Final Project

Stat 133, Fall 2016, Prof. Sanchez

Due Date: December 1, 2016

Abstract

The final project involves analyzing data of NBA players. This is a group project (you cannot work individually). You should form groups up to a max of 4 students. You will have the opportunity to put in practice all the concepts, skills, and ideas covered in the course.

Performance and Salaries of NBA Players

The final project involves analyzing data about basketball players from the National Basketball Association (NBA) League in the 2015-2016 season. The central topic behind this project has to do with the salary and performance of NBA players. To be more precise, the motivating question is: *"In the 2015-2016 season, how do the skills of a player relate to his salary?"*

To organize the project, you should use the file-structure described in these instructions. To submit the files and results, you should use the Open Science Framework (OSF) template project which mimics the file structure you will be using in your computer.

Data Sets

The primary source of data for this project is: Basketball Reference (by *Sports Reference LLC*) which provides statistics, scores, and history for the NBA, ABA, WNBA, and top European competition.

Among the available NBA data sets you will work with three kinds of raw data tables: **Roster**, **Totals** (i.e. player statistics), and **Salaries**. You can find these tables in the corresponding pages for each team. For example, all the tables about the Golden State Warriors (season 2015-2016) are here:

http://www.basketball-reference.com/teams/GSW/2016.html

- GS Warriors' Roster table:

http://www.basketball-reference.com/teams/GSW/2016.html#all_roster

- GS Warriors' Totals table:

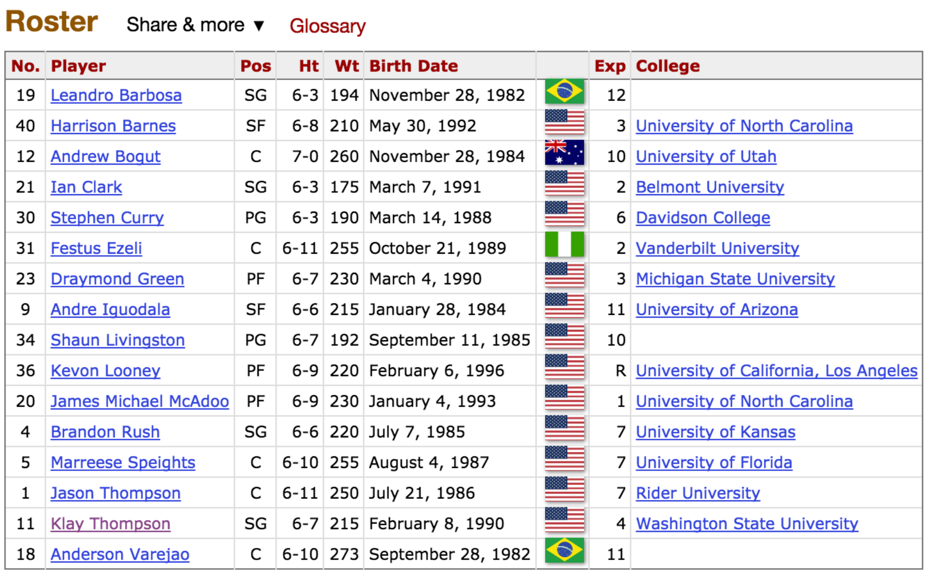
http://www.basketball-reference.com/teams/GSW/2016.html#all_totals

- GS Warriors' Salaries table:

http://www.basketball-reference.com/teams/GSW/2016.html#all_salaries

Roster table

The Roster table contains information about each palyer's name, position, height, weight, birth date, country of origin, years of experience, and attended college. Below is a screenshot of this type of table for the Golden State Warriors:

