Isaak Hedding

Exploring NBA Height Data with Sportradar API

Project Goals

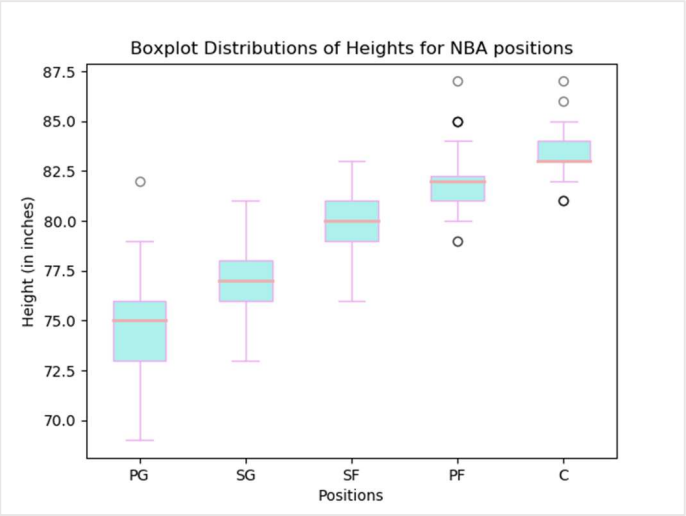
Being tall is a common stereotype for basketball players, but what is the standard for players in the NBA? In my final project, I set out to answer this question. I wanted to do a few things specifically:

1. Make a breakdown of NBA players' heights based on position
2. Make a breakdown of heights for all teams in the NBA
3. Find the average height for all players in the NBA

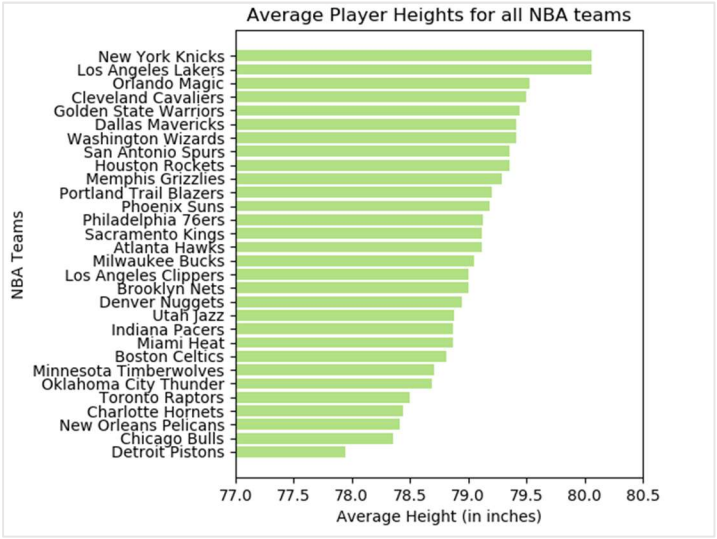
The first goal was my primary objective for the project. The second goal was an additional visualization I wanted to make. I figured the best way to achieve the first two goals would be to use side-by-side boxplots. A boxplot is useful for showing distributions of data in a set, such as the minimum value, the maximum value, the average value, and outliers. Hence, the first two goals would require visualizations. The third goal could be achieved using a simple average value calculation, and could be represented by a terminal output instead of its own visualization.

Project Goals Analysis

I was able to achieve most of the goals I set for myself. I successfully made a visualization of side-by-side boxplots distributions for heights based on position. I fetched data for all players in the NBA, including their name, the team they play for, their position, and their height. My code then selected all players for a given position and made a boxplot of this subset. This was done for all five positions (Point Guard, Shooting Guard, Small Forward, Power Forward, Center). The result can be seen below.



I was unable to make a visualization of side-by-side boxplot distributions of height for all NBA teams. I attempted to do so but the visualization was messy and difficult to read. Instead, I made a bar graph visualization that displays the average height for each team (shown below). I portrayed data about average height, but I needed to do it in a different format. Therefore, I was partially successful in achieving this goal.



I was also successful in calculating the average height in the NBA. As mentioned earlier, this did not require a visualization, but after running the code it can be seen in the terminal output.

Project Problems

Problem #1 – Finding an API

Initially, I wanted to use Yahoo! Fantasy Sports API to retrieve player performance data such as points per game, assists per game, and rebounds per game, to estimate their expected fantasy

points for a given day. However, the service was outdated and did not provide the data that I wanted. This was resolved by switching to the Sportradar API. While Sportradar offered the same data, they do not have their own version of Fantasy Basketball so I could not use the data to estimate fantasy points. Additionally, I was intrigued by the height data that the API offers so I decided to collect that data instead.

Problem #2 – Requesting the Data

I wanted to use a GitHub repository that contains code that accesses data from the Sportradar API. I installed the module using pip and implemented it in my code, but it did not function properly, as it attempted to access an older version of the API and would return an error.

