Case Study 1: College Basketball

prepared by Kelly Bodwin

In this Case Study, you will refresh your memory of STOR 155 while you learn some basic commands and tools for analyzing data with **R**. We'll be looking at some data from college basketball games in 2015.

Run the following **R** code to load the data into your RStudio and take a look at it.

Summarizing data

```
# Load dataset
bball = read.csv("basketball.csv")

# Look at dataset
head(bball)
```

```
##   X     Date         Team Team.Location Team.Score     Opponent
## 1 1 11/13/15 Old Dominion       Neutral         67      Niagara
## 2 2 11/13/15      Niagara       Neutral         50 Old Dominion
## 3 3 11/13/15 Sacred Heart       Neutral         76   Quinnipiac
## 4 4 11/13/15   Quinnipiac       Neutral         64 Sacred Heart
## 5 5 11/13/15        Texas       Neutral         71   Washington
## 6 6 11/13/15 Wright State       Neutral         77 South Dakota
##   Opponent.Score Team.Result
## 1             50         Win
## 2             67        Loss
## 3             64         Win
## 4             76        Loss
## 5             77        Loss
## 6             69         Win
```

```
summary(bball)
```

```
##        X                Date                          Team
##  Min.   :    1   11/13/15: 310   Akron                :   35
##  1st Qu.: 2914   2/6/16  : 306   Austin Peay          :   35
##  Median : 5827   1/9/16  : 300   Bowling Green State:   35
##  Mean   : 5827   1/16/16 : 288   Buffalo              :   35
##  3rd Qu.: 8740   1/30/16 : 286   Fresno State         :   35
##  Max.   :11653   2/13/16 : 286   Green Bay            :   35
##                  (Other) :9877   (Other)              :11443
##  Team.Location    Team.Score                    Opponent
##  Away   :5240   Min.   : 25.00   Akron                :   35
##  Home   :5251   1st Qu.: 64.00   Austin Peay          :   35
##  Neutral:1162   Median : 72.00   Bowling Green State:   35
##                 Mean   : 72.54   Buffalo              :   35
##                 3rd Qu.: 81.00   Fresno State         :   35
##                 Max.   :144.00   Green Bay            :   35
##                                  (Other)              :11443
##  Opponent.Score   Team.Result
##  Min.   : 25.00   Loss:5821
##  1st Qu.: 64.00   Win :5832
##  Median : 72.00
##  Mean   : 72.51
##  3rd Qu.: 81.00
##  Max.   :144.00
##
```

The command `read.csv( )` will read a dataset into R from your computer or from online. "csv" stands for "comma separated value", a common file type where the data is listed in a text file, with variables separated by commas. For now, you don't need to worry about the details of `read.csv( )`. Once you have loaded the data, the command `summary( )` will tell you about the variables in the dataset and their values. Another useful function is `head( )`, which shows you the first 6 rows of the dataset.

Question 1:

a. Look at the outputs of `summary(bball)` and `head(bball)`, and describe the variables using vocabulary from STOR 155.
b. If `head( )` shows the first 6 rows of the dataset, what command do you think might show the *last* 6 rows? Try out your proposed function and see what happens
c. Try the commands `ncol( )` and `nrow( )`. What do these do? How could you get the same information from `head( )`, `summary( )`, and/or the command you figured out in part (b)?

Sometimes, we will want to look at individual entries, rows, or columns of our data matrix. We can do this using brackets `[ ]` after our dataset. We can also look at a variables (columns) by name using the $ symbol. Try the following examples.

```
# Look at a single row
bball[123, ]

# Look at a single column
bball[ , 5]
bball$Team.Score

# Look at a single entry
bball[123, 5]
bball$Team.Score[123]

# Calculate mean, median, variance, and standard deviation
mean(bball$Team.Score)
median(bball$Team.Score)
var(bball$Team.Score)
sd(bball$Team.Score)
```

Question 2:

a. What is the difference between `mean(bball$Team.Score)` and `mean(bball[,5])`? Why might it be useful to have two ways to get access the variable `Team.Score`?
b. In plain English, what were the events of the game represented by the first row of the dataset?

   (Note: If you don't know much about basketball - for example, if you don't know what it means to play a game "Home" versus "Away" - ask people around you.)

All these commands we have been using, like `summary( )` and `mean( )` are called *functions*. A function can take all different kinds of input depending on what you are trying to do: datasets, vectors such as `bball$Team.Score`, etc. An important skill in **R** is figuring out for yourself how functions work.

For example, type `?boxplot` into your **R** console. A help page will pop up telling you about this function. Notice that under **Usage**, it says `boxplot(x, ...)`. This tells you that you need to supply something called *x* to the function, and the rest of the input is optional. But what is *x*? Ah-ha! There is a section called **Arguments**, which tells us that *x* is the vector of values you want to put in a boxplot.

