

# Statistics with R

## *Importing Data Sets and Tables*

Scott Hetzel

University of Wisconsin – Madison

Summer Institute for Training in Biostatistics (2009)

Derived from: “Introductory Statistics with R” by: Peter Dalgaard

and from previous notes by Deepayan Sarkar, Ph.D

# What we Discussed Last

---

- Indexing vectors and matrices is done with "[ ]" after the object's name. Indexing can be done with positive integers, negative integers, and conditionally.
- Logical operators in **R** are as follows `<`, `>`, `>=`, `<=`, `==`, `!=`, `&`, `|`
- We can modify objects using indexing and logical formulas
- We can add new columns to a data frame with `data.F$name <- vector`
- We can subset data frames with `subset( )`
- We can place a data frame in the search path of **R** with `attach( )`
- **R** is its own programming language and therefore can do for loops, if/then statements, and while loops.

# Importing Data Sets into R

---

The most convenient way of reading data into **R** is via the function called `read.table`. The two main arguments for `read.table` are the file name path and whether or not the first row of the txt file are variable names. Any missing values in the original data file needs to be represented by an 'NA' in those places.

```
> dat <- read.table("/u/hetzel/private/SIBS/example.txt", header=T)
```

After using `read.table`, `dat` will be a data frame in **R**. But will need to be attached if you want to call the variable names without using the `$` operator.

Other similar functions will work for different types of files you may want to import into **R**. Look at `help.search("read")` for a list of functions that can be applied to different data files. The package `foreign` has a bunch of functions that can be used to import data files from other statistical programs, such as SPSS, SAS, Stata, etc.

# Importing Data Sets into R (Cont.)

---

If your file is already in the working directory for R then you can shorten the first argument to just the name of the file.

To check what the working directory path is, type in `getwd()` and hit enter. My path at work might look like this.

```
> getwd()  
[1] "/w/w00/h/hetzel/private/SIBS"
```

If you would like to change the working directory to a different folder path the function `setwd()`. On a Windows operating system it could look like this:

```
> setwd("C:/Documents and Settings/Scott/MyDocuments/SIBS")
```

# Random Sampling

---

Probability theory was developed mostly due to games and gambling issues. The basic notion was that of a random sample such as, dealing from a shuffled deck of cards or picking balls out of an urn.

In **R** you can simulate these situations with the `sample` function.

```
> sample(1:40, 5)
[1] 29 6 20 5 30
```

The default is to sample without replacement. If you want to pick and replace before the next pick then the argument `replace` must be set to `TRUE`

```
> sample(c("H","T"), 10, replace=T)
[1] "H" "H" "H" "H" "T" "H" "H" "H" "H" "T"
```

Another default is to treat each possible selection to have equal probability of being selected at each draw. If you want to have a "weighted" die the `prob` argument can be modified.

```
> sample(c(1,2,3,4,5,6), 8, replace=T, prob=c(.1,.1,.1,.5,.1,.1))
[1] 5 4 3 4 3 2 4 4
```

# Distributions in R: Binomial

---

We have discussed a bit about the different types of distributions that are available in R. Two we will discuss a little further are the binomial distribution and normal distribution and how we can use them to answer probability questions.

Assumptions of the Binomial Distribution are:

- The Binomial is the total count of Successes after  $n$  trials.
- Each trial only has two possible outcomes: Success or Failure
- Probability of Success ( $p$ ) is constant from trial to trial
- Trial outcomes are independent of each other
- `dbinom(x, size, prob)` gives the  $P(X=x)$  for certain size and probability. A vector of  $x$ 's will return a vector of corresponding probabilities.
- `pbinom(q, size, prob)` gives the  $P(X \leq x)$ , if argument `lower.tail = FALSE` then it is  $P(X > x)$ .
- `rbinom(n, size, prob)` gives randomly generated binomial outcomes.

