

WebScraping Report – Python Scripting

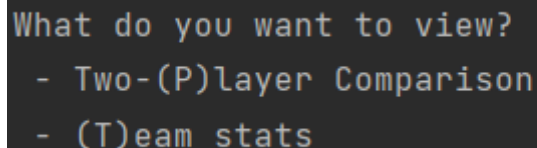
Our program is designed to scrap all the data from the stat table found on the NBA official website: https://www.nba.com/stats/players/traditional/?sort=TEAM_ABBREVIATION&dir=1

To do so, we made sure to accept cookies upon launching the driver on the website and proceed to retrieve every player's data from the table. Each player's info is saved in a list itself in a list and after ensuring we retrieved the column names, we put all of that information into a pandas DataFrame and eventually in a csv File at disposal for our plot.

I/ Program Walkthrough:

In order to make the program enjoyable to use, interactivity was an important step for us. For this reason, the user can choose multiple options while navigating the program :

First of all, when launching the program we are presented a choice : either compare two players' stats by giving the program their last names, or see stats of players of a selection of teams.

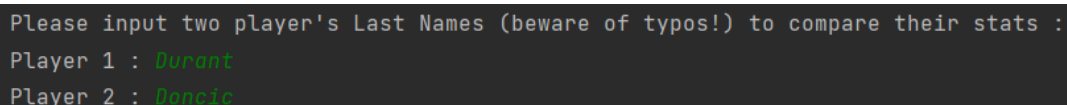


```
What do you want to view?  
- Two-(P)layer Comparison  
- (T)eam stats
```

Figure 1. First choice by the user

Until the user puts in the correct input ("P", "p", "T", or "t"), we will keep asking the question. Through the rest of the program, at each choice, we use the same mechanism to ensure a valid answer to the question asked.

- If the User chooses to view the Two-Player Comparison :



```
Please input two player's Last Names (beware of typos!) to compare their stats :  
Player 1 : Durant  
Player 2 : Duncle
```

Figure 2. Player Name input by the User

They will be asked to put in the last names of two players they choose. The names of the players must be without mistake (as there is no "closest match") and be written as in the table, or the graph for this player will return white. In this example, the two names in green have been chosen arbitrarily.

After writing in these names, we will get the following result :

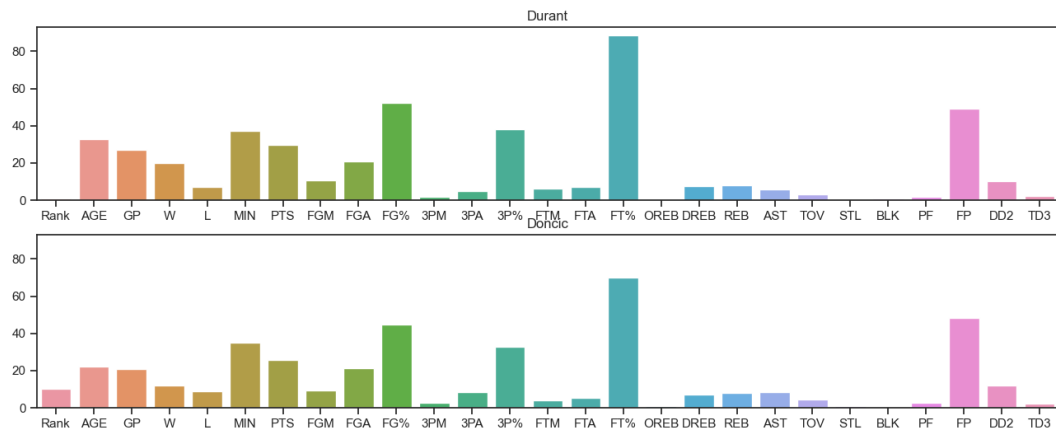


Figure 3. Side by side comparison of the stats of the selected Players

Note : if one of the last names appears more than once in the data, the corresponding graph will show the average of those players.

This graph will allow the User to compare immediately a player's stat, and compare if one or the other has obvious differences with the other. For example here we notice that Donic is ranked much lower than Durant.