Run the code below to make a boxplot of the team scores of college basketball games.

```
# make boxplot of team scores
boxplot(bball$Team.Score)
```

Question 3:

a. Now check out `?hist`, a function for making histograms. Below is basic code to make a histogram of `Team.Scores`, and also code for the same histogram but with a lot of the optional input changed. Mess around with these inputs until you understand what each is doing.

```
# Boring histogram
hist(bball$Team.Score)

# Fancy histogram
hist(bball$Team.Score, breaks = 5, main = "I am a title", xlab = "I am an x-axis label", col = "grey", freq =
FALSE)
```

Explain in your own words what `breaks` and `freq` change about the histogram.

b. The optional inputs `main`, `xlab`, `ylab`, and `col` are common to most plotting functions. Use what you learned in (a) to make a boxplot of `Team.Scores` with proper axis labels and title.

```
# make boxplot of team scores
boxplot(bball$Team.Score, YOUR CODE HERE)
```

c. To check if the histogram is Normal, or to help visualize its shape, we might want to overlay a Normal curve on top of the histogram. The code below will do so - but the curve doesn't fit very well.

```
# Boring histogram
hist(bball$Team.Score, freq = FALSE)

# overlay Normal Curve

curve(dnorm(x, mean=120, sd=20),
      add = TRUE, col = "blue", lwd = 2)
```

Explain what the role is of the functions `curve( )` and `dnorm( )`. Why did we put `add = TRUE` in the inputs?

d. Alternatively, we can overlay a line that is a "smoothed" version of the data, as follows:

```
# overlay smoothed curve
lines(density(bball$Team.Score),
      col = "red", lwd = 2, lty = 2)
```

What is the difference between `lines( )` and `curve( )`? When might we want to use `density( )`, and when would it be better to overlay a Normal curve on a histogram?

e. Now make your own histogram with well-chosen inputs and with a Normal overlay that fits better. Would you say the data looks Normal?

```
YOUR CODE HERE
```

Subsetting

One of the most powerful qualities of **R** is the ability to select a subset of a dataset. Suppose we want to look only at games involving UNC or Duke. We would need to figure out which rows of `bball` involve one of those teams, and then make a new dataset out of only those rows.

For this, we will use *booleans*, which are variables with the value `TRUE` or `FALSE`. Play around with the following code until you feel comfortable with `==`, `>`, `<`, and `%in%` as well as `&` (and) and `|` (or).

```
# booleans practice

1 == 1
1 == 2
1 < 2

1 == 1 | 1  > 2
1 == 1 & 1 > 2
```

You can make up your own vector using the function `c( )`, which stands for "concatenate". This is like making a new variable - the variable can contain anything you want, such as numbers, strings, booleans. Try the example below to make a vector and subset it. Note that we can use either `<-` or `=` to store information in a variable.

```
vec <- c("cat", "dog", "horned toad", "Her Majesty Queen Elizabeth", "dog")
vec

# Some more booleans
vec == "dog"
"dog" == vec
vec %in% c("dog", "cat")
c("dog", "cat") %in% vec


# Finding indices

which(vec == "dog")
which(vec %in% c("dog", "cat"))
which(c("dog", "cat") %in% vec)

# Subsetting
new = vec[vec %in% c("dog", "cat")]
new
```

Question 4:

a. The following code will give you an error. What happened?

```
vec = c(1, 2, 3, "4")
vec + 2
```

b. The following code will NOT give you an error? What is going on here?

```
vec = c(TRUE, FALSE, FALSE, TRUE)
vec + 2
```

c. Now we are ready to make a new dataset. We'll get a list of booleans to tell us where UNC or Duke's games are, and use that to subset the datset `bball`.

Try running each of the following lines of code. None of them will make the datset we want. What was the problem with each one?

```
# Make new dataset with only UNC or Duke games


#A
my_subset = bball[Team == "North Carolina" | Team == "Duke", ]

#B
my_subset = bball[bball$Team == "North Carolina", bball$Team == "Duke"]

#C
my_subset = bball[bball$Team = "North Carolina" | bball$Team = "Duke", ]

#D
my_subset = bball[bball$Team == "North Carolina" & bball$Team == "Duke", ]

#E
unc_games = which(bball$Team == "North Carolina")
my_subset = bball[unc_games | bball$Team == "Duke", ]

#F
my_subset = bball[bball$Team == "North Carolina" | bball$Team == "Duke"]
```

d. Now write your own code to make the correct dataset.

```
YOUR CODE HERE
```

Z-Scores and t-scores

Alright, enough of that data wrangling. Time to do some statistics.

