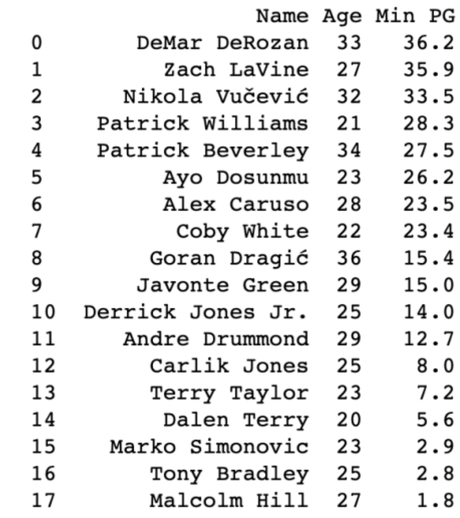
Cleaning, Engineering, and Visualizing, with Web Scraped NBA Advanced Statistics

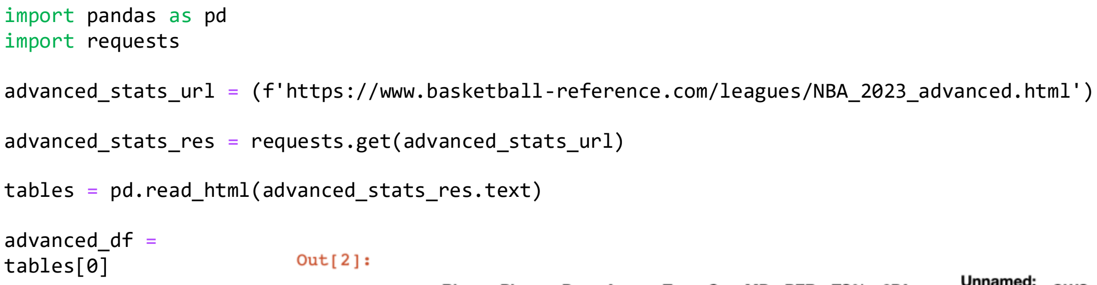
I've been curious about doing my own calculations and visualizations of NBA data for my own research but have been stunted due to my lack of access to public downloadable current NBA data. To ameliorate this issue, I went through a long process of figuring out how to web scrape public data from Basketball Reference and manipulate the data in my own local environment. Here are the steps I took to accomplish this task, and the ways I've been able to utilize the data since.

I started by just trying to scrape data from the Chicago Bulls. Originally, I made the job much tougher for myself, importing many libraries (BeautifulSoup, requests, pandas, re) and utilizing dense code. Additionally, I was writing functions to columns in the table based off attributes and tags, eventually creating a list of dictionaries which I would then convert to a pandas data frame. This took quite some time and required a lot of code, but eventually resulted in a usable pandas data frame. I ended up with this resulting table below.



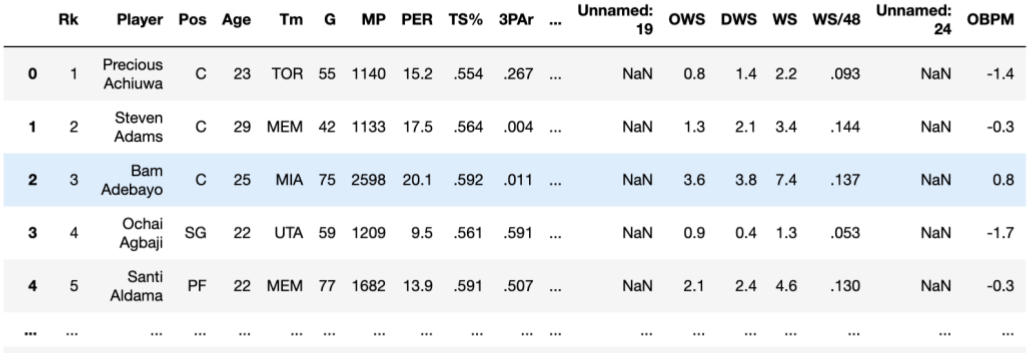
```
                 Name  Age  Min PG
0       DeMar DeRozan   33    36.2
1         Zach LaVine   27    35.9
2       Nikola Vučević   32    33.5
3     Patrick Williams   21    28.3
4     Patrick Beverley   34    27.5
5         Ayo Dosunmu   23    26.2
6         Alex Caruso   28    23.5
7          Coby White   22    23.4
8        Goran Dragić   36    15.4
9       Javonte Green   29    15.0
10  Derrick Jones Jr.   25    14.0
11     Andre Drummond   29    12.7
12       Carlik Jones   25     8.0
13       Terry Taylor   23     7.2
14        Dalen Terry   20     5.6
15    Marko Simonovic   23     2.9
16       Tony Bradley   25     2.8
17        Malcolm Hill   27     1.8
```