The solution to this problem was ambiguous but was found deeper in the documentation for the API. The data that Sportradar outputs comes in a very clean XML format, and the documentation suggests using the <http.client> Python module to request it. The suggested module worked.

Problem #3 – Parsing the Data

Once the data was retrieved, I faced difficulty accessing the parts I needed. Being that it was in XML format, I wanted to use BeautifulSoup to parse it. It did not work initially, but was resolved by assigning <'xml'> to the <features> parameter, instead of <'html.parser'> which had been used in previous class-related applications of BeautifulSoup. This enabled me to use the .get() and .find_all() methods on the XML data.

Problem #4 – Displaying the Data

Overall, creating boxplots from the data didn't present any difficulties, aside from changing their color. However, formatting the bar graph proved to be a challenge. A normal bar graph showed the data in a messy way, and I was unsure how to display the data. I decided to use a horizontal bar graph which produced a relatively cleaner representation of the data. However, I needed to change the intervals of the x-axis (being that it was a horizontal bar graph). When the x-axis values started from 0, the disparity between team heights was difficult to observe. All NBA teams have an average height that is between 77 and 81 inches, and my x-axis needed to show only values in this range. This was resolved using the .xlim() method, and passing in 77 and 81 as the parameters .

Project Outcomes (i.e., the "report")

The calculated data can be seen in the terminal output. The code is able to show a breakdown on either height based on position or based on team, with a summary of important points from the data. For example, the summary of height based on position shows the maximum height for a player in that position, the minimum height for a player in that position, and the average

height for a player in that position. The summary of height based on team shows the team whose players have the highest average height and the team whose players have the lowest average height.

Interesting facts from the data (that is not directly shown in the summary):

- The tallest player in the NBA is 87 inches
- The shortest player in the NBA is 69 inches
- The position with the highest average height is Center (C)
- The position with the lowest average height is Point Guard (PG)
- Average height for position increases in the order: [Point Guard, Shooting Guard, Small Forward, Power Forward, Center]
- The tallest point guard is taller than the tallest shooting guard

Instructions for Running the Code

The only requirement for running the project is to run playerHeightProj.py in a terminal window. Running it will create two files: 'TeamIDCache.json' and 'NBA_players.sqlite'. Sportradar's API is layered in the sense that you have to request some data to get more data.

The first thing my code does is request the unique team IDs that Sportradar has for each team in the NBA. It then caches those IDs in 'TeamIDCache.json' so that they do not have to be requested again.

Next, it creates the database 'NBA_players.sqlite' if it does not already exist. Then, it uses each team ID to request the team's roster data from Sportradar. This contains player data, which is then stored in the database.

Once the data has been collected, the code will ask for user input. To view the primary visualization that is the boxplot distributions of heights by position and its associated terminal output, enter '1'. To view the additional visualization that is the bar graph distribution of average heights in the NBA and its associated terminal output, enter '2'.

**Important note: you must enter '1' first and create the database. Entering '2' first without the database created will lead to an error. Run the code, enter '1', then run it again and enter '2'.*

When viewing the bar graph distribution, please adjust the size of the output window using the 'Configure subplots' feature that is found on the navigation bar at the bottom of the window. Some of the team names are quite long so they do not fit in the initial window that is created. On my computer, I needed to adjust the 'left' configuration to 0.33 and the 'top' configuration to .95.

Function Documentation

getTeamIDs(api_key, cacheDict, fileName)

Inputs: a Sportradar API key (string object), a dictionary, and a file name

Function: takes in a dictionary that contains (or will contain) cache data of the unique Sportradar team IDs. It then checks length of the dictionary. If its length is 0, there must be no cache data in it, so the function uses the API key to make a request to Sportradar and fetch the team IDs. These team IDs are then placed in the dictionary, with the team name as the key and the associated unique team ID as the value. This dictionary is then written to the file that was passed in as the third parameter. If the length of the dictionary is greater than 0, the team IDs must already be cached in the file. The function returns the cache dictionary and creates a file with cache data if one did not already exist.

getPlayerInfo(api_key, teamIDDict, conn, cur)

Inputs: a Sportradar API key, a dictionary with team names and their unique ID, an SQL connection object, an SQL cursor object

Function: the function first checks the SQL cursor object, creating a table that will contain player data if one does not already exist. It then checks the length of the database. If the length is 0, it means that there must not be any player data stored in it. The function will then make a request to Sportradar to get the rosters for all teams that are in the teamIDDict (this is the dictionary that was returned from the preceding getTeamIDs function). It makes one request to Sportradar per team, so there are 30 requests total (this takes approximately 2 minutes). Player data is extracted from each response, including their name, the team they play for, their position, and their height (in inches). This data is then inserted into the database. If the length of the database is greater than 0, player data must already be in the database, so no requests are made. The function does not return anything, it just creates the database if it does not already exist.

getPositionHeightInfo(cur)