- If the User chooses to view the Team Stats :

```
Do you want to view for (A)ll teams, or (S)elect the ones to view ?
*warning! viewing with all teams might be hard to read...
```

Figure 4. Choice of the Team selection mode

Either the user selects to view data for every team registered (because of the number of player in total, this option, it is not recommended by the warning) or select. In the case where we decide to keep a selection of teams, we get the following outputs :

```
Here are all the teams :
['BKN', 'ATL', 'MIL', 'GSW', 'CHI']
['DEN', 'BOS', 'LAL', 'DAL', 'UTA']
['LAC', 'MIN', 'PHI', 'MEM', 'NOP']
['WAS', 'PHX', 'POR', 'MIA', 'OKC']
['SAC', 'CHA', 'DET', 'TOR', 'ORL']
['IND', 'NYK', 'CLE', 'SAS', 'HOU']

(press enter after each team name,
and once more to finish selection)
Team names (3 letters) to add to Selection filter :
BKN
ATL
PHX
```

Figure 5. Team Selection Screen

After getting a list of the teams available, the user can input as many teams as he wants into the selection. If the 3 letter code doesn't correspond to an existing team, it will be ignored. From this point, the graphs will only display data relating to the teams entered in this selection.

This allows to reduce the amount of points shown on every graph, as well as to let the user decide what they want to view.

Once the selection has been made, we obtain 5 plots that open.

1. A blank figure

Opened by default by matplotlib when using the command “plt.show()”, this graph takes the title of the next opened graph (currently there is no found solution to this issue on our part)

2. Average Time in Game depending on Points Scored, by amount of Wins

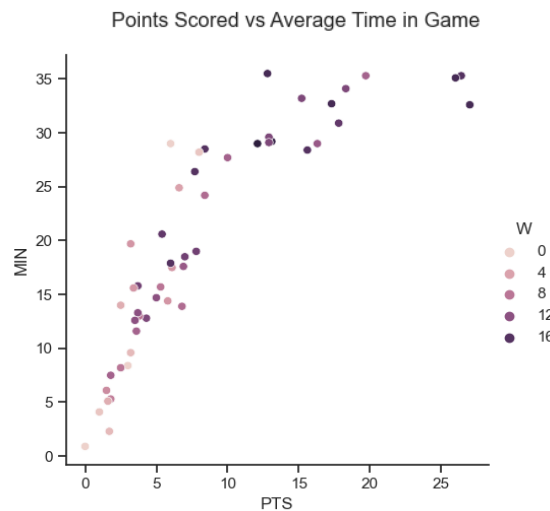


Figure 6. First graph -> PTS/MIN

This graph shows a strong correlation between those values, as we see that, for this selection of teams, players that score the less tend to spend the least time in games, and the amount of Wins seems related to the number of points scored strongly.

3. Regression Plot of the Number of points scored depending on Rank

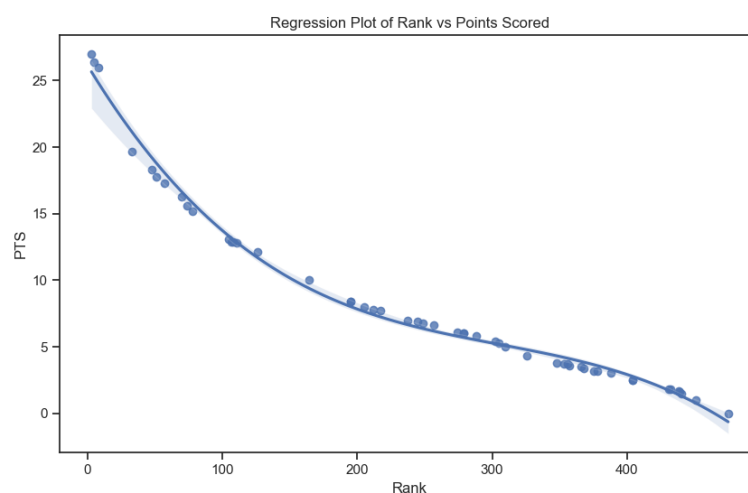


Figure 7. Second Graph -> 3rd Order Regression Rank/PTS

Here, we see a very strong correlation between the two variables, which is reinforced by the fact that a linear regression between the variables sticks very closely to the values. This allows us to see that for that selection (and in fact, overall), Rank and Points scored are very closely linked. Having the selection

and this graph, plus the previous one, allows us to see that the selection of teams we made have overall pretty low rankings, except a few outlying elements that are very highly ranked and score a lot.

