~~~
---
title: "Teaching Statistics with R Programming"
subtitle: "NEMATYC 2018"
author: "Billy Jackson, North Shore Community College"
output:
  html_document:
    fig_width: 5
    fig_height: 3
    theme: "united"
---


# Objectives

1.  Introduction to R and RStudio
2.  Chapter 1 - Descriptive Statistics
3.  Chapter 3 - Probability Distributions (Normal and Binomial)
4.  Chapter 5 - Inference on Numeric Variables
5.  Chapter 6 - Inference on Categorical Variables
6.  Chapter 7 - Linear and Multivariate Regression

* * *

# Introduction to R and RStudio

## RMarkdown Files

This is called an RMarkdown file (*.Rmd) which is a type of file that
allows you to execute code in the middle of written paragraphs.  It is
great for technical report writing.  In this kind of file, lines of code
are only executed that are in an "r chunk" which you'll see below.
These are created by clicking the "Insert" button above.

```{r}
# This line that begins with a # is called a comment line
# R will not execute anything in a comment line
# Comments are a way for you to explain what your code does

# This code adds 2 plus 3
2 + 3

# This code multiplies 54 times 78
54 * 68

# Press the green arrow in the top-right corner of this chunk to execute
the two lines of code
```

You'll see in the handout that the result came out disjointed.  There
are a variety of options you can set within chunks and one of them
~~~

allows a chunk to print out smoothly. That option can be set by setting "collapse = TRUE" like such:

```{r, collapse = TRUE}
# This code adds 2 plus 3
2 + 3

# This code multiplies 54 times 78
54 * 68
```
That's just something that can tidy up documents if you choose to use it. Its not a necessity.

## Objects
Data types come in different shapes and forms which R refers to as "objects". Often, we will store these objects into our environment so we can refer to them later when we please. We do that by using the assignment operator: <-

#### Vectors
A vector is a one-dimensional array of numbers. If we found everyone's age in this room, that result would be called a vector. We deal with vectors quite often.

```{r}
# Storing a vector of numbers into an object called numbers
# The combine function, c(), is used to create vectors
numbers <- c(18, 20, 25, 33, 22, 25, 19)
```

There are different classes of vectors: numeric, character, logical, and factor. Numeric are the most frequently used in a statistics class.

#### Scalar
A scalar is just a single number or value. Sometimes we might multiply an entire vector by a scalar. Or perhaps we want to store a single value like a mean to use in another formula later.
```{r}
# Store a scalar
x <- 12
mn <- mean(numbers)
```

#### Dataframes
A dataframe is the most common way for data to be stored. In a dataframe, the rows represent different observations of something and the columns represent different variables.

R comes with several datasets already installed into it. Let's load the `iris` dataset and store it into our environment.

```{r}
# Store iris dataset into our environment
iris <- iris
```

You can refer to a specific variable in a dataframe by using the $.

Example: to do something with the Sepal.Length variable in the iris
dataset, you would refer to that as `iris$Sepal.Length`. One thing to
be careful about is that R is case sensitive!

## Functions
Almost everything in R is done through functions. A function takes the
form of:

> function(arg1, arg2, ...)

where arg1, arg2, etc. are things that you can pass into the function in
order to get an output.

We saw already how we can take the mean of a variable, like the
`numbers` vector, by passing it into the `mean()` function.

```{r, collapse=TRUE}
# Mean of the numbers vector
mean(numbers)

# Find the sum of the numbers vector
sum(numbers)

# Find the sum of the values in the Sepal.Length variable in the iris
dataset
sum(iris$Sepal.Length)
```

#### Practice 1
(on your own if you'd like)
Find the following. If you are unsure how a function should be used or
what the input arguments should look like, try looking up their
documentation in the Help pane.
```{r}
# Use the dim() function to find the dimensions of the iris dataset



# Find the minimum value in the Sepal.Width variable in the iris dataset
```

~~~
# What is the class of the Petal.Length variable



```



## Reading datasets into R
R is capable of reading many different kinds of data files in.  The most
common types of data files are probably CSV files.  R has a base
function called read.csv() that can read CSV data files into R.  Then
you will typically store them as an object in your environment.