# Exercises in Using R

---

According to a National Health Survey, 9.8% of the population of 18- to 24- year-olds in the United States are left-handed. Solve the following probability questions:

1. What is the probability that exactly 3 of 10 people randomly selected from the population are left-handed?
2. What is the probability that at least 6 of the 10 people are left-handed?
3. What is the probability that at most two individuals are left-handed?
4. Produce 50, 500, and 5000 random binomial outcomes from a setup similar to the one above: 10 trials and probability of success = 0.098. Calculate the mean for each and compare to the expected value of this Binomial.

# Exercises in Using R Answers

---

1. 

```
> dbinom(3, 10, 0.098)
[1] 0.05486624
```
2. 

```
> pbinom(5, 10, 0.098, lower.tail=FALSE)
[1] 0.0001311050
```
3. 

```
> pbinom(2, 10, 0.098)
[1] 0.9332107
```
4. 

```
> mean(rbinom(50, 10, 0.098))
[1] 0.94
> mean(rbinom(500, 10, 0.098))
[1] 0.992
> mean(rbinom(5000, 10, 0.098))
[1] 0.978
```

Expected value is  $n * p$  which is 0.98. The more samples the better the approximation.

# Distributions in R: Normal

---

We looked at an important discrete distribution, now we will look at a very important continuous distribution. Most inference testing we will talk about later has an underlying assumption of normality.

In R the following functions apply to the normal distribution:

- `> dnorm(x, mean, sd)` Gives the density value from the density function on page 49 of the text. Not too useful.
- `> pnorm(q, mean, sd)` Gives  $P(X \leq x)$  and with argument `lower.tail = FALSE` gives  $p(X > x)$
- `> qnorm(p, mean, sd)` Gives the  $x$  value that satisfies  $P(X \leq x) = p$  for given  $p$ . If argument `lower.tail = FALSE` then it gives the  $x$  value that satisfies  $P(X > x) = p$  for given  $p$ .
- `> rnorm(n, mean, sd)` Creates  $n$  random draws from a specified normal distribution.

# Graphic Check of Normality

---

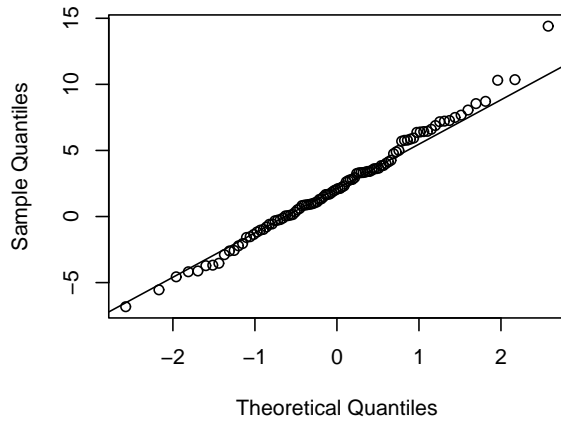
To see whether data can be assumed normally distributed we check the Q-Q Plot with the function `qqnorm()`. The function plots the  $k$ 'th smallest observation against the expected value of the  $k$ 'th smallest observation out of  $n$  in a standard normal distribution and it works for normal distributions with *any* mean and standard deviation. The data points should follow a straight line. To plot the straight line the points should follow use `qqline()`

```
> a <- rnorm(100, 2, 4)
> b <- rbinom(100, 20, 0.4)
> c <- runif(100, 0, 10)
> d <- rchisq(100, 4)
```

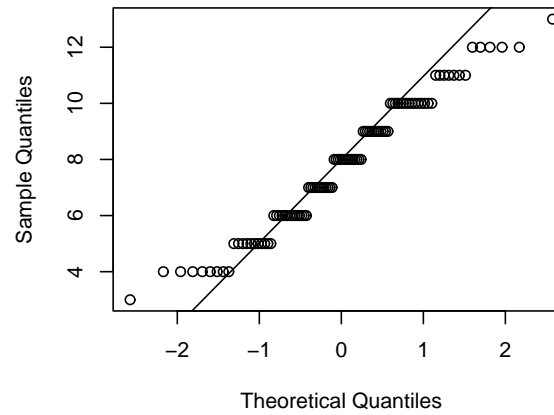
```
> qqnorm(a)
> qqline(a)
> qqnorm(b)
> qqline(b)
> qqnorm(c)
> qqline(c)
> qqnorm(d)
> qqline(d)
```