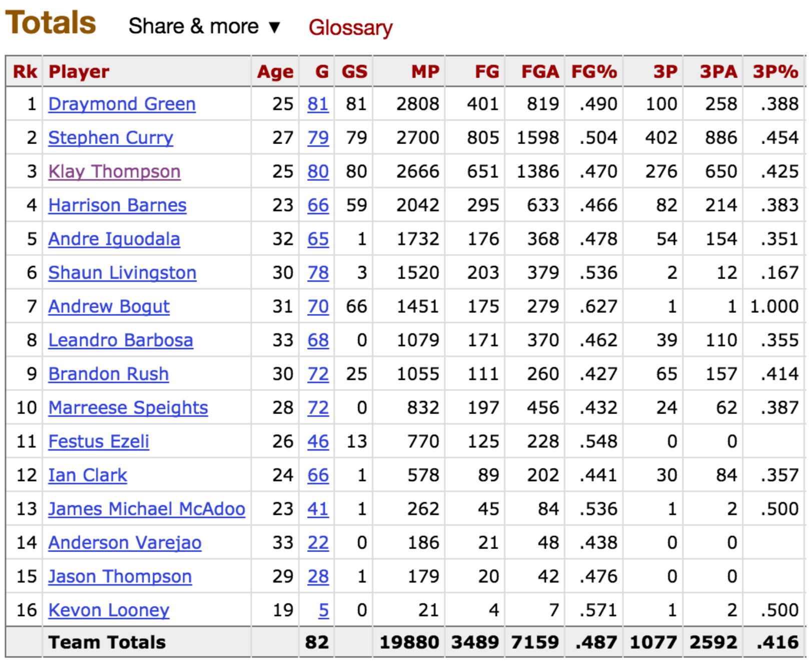


Roster
Share & more ▼
Glossary

No.	Player	Pos	Ht	Wt	Birth Date		Exp	College
19	Leandro Barbosa	SG	6-3	194	November 28, 1982	🇧🇷	12	
40	Harrison Barnes	SF	6-8	210	May 30, 1992	🇺🇸	3	University of North Carolina
12	Andrew Bogut	C	7-0	260	November 28, 1984	🇦🇺	10	University of Utah
21	Ian Clark	SG	6-3	175	March 7, 1991	🇺🇸	2	Belmont University
30	Stephen Curry	PG	6-3	190	March 14, 1988	🇺🇸	6	Davidson College
31	Festus Ezeli	C	6-11	255	October 21, 1989	🇳🇬	2	Vanderbilt University
23	Draymond Green	PF	6-7	230	March 4, 1990	🇺🇸	3	Michigan State University
9	Andre Iguodala	SF	6-6	215	January 28, 1984	🇺🇸	11	University of Arizona
34	Shaun Livingston	PG	6-7	192	September 11, 1985	🇺🇸	10	
36	Kevon Looney	PF	6-9	220	February 6, 1996	🇺🇸	R	University of California, Los Angeles
20	James Michael McAdoo	PF	6-9	230	January 4, 1993	🇺🇸	1	University of North Carolina
4	Brandon Rush	SG	6-6	220	July 7, 1985	🇺🇸	7	University of Kansas
5	Marreese Speights	C	6-10	255	August 4, 1987	🇺🇸	7	University of Florida
1	Jason Thompson	C	6-11	250	July 21, 1986	🇺🇸	7	Rider University
11	Klay Thompson	SG	6-7	215	February 8, 1990	🇺🇸	4	Washington State University
18	Anderson Varejao	C	6-10	273	September 28, 1982	🇧🇷	11	

Totals table

The Totals table contain player's statistics during the entire season: age, games played, games started, minutes played, field goals, field goal attempts, etc. The image below shows a screenshot for the GS Warriors with some of the variables in this type of table:

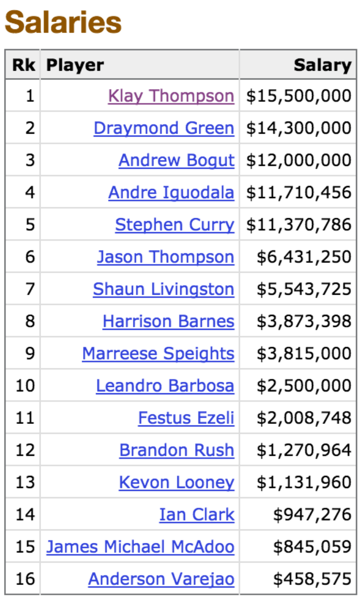


Totals
Share & more ▼
Glossary

Rk	Player	Age	G	GS	MP	FG	FGA	FG%	3P	3PA	3P%
1	Draymond Green	25	81	81	2808	401	819	.490	100	258	.388
2	Stephen Curry	27	79	79	2700	805	1598	.504	402	886	.454
3	Klay Thompson	25	80	80	2666	651	1386	.470	276	650	.425
4	Harrison Barnes	23	66	59	2042	295	633	.466	82	214	.383
5	Andre Iguodala	32	65	1	1732	176	368	.478	54	154	.351
6	Shaun Livingston	30	78	3	1520	203	379	.536	2	12	.167
7	Andrew Bogut	31	70	66	1451	175	279	.627	1	1	1.000
8	Leandro Barbosa	33	68	0	1079	171	370	.462	39	110	.355
9	Brandon Rush	30	72	25	1055	111	260	.427	65	157	.414
10	Marreese Speights	28	72	0	832	197	456	.432	24	62	.387
11	Festus Ezeli	26	46	13	770	125	228	.548	0	0	
12	Ian Clark	24	66	1	578	89	202	.441	30	84	.357
13	James Michael McAdoo	23	41	1	262	45	84	.536	1	2	.500
14	Anderson Varejao	33	22	0	186	21	48	.438	0	0	
15	Jason Thompson	29	28	1	179	20	42	.476	0	0	
16	Kevon Looney	19	5	0	21	4	7	.571	1	2	.500
	Team Totals		**82**		**19880**	**3489**	**7159**	**.487**	**1077**	**2592**	**.416**

Salary table

The salaries table simply contains the salary of the players.