The trickiest thing for students in loading data into R with read.csv()
is understanding directories and the paths to the files where they are
saved on their computers.  Luckily, there is a nice little shortcut to
take care of that.  One can load a dataset into your environment by
using the "Import Dataset" button in your Environment.

```{r}
# Copy code from Console into here to load the datafile in RMarkdown doc

```


A very useful function to call when you load in a dataset is the
"structure" function `str()`.  It gives you a snapshot of the dimensions
of the dataset, the names of all the variables, and the class of each
variable.

```{r}
# See the structure of the iris dataset
str(iris)
```



#### Creating a dataset by hand
(to be skipped in presentation)
```{r}
# Defining the vectors that will be the variables
name <- c("Mercury", "Venus", "Earth", "Mars", "Jupiter", "Saturn",
"Uranus", "Neptune")
diameter <- c(0.382, 0.949, 1, 0.532, 11.209, 9.449, 4.007, 3.883)
rotation <- c(58.64, -243.02, 1, 1.03, 0.41, 0.43, -0.72, 0.67)
rings <- c(FALSE, FALSE, FALSE, FALSE, TRUE, TRUE, TRUE, TRUE)

# Join those variables into a data frame using the data.frame() function
planets <-data.frame(name, diameter, rotation, rings)

```
~~~

~~~~
```{r}
# Take a peek at the first 6 rows of the dataset
head(planets)
```


## Indexing and subsetting in a dataframe
(To be skipped in presentation)
You can pick out a specific part of a dataframe by referencing its
index.  Each individual entry in a dataframe has a specific index given
by [row, column].

```{r, collapse = TRUE}
# Print out diameter of Mercury (row 1, column 3)
planets[1,3]

# Print out all data for Mars (entire fourth row)
planets[4,]
```

```{r}
# Get a subset of only the larger planets
large_planets <- subset(planets, diameter > 1)
```

## Packages in R
R is an open-source software and people/developers can write packages to
improve its functionality.  The textbook OpenIntro Statistics has a
package with several datasets used in the textbook's examples that are
tremendous for using in class or for out-of-class assignments.

To download the package called openintro, you first need to run in your
console the line:
 > install.packages("openintro")

Then after that has been downloaded to your machine, you can call the
package into R's memory whenever you need to refer to something in it
with the `library()` function.

```{r}
# Load openintro library
library(openintro)
```


* * * * *

# 2. Descriptive Statistics
Let's load the `ncbirths` dataframe from the openintro library into our
environment.  Clicking on the dataframe in your environment opens it up
in a new tab to view.
~~~~

~~~
```{r, include = FALSE}
# Load openintro library
library(openintro)

# Store ncbirths dataframe into object called ncbirths
ncbirths <- ncbirths
```


## Numeric Data
What is the average age of the fathers?  Standard deviation?
```{r, eval = FALSE}
# Sample mean of fathers ages
mean(ncbirths$fage)

# Sample standard deviation of fathers ages
sd(ncbirths$fage)
```

```{r}
# Summary Statistics
summary(ncbirths$fage)
```

## Categorical Data
How many of the mothers are married?
```{r}
# Table Summary of Categorical Data
table(ncbirths$marital)
```

## Visualizing Data with Histograms and Boxplots
```{r}
# Histogram of mothers ages
hist(ncbirths$mage)
```

```{r}
# Histogram with additional features added
hist(ncbirths$mage, main = "Histogram of Mothers Ages", xlab = "Age",
breaks = 16)
```

Boxplots are not so useful alone, but are very useful when used to
compare datasets side by side.  Let's take a look at the iris dataset
and see if there is a difference in the Sepal.Length variable depending
on the type of Species.
```{r}
# Boxplot of Sepal.Length based on species
boxplot(iris$Sepal.Length ~ iris$Species)
```
~~~

* * * * *

# 3. Probability Distributions (Normal and Binomial)
## Binomial Distribution
There are functions in R that work similarly to our trusty binompdf and
binomcdf. `dbinom()` gives the density of x successes in a binomial
distribution (aka binompdf) and `pbinom()` gives the cumulative density
(aka binomcdf).

Example: If a student guesses on all questions on a ten-question,
multiple-choice quiz, what is the probability he/she will:
a. Get exactly 0 questions correct
b. Score a 50% or worse
c. Score a 70% or better
```{r, collapse = TRUE}
# Binomial probabilities using dbinom and pbinom
# Exactly 0 successes with dbinom()
dbinom(x = 0, size = 10, prob = 0.25)