Inputs: an SQL cursor object

Function: the function first creates a list containing the five positions in basketball – Point Guard, Shooting Guard, Small Forward, Power Forward, and Center – as well as an empty dictionary. For each position, it selects from the database all players who play that position, takes their height, and appends it to a list. This list is then inserted as a value into the empty dictionary with the position as the key. The function then returns the dictionary. The keys are

the five positions, and the values are lists containing the heights of every player in the NBA who plays that position.

makePositionDistribution(dict):

Inputs: the dictionary that was created in getPositionHeightInfo

Function: the function takes in the dictionary that was created in getPositionHeightInfo. For each position in it (the keys), it calculates from its associated list of heights (its value), the maximum height, the minimum height, and the average height. This information is displayed in the terminal output when it is run. This function also creates a visualization of side-by-side boxplots representing a distribution of heights for each position.

getTeamHeightAvg(teamIDDict, cur)

Inputs: the cache dictionary returned from getTeamIDs, an SQL cursor object

Function: the function first gets all team names from the dictionary returned from getTeamIDs and assigns them to a list. It then creates an empty dictionary. For each team in the list, it selects from the created database all players who play for that team, gets their heights, and appends all their heights to a list. This list is then inserted into the empty dictionary as the value with the team name as the key. The function then creates another empty list. For each team in the dictionary (the keys), it calculates the average height from its associated list of heights (its value), and creates a tuple containing the team name and this average height. Each tuple is then appended to the empty list. The function then returns this list of tuples.

makeTeamHeightDistribution(tupList)

Inputs: a list of tuples (the one created in getTeamHeightAvg)

Function: the function first sorts the list of tuples based on the average height. The function then prints the team with the highest average height, the team with the lowest average height, and the average height for all teams. Next, it then makes two separate lists from the sorted list of tuples: one that contains all the team names from the tuples, and one that contains all the average heights from the tuples. These lists are used as parameters to make a bar chart for each NBA team and its average height.

runAll(api_key)

Inputs: a Sportradar API key

Function: this is the 'main' function that calls all other defined functions. It asks for user input. If the user enters '1', the function will make a distribution of heights based on position, as in it uses the getPositionHeightInfo and makePositionDistribution functions. If the user enters '2', the function will make a distribution of heights based on team, as in it uses the getTeamHeightAvg and makeTeamHeightDistribution functions.

**Must enter '1' first and create the database before being able to enter '2'*

Resources

Date	Issue Description	Location of Resource	Result
12/5/18	Needed a way to access Sportradar API and fetch data	GitHub https://github.com/johnwmillr/SportradarAPIs	No, the functions were not successful in accessing Sportradar API
12/7/18	Still needed a way to access Sportradar API and fetch data	Sportradar API documentation https://developer.sportradar.com/docs/read/basketball/NBA_v5#team-profile-rosters	Yes, the documentation Sportradar provides for accessing its API solved the problem
12/9/18	Sportradar only handles one request per second, needed a way to slow down code	Stack Overflow https://www.pythoncentral.io/pythons-time-sleep-pause-wait-sleep-stop-your-code/	Yes, the time module enabled me to create a delay in my code while its running
12/12/18	Difficulty accessing data in database using a for loop ("incorrect number of bindings supplied" error in sqlite)	Stack Overflow https://stackoverflow.com/questions/16856647/sqlite3-programmingerror-incorrect-number-of-bindings-supplied-the-current-sta	Yes, the addition of a comma after passing in parameter values enabled me to access SQL data
12/13/18	Needed to know how to create boxplots using matplotlib	Matplotlib Documentation https://matplotlib.org/api/_as_gen/matplotlib.axes.Axes.boxplot.html#matplotlib.axes.Axes.boxplot	Yes, the matplotlib documentation was useful in creating boxplots
12/13/18	Needed to know how to change the color of boxplots	Knowledge Stockpile http://blog.bharatbhole.com/creating-boxplots-with-matplotlib/	Yes, this blog instructed me how to change the color of boxplots in matplotlib

12/14/18	Needed to know how to create bar graphs in matplotlib	Matplotlib documentation https://matplotlib.org/api/_as_gen/matplotlib.pyplot.bar.html#examples-using-matplotlib-pyplot-bar	Partially, the data created from this was messy and difficult to read, but a bar graph was created
12/14/18	Needed to know how to create bar graphs in matplotlib	Matplotlib documentation https://matplotlib.org/api/_as_gen/matplotlib.pyplot.barh.html#matplotlib.pyplot.barh	Yes, the matplotlib documentation on horizontal bar graphs enabled me to represent team height data in a cleaner format than vertical bar graphs
12/15/18	Needed to change the interval of the x-axis for my horizontal bar graph	Matplotlib Documentation https://matplotlib.org/api/_as_gen/matplotlib.pyplot.xlim.html	Yes, using .xlim() method, I was able to change the interval