# Graphic Check of Normality (Cont.)

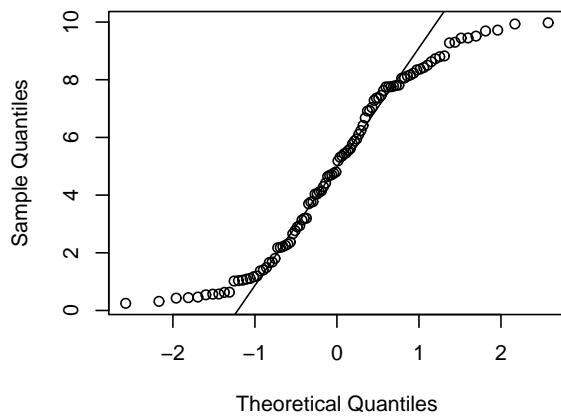
Normal Q-Q Plot



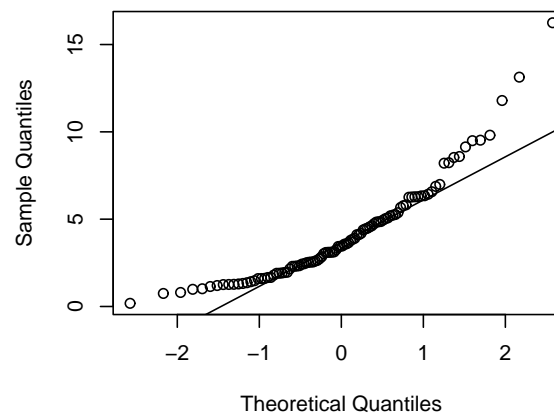
Normal Q-Q Plot



Normal Q-Q Plot



Normal Q-Q Plot



# Exercises in Using R

---

The distribution of weights for the population of males in the United States is approximately normal with mean  $\mu = 172.2$  pounds and standard deviation  $\sigma = 29.8$  pounds.

1. What is the probability that a randomly selected man weighs less than 130 pounds?
2. What is the probability that he weighs more than 210?
3. What weight cuts off the top 25% of the population?
4. What is the probability that among 5 males selected at random from the population, at least one will have a weight between 170 and 200 pounds?

# Exercises in Using R Answers

---

1. `> pnorm(130, 172.2, 29.8)`

```
[1] 0.07837203
```

2. `> pnorm(210, 172.2, 29.8, lower.tail=FALSE)`

```
[1] 0.1023175
```

3. `> qnorm(0.25, 172.2, 29.8, lower.tail=FALSE)`

```
[1] 192.2998
```

4. Two part question, use normal to get probability, then binomial to answer question.

```
> p <- pnorm(200, 172.2, 29.8) - pnorm(170, 172.2, 29.8)
```

```
> p
```

```
[1] 0.3539859
```

```
> 1 - dbinom(0, 5, p) # pbinom(0, 5, p, FALSE) would work too
```

```
[1] 0.8874852
```

# Generating Tables in R

---

We have already discussed about having data in a tabular format, more specifically a matrix. To do this we had to specify the data to be entered in the matrix by a vector and the number of rows and/or columns by `nrow()` or `ncol()`. Furthermore we had to manually give the row and columns names through `rownames()` and `colnames()`. More often we will have a data set where we might want to summarize factor level frequencies in a tabular format. The `table()` function can take care of this.