4. Free Throws Made depending on Offensive Rebounds, by team

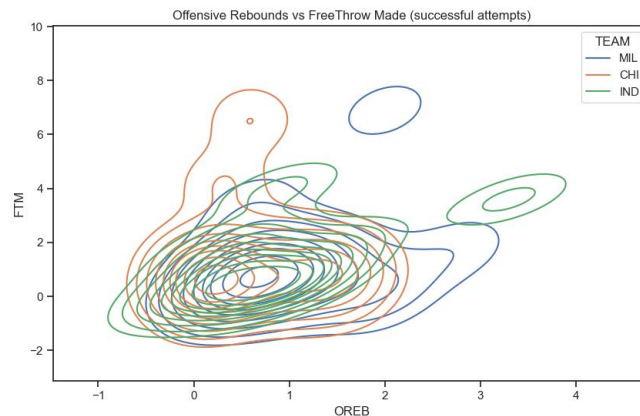


Figure 7. Third Graph -> Density Graph OREB/FTM

On this graph, we can see how the elements of the selected teams tend to be successful, or not, when attempting shots. In this specific graph it is useful to have a selection of only a few teams, as the number of teams quickly makes the graph very hard to read. However, we can still gather information about said teams : there is a high density point around (1;2) that players seem to concentrate around, despite some “branches” and outlying elements that display different characteristics.

On average, 1 offensive rebound means the player gets about 2 successful Free Throw.

5. Rank correlation to all other variables

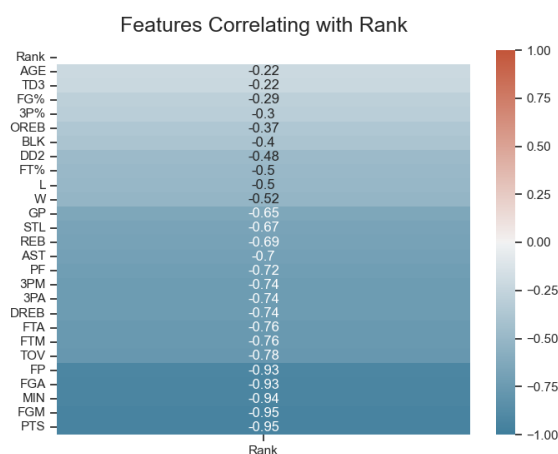


Figure 8. Fourth Graph -> Pearson Correlation Rank/Variables

Lastly, this is an informative and non-varying graph that shows how each variable correlates to the Rank of a player. This helps the user understand better how each player earns their position. Its values are negative, as having good scores makes your rank decrease in value (a higher rank obviously does not mean a higher number as a rank). We can observe for example that a player with good Fantasy Points (used for fantasy leagues by people) has a very good chance of being highly ranked as well, or that

scoring lots of points is the best way to get a high rank.

Finally, after closing all those plots, the player is presented with an opportunity to plot their own custom graph :

```
Do you want to make your own graph ? Y/N
```

Figure 9. Custom Graph decision by User

If they don't want to, the program is over and new choices can be made another time. But if they do, they get to choose which parameters to compare from a printed list (once again, a typo will result in a blank graph) and then the desired graph gets plotted as a standard scatter plot.

```
Do you want to make your own graph ? Y/Ny
['Rank', 'FirstName', 'LastName', 'TEAM', 'AGE']
['GP', 'W', 'L', 'MIN', 'PTS']
['FGM', 'FGA', 'FG%', '3PM', '3PA']
['3P%', 'FTM', 'FTA', 'FT%', 'OREB']
['DREB', 'REB', 'AST', 'TOV', 'STL']
['BLK', 'PF', 'FP', 'DD2', 'TD3']
Here are all the parameters, which two (beware of typos!) do you want to compare ?
Parameter 1 : GP
vs
Parameter 2 : OREB
```