Salaries

Rk	Player	Salary
1	Klay Thompson	$15,500,000
2	Draymond Green	$14,300,000
3	Andrew Bogut	$12,000,000
4	Andre Iguodala	$11,710,456
5	Stephen Curry	$11,370,786
6	Jason Thompson	$6,431,250
7	Shaun Livingston	$5,543,725
8	Harrison Barnes	$3,873,398
9	Marreese Speights	$3,815,000
10	Leandro Barbosa	$2,500,000
11	Festus Ezeli	$2,008,748
12	Brandon Rush	$1,270,964
13	Kevon Looney	$1,131,960
14	Ian Clark	$947,276
15	James Michael McAdoo	$845,059
16	Anderson Varejao	$458,575

Data Acquisition

There are 30 teams in the NBA. You should scrape the three kinds of tables (Roster, Totals, and Salaries) for each team, resulting in 90 raw data files. Use the `data/rawdata/` directory to store the raw data files, separated in folders `roster-data/`, `stat-data/`, and `salary-data/`. We recommend that you write special functions to scrape the data. Then `source()` these functions in an R script `download-data-script.R` that will be executed to achieve the data acquisition stage.

Data Cleaning

From the raw data files, you have to produce one CSV file, `roster-salary-stats.csv`, that will be stored in the directory `data/cleandata/`. This file should contain all variables from Roster, Totals, and Salary (with only one column for the Name of the player). Do not use abbreviated variable names. For example, instead of `"G"` use something more descriptive like `"games"`, or instead of `"2P%"` use `"two-point-pct"`.

You should write auxiliary cleaning functions that you keep in R files inside the `code/functions/` directory. The cleaning task should be done using a dedicated R script `clean-data-script.R` that `source()`s the functions and produces the clean csv file. The cleaning script should be stored in the folder `code/scripts/`.

One of the key variables is `"position"`. The standard basketball positions are:

- *C*: center
- *PF*: power forward
- *SF*: small forward

- *SG*: shooting guard
- *PG*: point guard

If you find players that have a position different from those listed above, exclude them from your clean data set. For example, if there is a player that happens to have position `"G"`, remove it so it is not part of the exported clean set `roster-salary-stats.csv`.

Analysis

As mentioned in the introduction, the motivating question for this project is: *"In the 2015-2016 season, how do the skills of a player relate to his salary?"* This is an open-end question that can be answered in different ways depending on how you define the concepts: *skills* and *relate* (i.e. type of relationship). Derived from the main question, we can consider the following points:

- How can we evaluate the performance of the players?
- Which skills are more correlated with salary?
- Are there any differences in skills and salary depending on the players' position?
- Are the players really worth the amount of money clubs pay for?
- Are there any undervalued or overvalued players?

To tackle all these points, you will have to perform a global analysis that targets:

1. Salaries aggregated by team
2. Perfomance (i.e. efficiency) of players by position
3. Correlations between salary and skills
4. Value of a player

Exploratory Data Analysis

After clening the data, every data analysis project should involve an exploratory phase commonly referred to as Exploratory Data Analysis (EDA). Usually, this phase comprises analyzing one variable at a time (i.e. univariate analysis) calculating descriptive summaries for quantitative variables (e.g. mean, median, min, max, std dev, range, etc), or frequencies for qualitative variables.

You should create a dedicated R script `eda-script.R` for this analysis. Store this script in the folder `code/scripts/`. You should compute summary statistics for each variable and `sink()` those results to a text file `eda-output.txt` in the fodler `/data/cleandata`. You should also produce histograms, boxplots, and bar-charts for each variable and, assuming that you plot them with `ggplot2`, `save()` them in png or pdf format in the folder `images/`.

Salary statistics per team

To analyze salaries aggregated by teams, you will have to obtain a table `team-salaries.csv` containing the following salary's summary statistics for each team (teams in rows, summaries in columns):

- total payroll
- minimum salary
- maximum salary
- first quartile salary
- median salary
- third quartile salary
- average salary
- interquartile range
- standard deviation

Store the table `team-salaries.csv` in the folder `data/cleandata/`.