For example: Say a windy day is a day with greater than 12 mph wind. Using the `airquality` data, produce a table of the number of windy and non-windy days in each month.

```
> Status <- airquality$Wind > 12 # Saves Status as logical vector
> airquality$Status <- factor(Status, levels=c(TRUE, FALSE),
+ labels=c("Windy", "Calm")) # Creates new column in airquality called Status
> windTab <- table(airquality$Status, airquality$Month)
> windTab
```

	5	6	7	8	9
Windy	11	7	4	4	8
Calm	20	23	27	27	22

# Tables (Cont.)

---

Marginal tables can be produced through `margin.table`.

```
> margin.table(windTab,1)
Windy Calm
   34   119
```

```
> margin.table(windTab,2)
   5   6   7   8   9
31  30  31  31  30
```

If you want a table of the proportion of each row or column, use `prop.table`

```
> prop.table(windTab,1)
           5           6           7           8           9
Windy 0.3235294 0.2058824 0.1176471 0.1176471 0.2352941
Calm  0.1680672 0.1932773 0.2268908 0.2268908 0.1848739
```

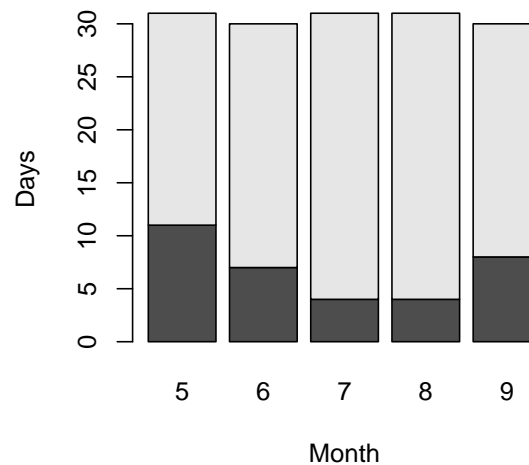
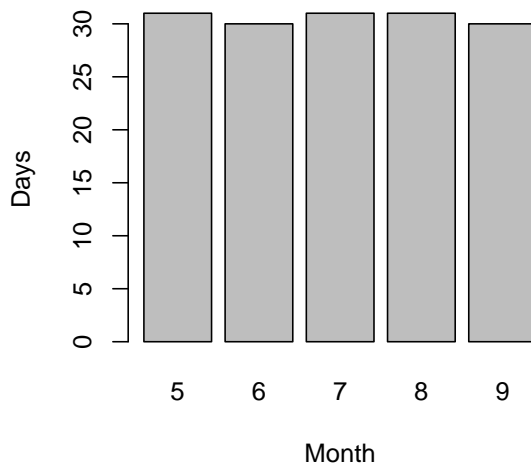
```
> prop.table(windTab,2)
           5           6           7           8           9
Windy 0.3548387 0.2333333 0.1290323 0.1290323 0.2666667
Calm  0.6451613 0.7666667 0.8709677 0.8709677 0.7333333
```

# Graphical Representation of Tables

---

For presentation purposes, it may be desirable to display a graph rather than a table of counts or percentages. Bar plots are made using `barplot`. This function takes an argument of data, which can be a vector or a matrix.

```
> layout(matrix(c(1,2), nrow=1)) # Two plots will be placed in the same window  
> barplot(margin.table(windTab, 2), xlab="Month", ylab="Days")  
> barplot(windTab, xlab="Month", ylab="Days")
```