This first example only included a few columns, as I was still writing code for each column at the time. It displays each of the player's names, ages, and minutes per game. This was useful, I could continue to add columns if I wanted and go about running tests and creating visualizations. However, this is not practical if my goal is to analyze player data from the 2022-2023 season league wide. Thankfully, I came up with a much easier way to parse the HTML content from Basketball Reference. Instead of iterating through each row and storing a list of dictionaries, I instead wrote code to read the entire table on a page. To get closer to my end goal, I did this on the Advanced Stats page for all players for the 2022-2023 season. After importing pandas and requests, this new method only took a whopping four lines of code. Here was my code and resulting table:



```
import pandas as pd
import requests

advanced_stats_url = (f'https://www.basketball-reference.com/leagues/NBA_2023_advanced.html')

advanced_stats_res = requests.get(advanced_stats_url)

tables = pd.read_html(advanced_stats_res.text)

advanced_df =
tables[0]
```

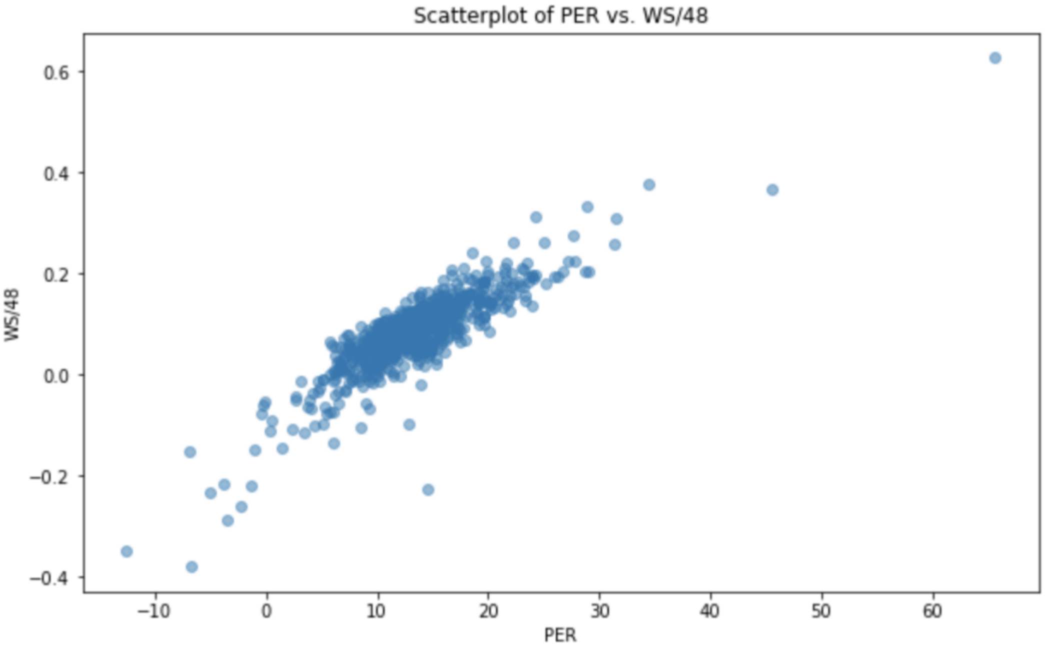
Out[2]:



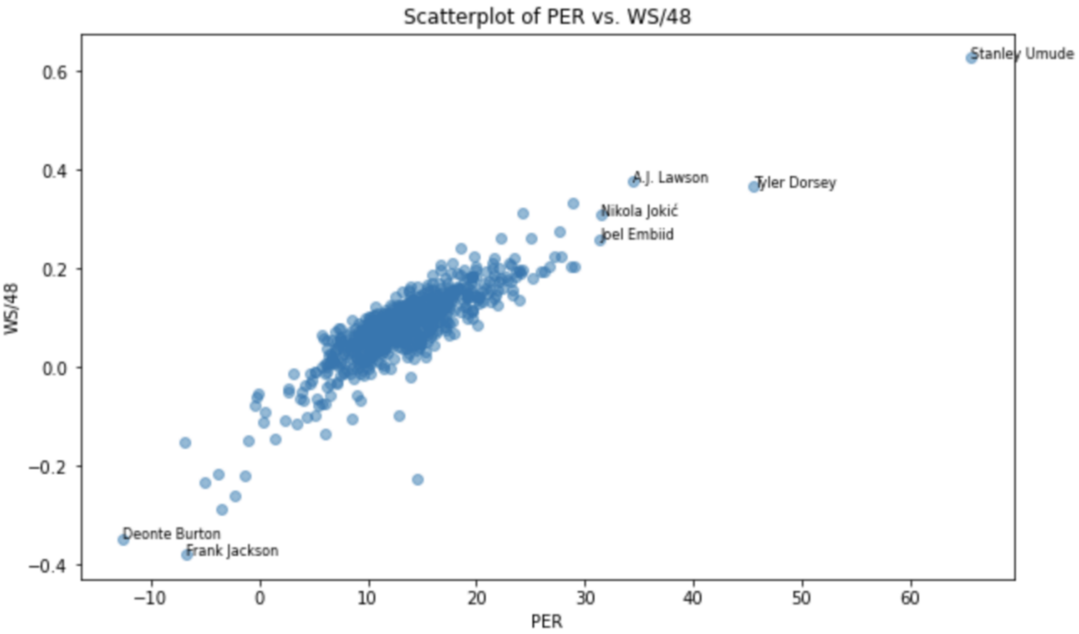
	Rk	Player	Pos	Age	Tm	G	MP	PER	TS%	3PAr	...	Unnamed: 19	OWS	DWS	WS	WS/48	Unnamed: 24	OBPM
0	1	Precious Achiuwa	C	23	TOR	55	1140	15.2	.554	.267	...	NaN	0.8	1.4	2.2	.093	NaN	-1.4
1	2	Steven Adams	C	29	MEM	42	1133	17.5	.564	.004	...	NaN	1.3	2.1	3.4	.144	NaN	-0.3
2	3	Bam Adebayo	C	25	MIA	75	2598	20.1	.592	.011	...	NaN	3.6	3.8	7.4	.137	NaN	0.8
3	4	Ochai Agbaji	SG	22	UTA	59	1209	9.5	.561	.591	...	NaN	0.9	0.4	1.3	.053	NaN	-1.7
4	5	Santi Aldama	PF	22	MEM	77	1682	13.9	.591	.507	...	NaN	2.1	2.4	4.6	.130	NaN	-0.3
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

This was quite exciting, I got the data frame I was looking for, with all players from the season included. Now I had to start the cleaning process. First, there are a couple columns listed here that are unnamed and don't contain any data, so I removed those first. Further inspection of the data showed that there were 3 unnamed rows that didn't contain any data, so I additionally dropped those rows. From here, I thought I was free to start making visualizations. This unfortunately was not the case. When I went to start making scatterplots using matplotlib, I was surprised by the results I was getting, they didn't seem to be following any sort of patterns that would make sense. When I tried to calculate the correlation coefficient, I realized that my columns were exclusively comprised of strings, not integers. The values that were being plotted were the values of strings rather than the value of the integers, which is why none of the plots made any sense. I wrote some code to convert all of the strings to integers instead for the columns that need it.