Shiny App "team-salaries": To visualize the different statistics, you will have to create a shiny app that displays a horizontal bar-chart (one bar per team). Use one widget to select the desired statistic, and another widget to indicate whether the bars should be displayed in decreasing order or in ascending order. Optionally, you can add more widgets to provide more options for customizing the displayed bar-chart (colors, bar widths, legends, annotations, etc).

Performance of players

Performance of NBA players is usually measured with a formula that calculates what is known as **EFF** or "efficiency" statistic: https://en.wikipedia.org/wiki/Efficiency_(basketball)

EFF computes performance as an index that takes into account basic individual statistics: points, rebounds, assists, steals, blocks, turnovers, and shot attempts (per game). It is derived by a simple formula:

```
EFF = (PTS + REB + AST + STL + BLK - Missed FG - Missed FT - TO) / GP
```

- EFF: efficiency
- PTS: total points
- REB: total rebounds
- AST: assists
- STL: steals
- BLK: blocks
- Missed FG: missed field goals
- Missed FT: missed free throws
- TO: turnovers
- GP: games played

One of the issues with EFF is that it favors offense-oriented players over those who specialize in defense, as defense is difficult to quantify with currently tabulated statistics. Closely related with this issue, EFF does not take into account the different positions of the players:

Alternative EFF indices with PCA: To compensate for the drawbacks in EFF, you will have to come up with different efficiency indices that do take into account the players' positions. To calculate these indices, you will use Principal Components Analysis (PCA). PCA will give you a weight for each term in the original EFF formula:

$$EFF = w_1 \frac{PTS}{GP} + w_2 \frac{REB}{GP} + \cdots + w_8 \frac{TO}{GP}$$

- The EFF formula uses `Missed FG` (missed field goals) and `Missed FT` (missed free throws). These variables are not in the table of statistics.
- Write a function that takes the information in `"FG"` and `"FGA"` to compute `Missed FG`.
- Likewise, write a function that takes the information in `"FT"` and `"FTA"` to compute `Missed FT`.
- Notice that the EFF formula divides all the variables `"PTS"`, `"REB"`, ..., `"TO"` by games played `"PG"`. This means that you should divide each of the statistics by `"PG"`.
- Also notice that `"Missed FG"`, `"Missed FT"`, and `"TO"` contribute negatively to EFF. This means that you should affect these variables with a negative sign.
- To calculate separate EFF indices for each position, you need to subset the player statistics dataset by positions, that is, a subset for C, PF, SF, SG, and PG.
- There are various ways (i.e. functions and packages) to compute PCA. Perhaps the simplest way is with the function `prcomp()`.
- Perform a PCA on each subset (use scaled data: mean-centered and standardized).
- Take the weights of the first principal component as the coefficients of the EFF index. These weights are in the first column of `rotation`, from the object returned by `prcomp()`.
- Because the PCA weights are obtained using standardized variables, you have to re-express the weights to take into account the standard deviation of each variable.
- The modified efficiency, EFF^*, should be computed like this:

$$EFF^* = \frac{w_1}{\sigma_1}\mathbf{x_1} + \cdots + \frac{w_8}{\sigma_8}\mathbf{x_8}$$

where:

- $\mathbf{x_j}$ is the j-th variable, $j = 1, 2, \ldots, 8$
- w_j is the weight for variable $j = 1, 2, \ldots, 8$
- σ_j is the standard deviation of variable $j = 1, 2, \ldots, 8$

Data set `"eff-stats-salary.csv"` Create a new table `eff-salary-stats.csv` in the directory `data/cleandata/`. This table should contain the following variables:

variable	variable
1) player's name	2) total points
3) total rebounds	4) assists
5) steals	6) blocks
7) missed field goals	8) missed free throws
9) turnovers	10) games played
11) efficiency index	12) salary

Shiny app "stat-salaries": To visualize the relationship between all the player statistics—including the EFF^* index—and the salary, you have to create a shiny app that displays a scatterplot in which the x-axis corresponds to one statistic, and the y-axis corresponds to salary (by default). Include a widget that lets you select the specific statistic for the x-axis. Include another widget that lets you select a variable for the y-axis (`salary` by default). Include another widget to indicate whether the dots should be colored by position. The app should also display the correlation coefficient between the chosen variables. Optionally, you can add more widgets to further customize the displayed graphic.