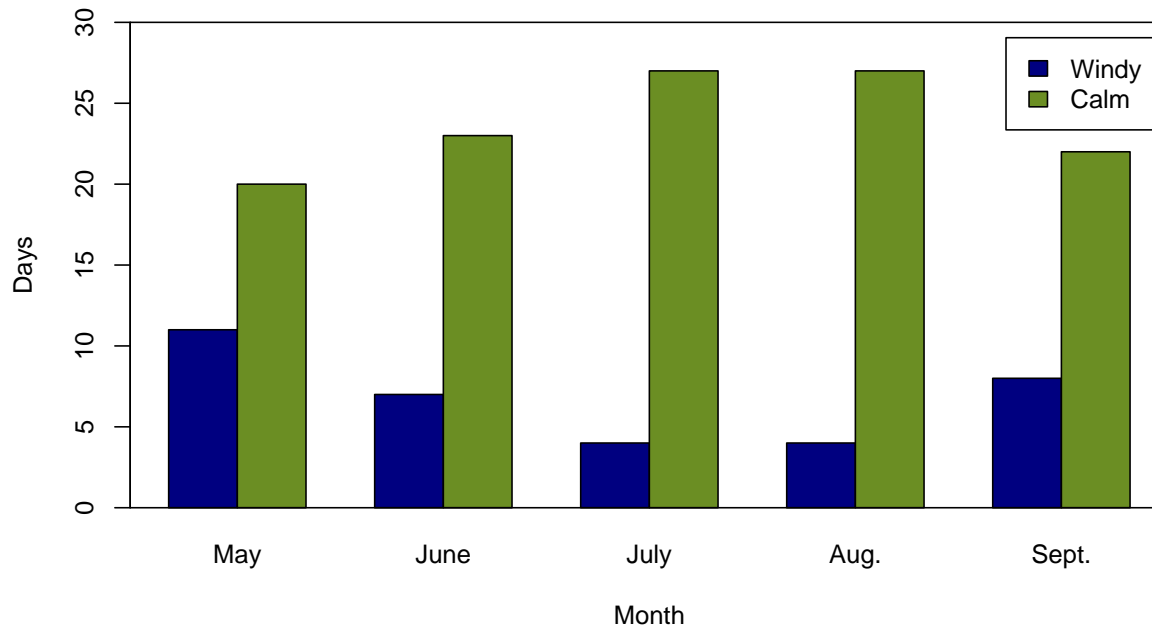


# Bar Plots (Cont.)

---

There are plenty of arguments to help improve the bar plot to make it look more presentable.

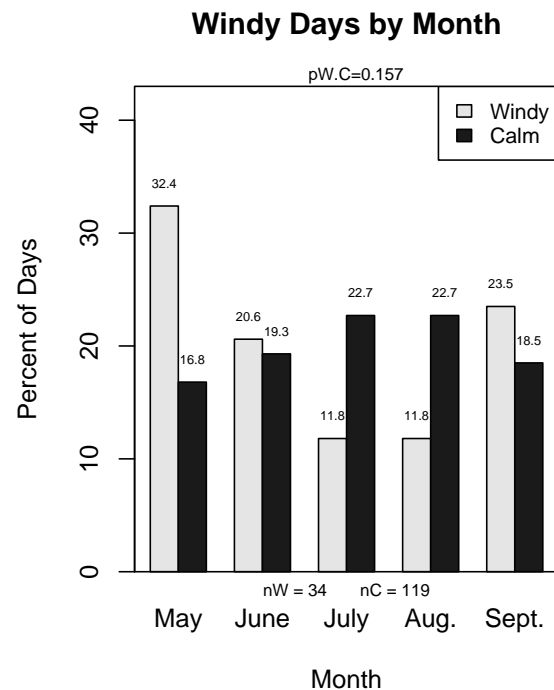
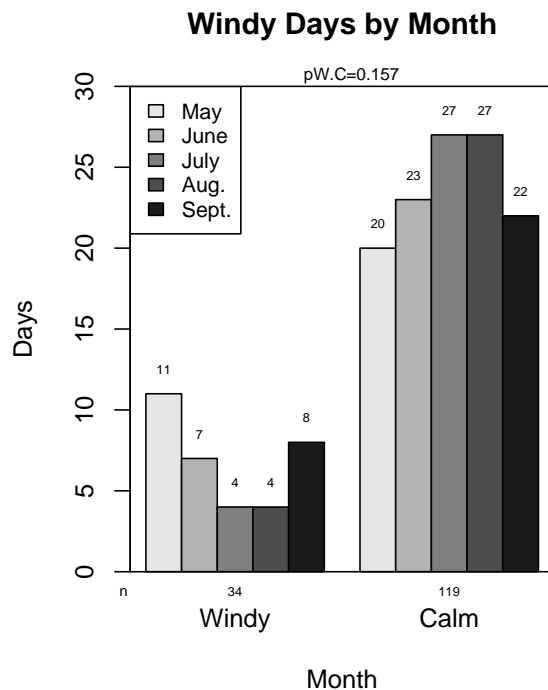
```
> layout(1) # Need to reset layout of window
> barplot(windTab, xlab="Month", ylab="Days", beside=T,
+ legend.text=c("Windy", "Calm"), col=c("navy", "olivedrab"),
+ ylim=c(0,30), names.arg=c("May", "June", "July", "Aug.", "Sept. "))
> box()
```