Check out `?Normal`. These are some functions that will help us calculate probabilities about the Normal distribution. (No more using Table A!) The most important ones are `pnorm` and `qnorm`.

`pnorm(q)` will tell you the probability of a standard Normal being below the value `q`

`qnorm(p)` will tell you the z-score that has area `p` below it on a standard Normal curve

Question 5

a. For each of the following lines of code, think about what the result will be **before** running the code. **Draw a picture for each one to visualize what is going on with `pnorm` and `qnorm`.**

```
# practice with Normal densities in R

#i
pnorm(0)
qnorm(0)

#ii
pnorm(100)
qnorm(100)

#iii
qnorm(pnorm(0))
qnorm(pnorm(7))

#iv
pnorm(qnorm(0))
pnorm(qnorm(0.5))

#v
pnorm(0, sd = 10)
pnorm(0, mean = 1, sd = 10)

#vi
qnorm(0.05)
qnorm(0.05, sd = 10)
qnorm(0.05, mean = 1, sd = 10)
```

b. Why did you get an error in part (ii)?

Now use this code to make a new variable for the total score of a game:

```
# Make new variable
bball$Total.Score = bball$Team.Score + bball$Opponent.Score
```

We will use *z-scores* and *t-scores* to think about whether a game is unusually high scoring.

Question 6:

a. As you may have noticed, the dataset `bball` actually displays each game twice: once for each team. Make a new dataset with each game listed only once by subsetting `bball`.

```
YOUR CODE HERE
```

b. On Feb 17, 2016, UNC played Duke. Using the Normal distribution, what percent of games have higher scores than the UNC/Duke game? (Assume that the mean and standard deviation of `Team.Score` are actually the *population* mean and standard deviation.)

```
YOUR CODE HERE
```

c. What percentage of games in the dataset did we observe to be higher scoring than the UNC/Duke game? The functions `sum( )` and `length( )` will help you answer this question.

```
YOUR CODE HERE
```

d. What is the difference between what we did in (b) and (c)? Do you think the Normal approximation is reasonable for this data? Why or why not?

Recall that *t-scores* are used instead of *z-scores* when the population standard deviation is unknown. The functions `pt` and `qt` work almost same way as `pnorm` and `qnorm`, but for the t-distribution instead of the Normal. However, be careful, and read `?pt` for help! These functions don't let you enter the mean and standard deviation as input - you need to figure out what do about that!

Question 7:

Use all your new **R** skills to answer this question: Was the Feb 17th game between UNC and Duke particularly high scoring *for a UNC game*?

```
YOUR CODE HERE
```

Confidence Intervals and Proportions

You now have all the **R** knowledge you need to make some confidence intervals! You may wish to go over your lecture notes for this section, especially to remind yourself how to deal with proportions.

Question 8:

a. Make a 95% confidence interval for the number of points UNC scores in a given game. You will need to think about which **R** commands will give you critical values of the *t*-distribution, and how to use these to make a confidence interval.

```
YOUR CODE HERE
```

b. What percentage of games did UNC win in 2015-2016? Make a 95% confidence interval for their win percentage.

```
YOUR CODE HERE
```

Hypothesis Testing

You have now had lots of practice learning to use a function by reading the documentation. Part of the point of this course is for you to become familiar enough with **R** to learn new commands and functions without being shown how to use them. This will make you a skillful (and hireable!) programmer in the future.

Check out `?t.test` and `?prop.test`. Figure out what these functions do, what input they take, etc. Then answer the following questions.

Question 9:

a. Does UNC tend to win more games than they lose? That is, is there evidence at the 0.05 level that the "true" probability of UNC winning a given game in 2015-2016 is larger than 0.5?
b. Based on how many points they tend to score in a game, would you say UNC and Yale were equally good teams?
c. Based on win percentage, would you say UNC and Yale were equally good teams? Discuss this result and the result in (b).

```
yale.won = sum(yale_games$Team.Result == "Win")
n2 = nrow(yale_games)

prop.test(c(unc.won, yale.won), c(n, n2))
```

Comparing multiple means (Analysis of Variance)

What if we want to compare more than one team? In lecture, you learned about using an Analysis of Variance (ANOVA) F-test to check if more than two means are equal. We will use the function `aov( )` to find out if the big three North Carolina teams - UNC, Duke, and NC State - all tend to score the same number of points.

Question 10

a. Make a dataset called `nc_games` that includes only games for the North Carolina teams, and then alter the code below to create a box plot of the scores for the three North Carolina teams. Does it look like any of the means are significantly different?
b. Perform an ANOVA F-test on the means. Interpret the output. Is there evidence that the average scores of the three teams are not all equal?