# Warmup 03 - Basic Data Manipulation

Stat 133, Spring 2019

## Introduction

The purpose of this assignment is to explore data collected for NBA players during the regular season 2017-2018. You will explore player salaries, scored points, and a few other variables such as the age and position of players.

The main goal of this warmup is to give you practice working with different data types, subsetting, and creating vectors and data frames. A side benefit will be that you will gain some practice thinking with data as you carry out your computations. The analysis of the data includes examining univariate distributions, exploring relationships between variables, and plotting.

### General Instructions

- Write your narrative and code in an `Rmd` (R markdown) file.
- Name this file as `warmup03-first-last.Rmd`, where `first` and `last` are your first and last names (e.g. `warmup03-gaston-sanchez.Rmd`).
- Please do not use code chunk options such as: `echo = FALSE`, `eval = FALSE`, `results = 'hide'`. All chunks must be visible and evaluated.
- Submit your Rmd and html files to bCourses.

---

## 1) Importing Data

The first task involves reading the data into R. Go through each of the following steps so that you begin to learn about the reading table functions in R.

- Look at the content of the data file, located in the github repository for homework assignments (see folder `data/`):

https://raw.githubusercontent.com/ucb-stat133/stat133-hws/master/data/nba2018-players.csv

- Examine the layout of the data. Do not download it to your computer.
  - Do the data have a header containing the variable names?
  - Are the values for an observation separated by a comma, blank, or tab?
  - Are there any missing vales? If so, how are they codified?

- Read the documentation for `read.table()` and `read.csv()` and examine the parameters that are used to specify whether the data have a header and how the values in a row are separated. Notice that these functions are similar, but they have different default values for these parameters. You can also look at the related slides and cheatsheet (links in github repo).
- The code below creates a string `datafile` with the url for the location of `nba2018-players.csv`. Use this string as the value of the argument `file` for `read.table()` and `read.csv()`.

```
# assembling url so it fits on the screen
github <- 'https://raw.githubusercontent.com/ucb-stat133/stat133-hws/'
repo <- 'master/data/nba2018-players.csv'
datafile <- paste0(github, repo)
```

a) In order to import the data table, you will have to specify data types for each column—via the `colClasses` argument of `read.table()` and `read.csv()`. To do this, create a character vector `data_types` based on the following specifications:

- Variables `player` and `college` must be imported as `"character"` (not as factors).
- Variables `team` and `position` must be imported as `"factor"`.
- Numeric variables `height`, `weight`, `age`, `experience`, `games`, `minutes`, `points`, `points3`, `points2`, and `points1` must be imported as `"integer"`.
- Variable `salary` must be imported as `"double"` or `"real"`.

b) Following the specifications listed above, use `read.csv()` to read the data from the github repository into a data frame called `nba`. After you imported the data, use `str(nba, vec.len = 1)` to get a report of the dimensions of the data frame, as well as the class of each column.

c) Use `read.table()` to read the data from the github repository into a data frame called `nba2`. After you imported the data, use `str(nba2, vec.len = 1)` to get a report of the dimensions of the data frame, as well as the class of each column.

d) Finally, run the following command:

```
identical(nba, nba2)
```

---

## 2) Technical Questions about importing data

Answer the following questions (using your own words). You do NOT need to include any commands.

a) What happens to the column names of the imported data when you invoke `read.csv(datafile, header = FALSE, nrows = 10)`?

b) What happens to the data types of the columns when you invoke `read.csv(datafile, header = FALSE, nrows = 10)`?

c) Why does the command `read.table(datafile, nrows = 10)` fail to import the data?

d) Say you import `nba2018-players.csv` in two different ways. In the first option you import the data without specifying the data type of each column. In the second option you do specify the data types as in `data_types`. You may wonder whether both options return a data frame of the same memory size. You can actually use the function `object.size()` that provides an estimate of the memory that is being used to store an R object. What data importing function returns the smallest data frame (in terms of memory size)?

e) Say the object `nba` is the data frame produced when importing `nba2018-players.csv`. If you invoke `as.matrix(nba)`, what happens to the data values of this matrix?

---

## 3) Examine Salary

a) We start by exploring salary. Make summary statistics of salary. Remember that you can access a variable within a data frame with the `$` sign.