# UW Bar Plot functions

```
> par(mfrow=c(1,2)) # Same as the layout function before
> uwBarPlot(airquality, catFactorName="Month", TrxName="Status", LegendLoc="topleft",
+ catFactorNames=c("May","June","July","Aug.,""Sept."), yLab="Days", xLab="Month",
+ pTitle="Windy Days by Month", Percent=FALSE, axLim=c(0,30))

> uwDemoBarPlot(airquality, catFactorName="Month", TrxName="Status", xLab="Month",
+ yLab="Percent of Days", catFactorNames=c("May","June","July","Aug.,""Sept."),
+ pTitle="Windy Days by Month", NumDec=1)
```



# Exercises in Using R

---

Again looking at the Motor Trend car road tests data set `mtcars` conduct the following:

1. Create a table showing the number of cars matched by number of cylinders and number of forward gears.
2. Using indexing on the table, how many cars have 8 cylinders and 3 gears?
3. What percentage of 4 cylinder cars have 4 gears?
4. What percentage of 5 geared cars have 6 cylinders?
5. Create a bar plot figure showing the breakdown of number of cylinders by whether the cars is an automatic or manual transmission.
  - Give the plot a title
  - Label the x- and y-axis appropriately.
  - Make sure the axes are wide enough
  - Create a legend
  - Make sure the bars are not stacked together
  - Make the bars be horizontal.
  - Place a box around the plot

# Exercises in Using R Answers

---

1. `> tab <- table(mtcars$cyl, mtcars$gear)`

```
> tab
      3   4   5
4     1   8   2
6     2   4   1
8    12   0   2
```

2. `> tab[3,1]`

```
[1] 12
```

3. `> propTab <- prop.table(tab,1)`

```
> propTab
      3           4           5
4 0.0909091 0.7272727 0.1818182
6 0.2857143 0.5714286 0.1428571
8 0.8571429 0.0000000 0.1428571
> propTab[1,2]
[1] 0.7272727
```

