $\hat{p}$. Note that the standard deviation can be obtained directly as $\sigma = \sqrt{p(1-p)/n} = \sqrt{(.48)(.52)/500} = 0.022$.

The **Central Limit Theorem** extends this result to a sampling situation in which a sample of size n is randomly selected from a very large population with mean μ and standard deviation σ. Let $\overline{X}$ denote the mean of this sample. We can treat the sample mean as a random variable that is the numerical value associated with the particular sample we obtain when we perform the sampling experiment. The *Central Limit Theorem* states that the distribution of this random variable is approximately a normal distribution with mean μ and standard deviation $\sigma/\sqrt{n}$. Suppose we looked at every possible sample of size n that could be obtained from the population, and we computed the sample mean for each of these samples. What the **CLT** implies is that the histogram of all these sample means would be approximately a normal curve with mean μ and standard deviation $\sigma/\sqrt{n}$. The following plots illustrate this.



Note that there is less asymmetry in the histogram of $\overline{X}$ with $n = 10$ than in the population histogram, but some asymmetry still remains. However, that asymmetry is not present in the histograms corresponding to the larger sample sizes. Note also that the variability decreases with increasing sample size. This theorem holds for any distribution, but the more *non-normal* the distribution, the larger n must be for the distribution of $\overline{X}$ to be close to the normal distribution. However, if the population distribution is itself a normal distribution, then the Central Limit Theorem holds for all $n \geq 1$.

One remaining question that will also be applicable to methods discussed later is the problem of determining how far a data set is from normality. This is accomplished most commonly by a *Quantile-Quantile* plot. Let n denote the sample size and let $y = ((1 : n) - .5)/n$. Then y represents the quantiles of the ordered values of the data. That is, $y[i]$ represents, up to a correction factor, the proportion of the sample that is at or below the i^{th} ordered value of the sample. Now let $x[i] = z_{y[i]}$. Then $x[i]$ represents the z-score such that the area below it equals the proportion of the sample that is at or below the data value corresponding to the i^{th} ordered value. If the data has a normal distribution, then these points should fall on a line with slope equal to the s.d. and intercept equal to the mean. The following plots show quantile-quantile plots for four distributions: normal, heavy-tailed, very heavy-tailed, and asymmetric.



Homework and Project Assignments

Homework assignments can be submitted to me by email. Please do not send Word documents. Instead, save it as pdf and send the pdf file with the subject line *stat3355 homework*.

Homework 1

Due date: Sept. 12. Homework assignments should be submitted to me by email. Please do not send a Word document. Instead, save the Word file as pdf and send the pdf file. Put *stat3355 hw1* on the subject line.

1. Use the `HairEyeColor` data set to obtain the joint frequency table for eye color and sex. Find the following percentages and express each as a probability statement.

   [**a**] The percentage of the group that are male.
   [**b**] The percentage of the group with green eyes.
   [**c**] The percentage of males with green eyes.
   [**d**] The percentage of those with green eyes who are male.
   [**e**] The percentage of those with brown eyes who are female.

2. `HairEyeColor` continued.
   [**a**] Obtain the expected frequencies under the assumption of independence for each combination of eye color and sex, the distance from independence for each combination, and the total distance of this frequency table from independence.
   [**b**] Construct a barplot that shows the relative proportions of eye color within sex categories.
   [**c**] Construct an informative barplot that shows the relative proportions of males and females within eye colors.

3. Use the data contained in the file
   `http://www.utdallas.edu/~ammann/stat3355scripts/Smoking.txt`

   [**a**] Find the means and standard deviations for each variable.
   [**b**] Which states are more than 2 sd's above the mean for cigarette consumption? for bladder cancer? for lung cancer?
   [**c**] Which states are in the top 10% of cigarette consumption? of bladder cancer? of lung cancer? (see documentation for **R** function *quantile()*)
   [**d**] Plot cigarette consumption versus lung cancer and add an informative title. Be sure to think about which variable should be plotted on the Y-axis. Do the same for cigarette consumption versus bladder cancer.