# 5 or fewer successes with pbinom which gives P(X ≤ x)
pbinom(q = 5, size = 10, prob = 0.25)

# 7 or more successes using lower.tail = FALSE which gives P(X > x)
pbinom(q = 6, size = 10, prob = 0.25, lower.tail = FALSE)
```

## Normal Distribution
There is a family of functions in R that are very helpful with the
normal distribution:

* `pnorm()` gives cumulative normal probability (aka normalcdf)
* `qnorm()` gives the normal quantile for a given probability (aka
invNorm)
* `rnorm()` is a random number generator for a normally distributed
variable

Example: If IQs are a normally distributed variable with mean = 100 and
sd = 15, then:
a. What is probability of selecting a random person with IQ > 110?
b. What IQ denotes the 2nd percentile (lowest 2% of IQs)?
c. Create a random sample of 50 people's IQs.

```{r, collapse = TRUE}
# prob of IQ > 110
pnorm(q = 110, mean = 100, sd = 15, lower.tail = FALSE)

# bottom 2%
qnorm(p = 0.02, mean = 100, sd = 15)
```

~~~
# random sample of 50 IQs
rnorm(n = 50, mean = 100, sd = 15)
```


## T Distributions
There also is a family of functions in R that compute probabilities and
t-scores for the t-distributions.  The `pt()` and `qt()` functions work
here the same way as pnorm and qnorm.

```{r, collapse = TRUE}
# Probability of t-score less than -2
pt(-2, df = 30)

# t-score that cuts off upper 10%
qt(0.1, df = 40, lower.tail = FALSE)
```


* * * * *


# 4. Inference on a Population Mean
There is a single function `t.test()` which handles so many varieties of
inference on numeric data.  All you have to do is give it a variable of
data, a null hypothesis value, your tails, and a confidence level and it
calculates the p-value of a hypothesis test and the confidence interval.
It works with single variables, paired data, or two independent samples.

Question 1: Are birth weights in NC different from the average US
birthweight of 7.7lbs?
```{r}
# single sample mean t-test
t.test(ncbirths$weight, mu = 7.7)
```

Question 2: Is there a difference in age between mothers and fathers of
children?
```{r}
# difference in paired data t-test
t.test(ncbirths$fage, ncbirths$mage, paired = TRUE)
```

Question 3: Is there a difference in child birthweight between mothers
who smoke and mothers who do not smoke?
```{r}
# two independent samples t-test
t.test(ncbirths$weight ~ ncbirths$habit)
```
~~~

~~~
* * * * *


# 5. Inference on a Population Proportion
For inference on a population proportion, there are two handy functions
in R:

* `binom.test()` for simple tests of proportions differing from some
null hypothesized value.
* `prop.test()` for chi-square test statistics and probabilities of
several groups


Question: Is 59 heads in 100 flips of a coin enough to deem a coin
unfair?
```{r}
# Single Proportion Hypothesis Test
binom.test(x = 59, n = 100, p = 0.5)
```



* * * * *


# 6. Linear and Multivariate Regression
Creating scatterplots and linear regression models are very simply in R
with the `plot()` and `lm()` functions.

Let's store the `mtcars` dataframe that comes with R.
```{r}
# Store mtcars into our environments
mtcars <- mtcars
```



Question: Does the weight of a car affect its mpg?
```{r}
# Scatterplot
plot(mtcars$wt, mtcars$mpg)
```

```{r}
# Create linear model.  First argument is "formula" which is of the form
y ~ x
linmod1 <- lm(mpg ~ wt, data = mtcars)
linmod1
```

```{r}
# Details of the linear model
summary(linmod1)
~~~

~~~
```

Multivariate models are a breeze!  Just input more independent variables
into the function.
```{r}
# Multivariate model
multimod1 <- lm(mpg ~ wt + hp + cyl, data = mtcars)
multimod1
```

```{r}
# Details of the multivariate model
summary(multimod1)
```

It doesn't stop there!  Text analysis, classification models, etc.  Feel
free to email me to ask me any questions or to give me feedback on the
presentation at wjackson@northshore.edu.
~~~