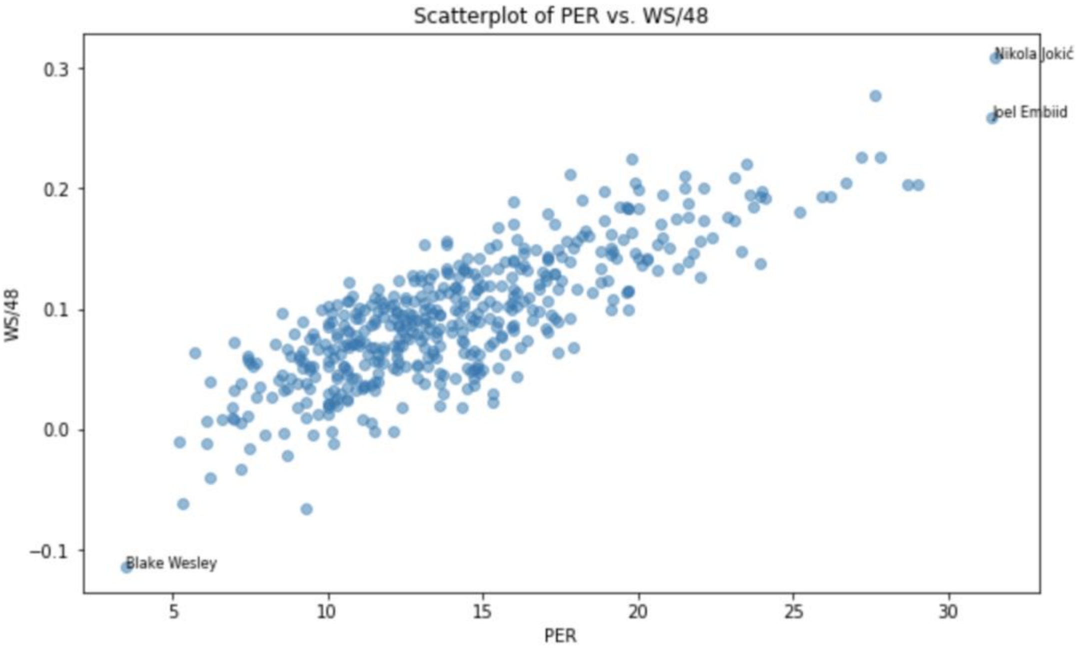
Now, I could make visualizations for all the data present in the table. I made a quick sample graph to start just to make sure things looked like they made sense, so I plotted something I expected to be strongly positively correlated, PER vs Win Shares per 48. The plot looked as I'd expected:



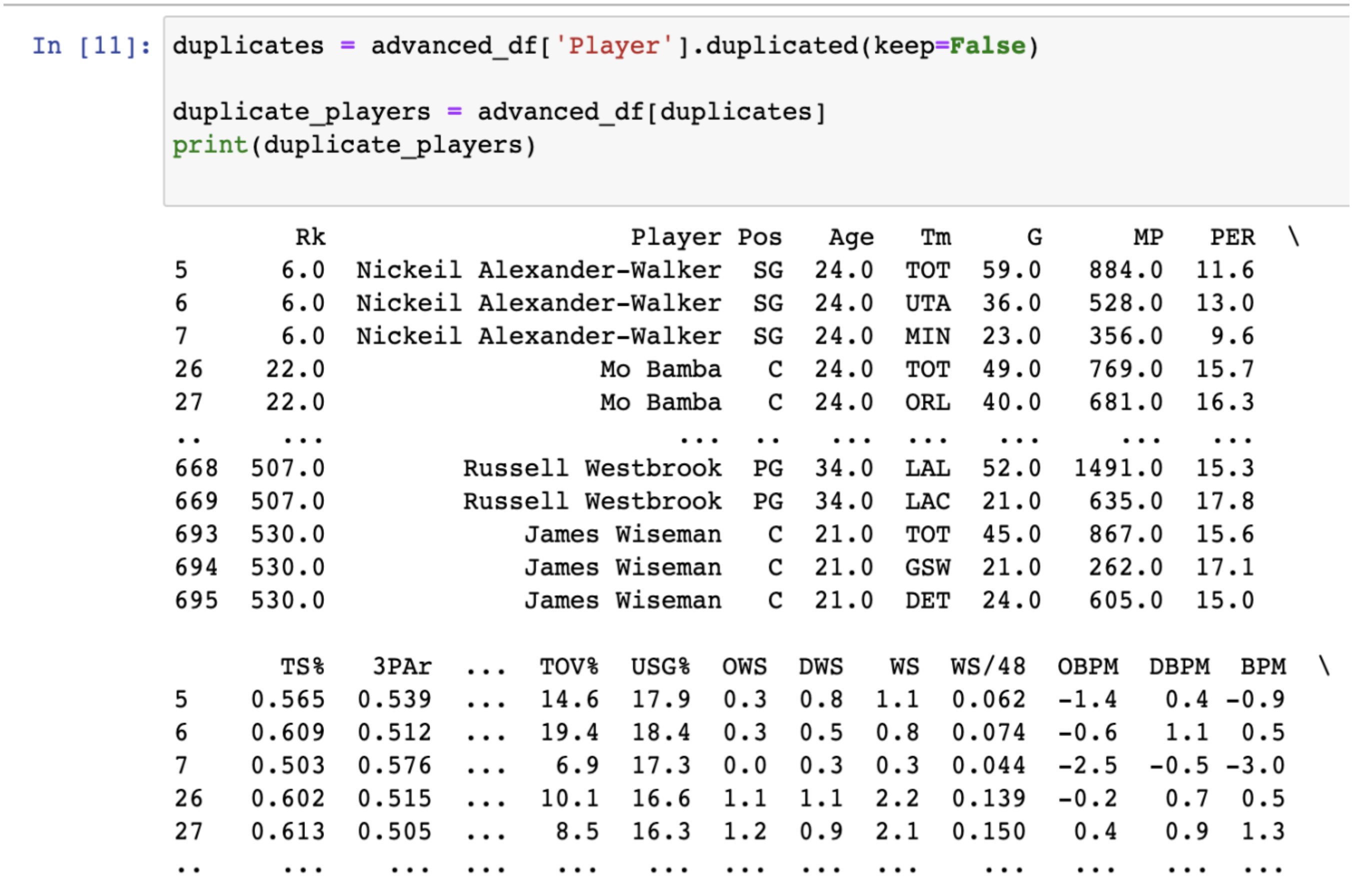
This looks great! Except, I'm curious about these outliers. At first I thought that top right data point must be Nikola Jokic, but even then I'm shocked that his numbers are that egregiously high. However, I wrote some code to get the names of the outliers and ended up with a new realization:



It turns out it is not Nikola Jokic, but instead Stanley Umude. Stanley Umude is a nice young wing but this exposes an issue with my data, players with low minutes and games played will contribute to outlier data that I do not want present in my visualizations or statistical tests. I decided to take out players that played less than 12 minutes a game, and players that played less than 20 games. This changed my data significantly, reducing my total rows from 702 to 455. My updated scatterplot makes a lot more sense:



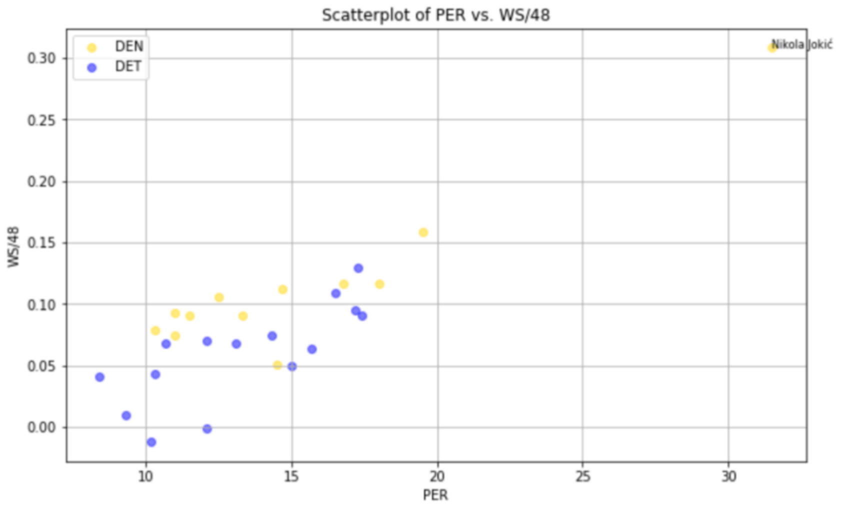
Now, there's a couple more things to accomplish. I was still a bit dubious about the number of rows, so I went through to double check if there were duplicate names, as I wasn't sure how to the data was handling players playing with different teams. Sure enough, there was loads of duplicate data:



```
In [11]: duplicates = advanced_df['Player'].duplicated(keep=False)

         duplicate_players = advanced_df[duplicates]
         print(duplicate_players)
```

```
        Rk                  Player Pos   Age   Tm     G      MP   PER  \
5      6.0  Nickeil Alexander-Walker  SG  24.0  TOT  59.0   884.0  11.6
6      6.0  Nickeil Alexander-Walker  SG  24.0  UTA  36.0   528.0  13.0
7      6.0  Nickeil Alexander-Walker  SG  24.0  MIN  23.0   356.0   9.6
26    22.0                  Mo Bamba   C  24.0  TOT  49.0   769.0  15.7
27    22.0                  Mo Bamba   C  24.0  ORL  40.0   681.0  16.3
..     ...                       ...  ..   ...  ...   ...     ...   ...
668  507.0         Russell Westbrook  PG  34.0  LAL  52.0  1491.0  15.3
669  507.0         Russell Westbrook  PG  34.0  LAC  21.0   635.0  17.8
693  530.0             James Wiseman   C  21.0  TOT  45.0   867.0  15.6
694  530.0             James Wiseman   C  21.0  GSW  21.0   262.0  17.1
695  530.0             James Wiseman   C  21.0  DET  24.0   605.0  15.0

       TS%   3PAr  ...  TOV%  USG%  OWS  DWS   WS  WS/48  OBPM  DBPM  BPM  \
5    0.565  0.539  ...  14.6  17.9  0.3  0.8  1.1  0.062  -1.4   0.4 -0.9
6    0.609  0.512  ...  19.4  18.4  0.3  0.5  0.8  0.074  -0.6   1.1  0.5
7    0.503  0.576  ...   6.9  17.3  0.0  0.3  0.3  0.044  -2.5  -0.5 -3.0
26   0.602  0.515  ...  10.1  16.6  1.1  1.1  2.2  0.139  -0.2   0.7  0.5
27   0.613  0.505  ...   8.5  16.3  1.2  0.9  2.1  0.150   0.4   0.9  1.3
..     ...    ...  ...   ...   ...  ...  ...  ...    ...   ...   ...  ...
```

As you can see in the printed output, the data includes each of their individual teams played for, and then has a total row as well. I'll only want the total row for the purposes of using all the players, but I'll want to include the player data for particular teams when inspecting individual teams. Therefore, I started by creating a dictionary that included all the individual teams' data frames so they could be inspected on their own. Then, I filtered duplicate data out from the main data frame, taking out individual teams from duplicated players and only keeping rows with 'TOT' values, which are the totals for those particular players for the season. My resulting data frame now has 376 players. This number makes a lot of sense, there are 30 teams with 15 players, but somewhere around 3-5 players on every roster that won't reach the games and minutes qualifications.



Now, I can do whatever I want with the data. Here's an example graph where I adjust the previous graph to only show the best and worst teams from last season, Denver, and Detroit (check out where Nikola Jokic). I can do analysis on team specific data or league-wide, and I can even import images of player headshots if I wanted that too. The NBA data for the season is clean and usable for whatever analyses I may want. In conclusion, I would like to point out that according to their terms listed on their website, scraping data from Basketball Reference is okay if you are not flooding their website with requests or using the data to create your own machine learning algorithms, or to feed to large language models. In my case, I am only using fairly small tables of data to create my own visualizations, with very limited requests.