It's a bit surprising to see that the lowest salary is only 5145. Also notice that the mean salary is about 5.8 million dollars—more than the median salary, which indicates that the distribution of salary may be skewed right, possibly by having a long right tail.

b) Graph a histogram of salary with `hist()`. No need to give fancy labels at this point; we're just exploring the data. Be sure to specify enough bins to see the shape of the data. You should be able to find that the salaries are skewed to the right with relatively few players earning above $15 millions. That skewness explains why the mean is higher than the median. The few players that earn more than 15 million dollars have pulled it away from the typical salary.

c) To have a more readable histogram, add a new variable to `nba` (i.e add a new column) called `salary2` containing the salary values in millions of dollars. And then graph another histogram for this new variable.

d) Does a log transformation make the `salary2` distribution more symmetric? Log-transform `salary2` and graph another histogram. Now we see those unusually small values cropping up on the left. What about the rest of the distribution? Make comments about its shape.

e) Let's use subsetting to dig a little deeper into the issue of the unusually low salaries. Create a logical vector that indicates if the log of `salary2` is less than -3. Assign this logical vector to the variable named `low`. How many players have what we call a `low` salary (in millions of dlls)?

f) Use logical subsetting to display a data frame with the values of the player name, weight, height, team, and position (in that order) for `low` salary players.

g) Use the variable `low` to (logically) subset the data frame `nba` and remove these rows from the data frame. Call the new data frame `nba`; that is, replace the existing data frame with the new one. Display the dimensions (number of rows and columns) of `nba`.

---

## 4) Explore Points

Next we explore the number of scored points. Do you find a similar issue with the salary?

a) Examine the summary statistics for scored points and make a histogram of the values for points.

We again find that the distribution is skewed to the right.

b) What about the log transformation of points? Because there are various players with 0 scored points, we need to add a small amount to these values in order to avoid calculating `log(0)`. Use logical subsetting to identify those players with 0 points, and then reassign these values with 0.1. Make a histogram of log-transformed scored points.

Comment on the shape of the distribution of the log-transformed points.

---

## 5) Transforming Age into a Factor

We next explore the relation between Salary and Age, as well as Points and Age.

a) Begin by making a *conditional* boxplot using the code below. Compare the boxplots by age, and comment on whether there seems to be a difference in salaries depending on the age of the players.

```
boxplot(salary2 ~ age, data = nba,
        xlab = "Age", ylab = "Salary (in millions)")
```

b) Repeat the same type of plot, but this time using `points` instead of `salary2`. Compare the boxplots by age, and comment on whether there seems to be a difference in scored points depending on the age of the players.

c) We next transform the `age` of players into a factor, i.e. into a categorical variable. We want to *categorize* age into three groups: 20-24, 25-29, 30 or more. To do this, follow this approach:

- Assign the variable `age` in `nba` to a new vector called `age2`.
- Use subsetting to set all of the values in `age2` that are less than 19 to 20.

- Use the `cut()` function to convert `age2` to a factor. Read carefully about the `breaks` and `labels` parameters of this function. For the categories, use the following strings: `"20-24"`, `"25-29"`, and `"30+"`. Assign the factor to `age2`, i.e., overwrite the numeric `age2` with the *cutted* factor `age2`.
- Display a summary of `age2`.

---

## 6) Plotting Salary against Points, by Age Group

We will now make a plot of Salary (y-axis) against Points (x-axis) (both on the log scale) and color the plotting symbols according to the factor `age2`. We will use the new factor `age2` to determine the color.

a) Begin by making a vector of three colors. Use the colors called `"#D4D62A"`, `"#4F9D66"`, and `"#9575AB"`, in that order. Call this vector, `palette1`.

b) Next, make a vector of colors that matches the length of `age2`, and where a value is `"20-24"`, the color is `#D4D62A`; where a value is "25-29", the color is `#4F9D66`, etc. Use subsetting by position of the vector `palette1` to do this. Call this new vector, `age_colors`.

c) Lastly, make the scatterplot as follows (Don't worry about the various arguments to the function call):

```
plot(salary ~ points, data = nba, log = "xy", main = "",
      xlab = "Points (log scale)", ylab = "Salary (log $)",
      col = age_colors, pch = 19, cex = 0.6)

legend("bottomright", fill = palette1, legend = levels(age2),
       title = "Age Groups", cex = 0.8)
```

You should be able to see in the plot that salary and points have a roughly linear relationship and that players with more scored points tend to have higher salaries. This makes sense.

---

## 7) Plotting Salary against Points, by Position

The variable `position` indicates the position of a player.

a) Begin by making a *conditional* boxplot of `salary2` by `position`. Compare the boxplots by age, and comment on whether there seems to be a difference in salaries depending on the position of the players.

b) Repeat the same type of boxplot, but this time using `points` instead of `salary2`. Compare the boxplots by position, and comment on whether there seems to be a difference in scored points depending on the position of the players.

c) Follow the approach in the previous section (6) to create a new variable called `pos_colors` with colors for positions:

- *Center* `C`: `"#66c2a5"`
- *Power Forward* `PF`: `"#fc8d62"`
- *Point Guard* `PG`: `"#8da0cb"`
- *Small Forward* `SG`: `"#e78ac3"`
- *Shooting Guard* `SG`: `"#a6d854"`

d) Make a plot of salary against points (determine if they should be on the log scale) and color the plotting symbols according to position. Don't forget to add a legend.