Figure 10. Parameter choice screen

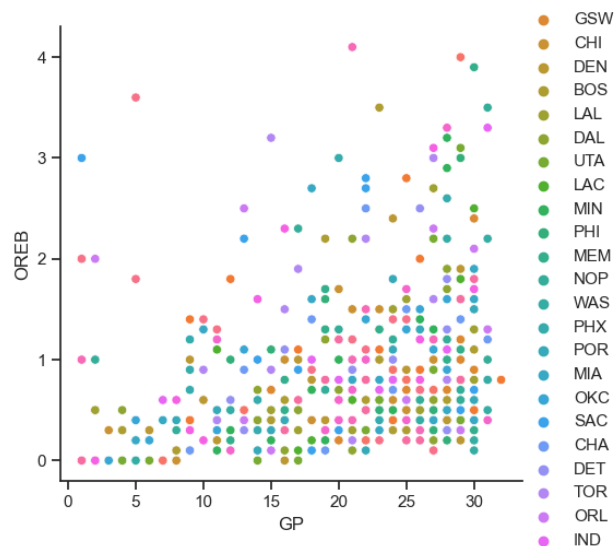


Figure 11. Custom graph example output

Here, the user decided to plot GP (Games Played) against OREB (Offensive Rebounds), whilst selecting all teams therefore they obtain the standard Seaborn scatterplot of these values.

II/ Decision Process

We decided to offer a program displaying stats by players as it is info that is harder to come by even by following matches closely, so having this program at hand could help a user to get precise info on a player, a team, and how their scores evolve, without having to keep a notebook next to their TV remote.

In order to decide which plots to offer the user, we used a correlation heatmap table, where every usable variable was correlated to every other:

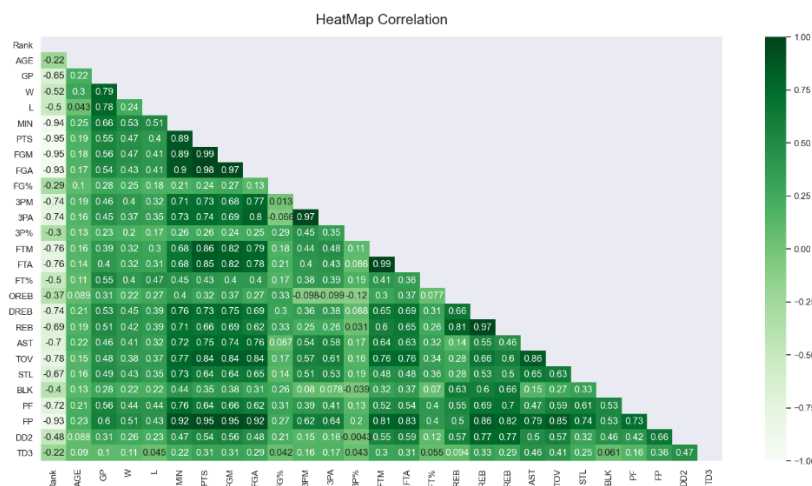


Figure 12. Heatmap Correlation graph of all variables

We can easily notice where items are correlated or inversely correlated: by a very dark or very light shade of green. Here we see for example that Wins and Losses are strongly correlated to the amount of Games Played (which is logical, as Games Played is the sum of the two), Time in game is somewhat related to the amount of Wins, but closely related to the number of points scored (as shown on the first graph of Team Stats), and so on. It can be noted that the “Features Correlating with Rank” graph corresponds to an organised version of the first column of this one.

The graphs were then chosen based on what interesting points they could offer to a user, and how interesting the correlation between the factors was to us. This way, we put into the selection of graphs things that were interesting from a data-analysis point-of-view, as well as interesting from a user experience point of view.