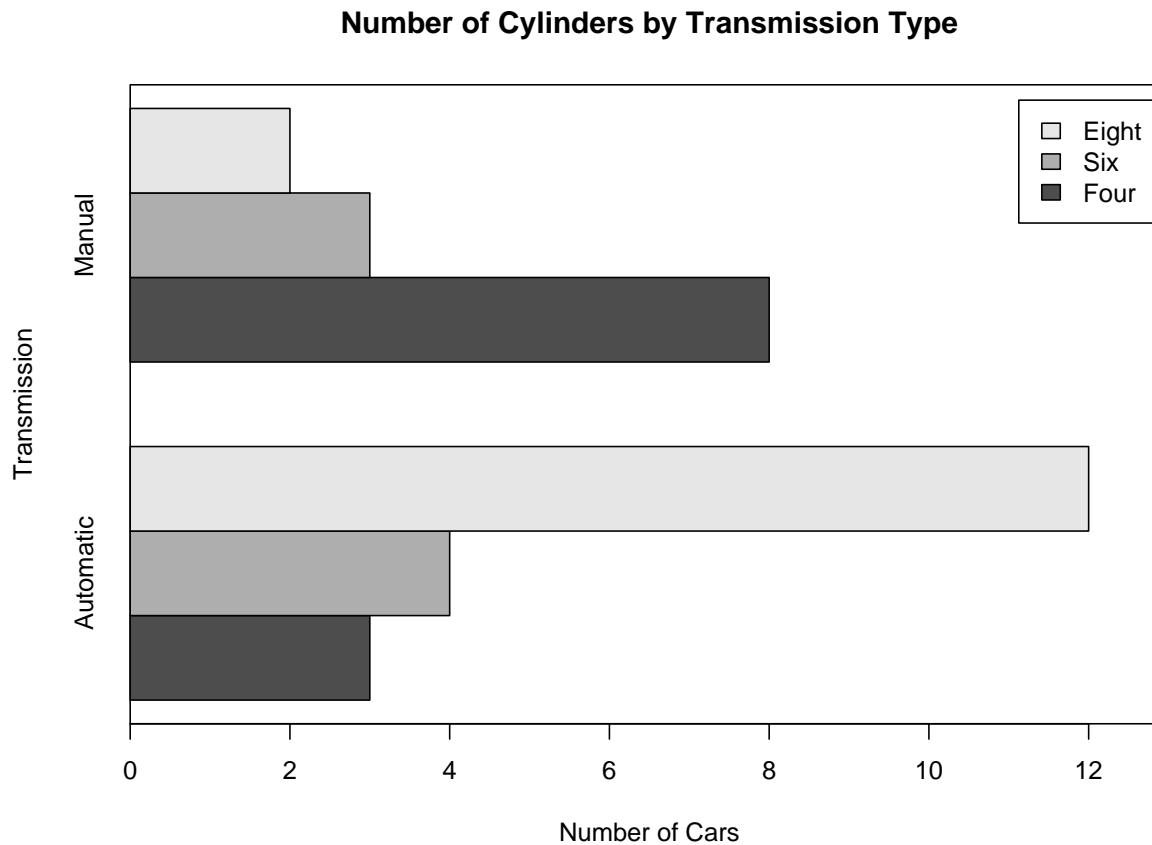
4. `> propTab2 <- prop.table(tab,2)`

```
> propTab2
      3           4           5
4 0.06666667 0.66666667 0.40000000
6 0.13333333 0.33333333 0.20000000
8 0.80000000 0.00000000 0.40000000
> propTab2[2,3]
[1] 0.20000000
```

# Exercises in Using R Answers

---

```
> tab.cyl.am <- table(mtcars$cyl, mtcars$am)
> barplot(tab.cyl.am, main="Number of Cylinders by Transmission Type",
+ horiz=T, xlab="Number of Cars", ylab="Transmission", beside=T, xlim=c(0,13),
+ legend=c("Four", "Six", "Eight"), names=c("Automatic", "Manual"))
```



# Exercises in Using R

---

- The scientists studying the iris data set think the Sepal Width measurement subtracted from the Sepal Length and the product of the petal measurements could help identify the species of an unknown flower.
- Add these two new variables to the data.
- Create two plots to see if the two variables help in determining species. Make sure the two plots are side by side in the same window.

Hint: Use `layout()` or `par()` to get the window right.

Hint: Plot NewVar1 vs. Species and NewVar2 vs. Species

# Exercises in Using R Answers

---

```
> data2 <- iris
> data2$diff.sepal <- data2$Sepal.Length - data2$Sepal.Width
> data2$prod.petal <- data2$Petal.Length*data2$Petal.Width
> layout(matrix(c(1,2),nrow=T))
> boxplot(data2$diff.sepal data2$Species), main="Sepal Diff."
> boxplot(data2$prod.petal data2$Species), main="Petal Product"
```