Value of a player

To measure the "value" of a player, you can compute an efficiency over salary ratio obtained as:

$$value = \frac{efficiency}{salary}$$

In other words, you have to compute this ratio for each player, using the EFF^* index (by position), and dividing it by the salary. Use this value to identify the top 20 players (i.e. most valuable) and the bottom 20 players (i.e. worst value). Save the list of these 40 players in a text file `best-worst-value-players.txt` inside the `data/cleandata` directory.

Project's File Structure

The following diagram depicts how you should organize folders and files in your computer. Elements that end with a forward slash `"/"` represent directories (i.e folders). Elements that represent files have an associated extension. Three dots `"..."` indicate a series of files: you should decide the number of files, as well as their names, that you put in these folders.

```
project/
    README.md
    code/
        functions/
            ...
        scripts/
            download-data-script.R
            clean-data-script.R
            eda-script.R
            compute-efficiency-script.R
    data/
        rawdata/
            roster-data/
                ...
            salary-data/
                ...
            stat-data/
                ...
        cleandata/
            roster-salary-stats.csv
            eff-stats-salary.csv
            eda-output.txt
    images/
      ...
    apps/
        team-salaries/
            ui.R
            server.R
        stat-salaries/
            ui.R
            server.R
    report/
        report.Rmd
        report.pdf
    slides/
        slides.Rmd
        slides.html
```

Open Science Framework (OSF) Template

The project's file structure previously described is the structure you should use to organize the files in your computer. However, in order to submit all these files, you will have to use an **Open Science Framework** (OSF) project: https://osf.io/.

OSF (see https://osf.io/4znzp/wiki/home/) "is a free, open source web application that connects and supports the research workflow, enabling scientists to increase the efficiency and effectiveness of their research Researchers use the OSF to collaborate, document, archive, share, and register research projects, materials, and data."

Each team member will have to sign up for free to OSF. Then, you have to decide who will be the "Administrator" of the project. To work on the OSF platform, you should *fork* (i.e. get a copy of) the template project for Stat 133: https://osf.io/e78jv/.

This means that all members will be registered in OSF, but just one person will be responsible of the team's OSF project. The Administrator should grant access and permissions to all the other team members, the GSI's, and the instructor.

The provided OSF project template mimics the file structure. There are various components such as: Data, Code, Apps, Report, and Slides. This OSF project is actually the one that will be graded, together with your live shiny apps shared via www.shinyapps.io/. You do NOT have to submit anything to bCourses.

Checklist

Organization

- Use the Open Science Framework (OSF) template. You will use an OSF project as a sort of repository for your project files. Please do NOT modify the structure of this template.
- All names of files and directories should NOT contain spaces.
- All files should contain an associated extension (e.g: .R, .Rmd, .txt, .csv, .pdf, .png, etc)
- Use a consistent naming convention (for files, directories, functions, objects)
- Use relative pathnames when manipulating files and directories. Do NOT use absolute pathnames!

Raw Data

- Use the `data/rawdata/` folder to store the raw files.
- Do NOT manipulate any of the values in the raw data files (leave them as you obtained them).
- Include a data dictionary describing the variables, fields, or columns of the raw data.

Clean Data

- Use the `data/cleandata/` folder store the clean data sets, as well as other derived data sets.
- If necessary, include a data dictionary describing the variables, fields, or columns of the clean data (e.g. when transforming variables).

General Coding

- Write lines of code having a width of less than or equal to 80 characters
- Add a header to all script files (general description, imported functions)
- If a script needs your programmed functions, source them with `source()` at the beginning of the script (after the header)
- Load all required packages at the beginning of each script (after the header)
- Organize scripts in differentiated sections
- Use indentation (e.g. inside body of functions, expressions in a loop)
- Use comments (but don't belabor the obvious)
- Use blank spaces (between sections, around function arguments, around operators, etc)

Functions

- Write functions in separated files from analysis-script-files
- Organize functions (possibly in different files) by similarity or purpose
- Add a header to files containing functions
- Use consistent naming style
- Use meaningful names for functions and their arguments
- Document all functions (short description, inputs, outputs)
- Write functions with a length of less than 15 lines (discounting blank lines)

README

- Is there a README file?
- Does it contain a title?
- Does it contain a description?
- Does it contain author(s) information (first, last)?
- Does it describe the directory-files structure?

Report

You should write a report for this project in the form of a paper with the following sections:

- Abstract
- Introduction
- Data
- Methodology
- Results
- Conclusions

Make sure all the images and tables have captions. Try to no exceed 30,000 characters in the content of the paper. This is approximately eight pages, single spaced in 12 point font, when pasted in a typical word processor (e.g. MS Word).