Data Visualization Project — Baseball

Due March 25rd, April 5th, April 10th, April 17th, April 19th at 11:55pm

(10, 65, 75, 85, 15 points)

Updated March 23rd

Objectives:

- Use the skills that you have acquired thus far to design a program in a modular way
- Become familiar with libraries that are commonly used for conducting data analysis in python
- Implement a larger program that works with real-world data

Turn In:

- (March 23rd) Fill out quiz on moodle.
- (April 5th) `baseball.py` (must be implemented up until *at least* the end of task 2)
    - `.pdf` files of saved versions of all of the graphs that you have produced (`lifetime_runs.pdf`)
- (April 10th) `baseball.py` (must be implemented up until *at least* the end of task 3)
    - `.pdf` files of saved versions of all of the graphs that you have produced (`lifetime_runs_positions_0.pdf`, `lifetime_runs_positions_100.pdf`, `team_lifespans.pdf`)
- (April 17th) `baseball.py` (must be implemented up until *at least* the end of task 4)
    - `.pdf` files of saved versions of all of the graphs that you have produced (`obp_best.pdf`, `homeruns.pdf`, `team_rbis.pdf`)
    - For extra credit, #1 should be turned in as `ec_analysis.pdf`, #2 - 4 should be included and commented in `baseball.py`
    - If you save new graphs for the extra credit, turn these in as well (make sure that they have different names from the requested graphs).
- (April 19th) `reflection.pdf`

Turn these in from the corresponding links on moodle.

Your code must be commented at each deadline that involves turning in code. Your code must follow the guidelines set forth in previous homeworks in terms of style and structure.

What is given:

You are given the file battingData1950Present.csv. This is a comma separated file with 74,501 rows. It has one row of headers and the rest of the rows correspond to statistics for a given player during a given season with a given team.

battingData1950Present.csv

yearID: This is the year that the stats correspond to
playerID: a unique player id for each player

nameFirst: the player's first name
nameLast: the player's last name
name: this is the team name
Games: the number of games that the player played in
AB: at bats
R: runs
H: hits
doubles: number of doubles
triples: number of triples
HR: number of homeruns
RBI: number of runners batted in by this player
BB: walks
HBP: number of times hit-by-pitch
stolenBases: number of bases successfully stolen
caughtStealing: number of times the player was caught when trying to steal
SO: strike-outs
sacFly: number of sacrifice flies
position: a code for the position that the player played (e.g. P = Pitcher, C = Catcher, 1B = first base)

Notice that 1) not all columns are the same type and 2) one column ("sacFly") doesn't always have a value. We'll have to deal with this later.

battingData1950Present.csv (first 5 rows)

```
yearID,playerID,nameFirst,nameLast,name,Games,AB,R,H,doubles,triple
s,HR,RBI,BB,HBP,stolenBases,caughtStealing,SO,sacFly,position
1950,aberal01,Al,Aber,"Cleveland
Indians",1,2,0,0,0,0,0,0,1,0,0,0,1,,P
1950,abramca01,Cal,Abrams,"Brooklyn
Dodgers",38,44,5,9,1,0,0,4,9,0,0,0,13,,OF
1950,adamsbo03,Bobby,Adams,"Cincinnati
Reds",115,348,57,98,21,8,3,25,43,0,7,0,29,,2B
1950,adamsbo03,Bobby,Adams,"Cincinnati
Reds",115,348,57,98,21,8,3,25,43,0,7,0,29,,3B
```

BattingData object:

We provide you with the following code that will help the legibility of your project by giving names to column numbers. Copy + paste this code into the top of your file (below your import statements). See lecture 19 for examples of how to use a similar object.

```
class BattingData:
```

```
year = 0
player_id = 1
first_name = 2
last_name = 3
team_name = 4
games = 5
at_bats = 6
runs = 7
hits = 8
doubles = 9
triples = 10
home_runs = 11
rbi = 12
walks = 13
hbp = 14
stolen_bases = 15
caught_stealing = 16
strike_outs = 17
sac_flies = 18
position = 19
```

Task 1: Your Plan (Sunday, March 25th at 11:55pm)
Read through the rest of this write-up. Before you begin implementing this project, write down a plan of how you are going to accomplish each step. This can be pseudocode. This can be a list of tasks. Do this in the format that makes the most sense and will be most helpful to you. This should be at least 1 page/ 50 lines long.

Task 2: Read the data and create the user interface (Thursday, April 5th at 11:55pm)
2a. main() function

- Always begin your code by writing out the main() function.
- From your main() call a function named *read_data()* that takes no argument but returns a value (of type - numpy array)
- When you have read in your data, this is a great place to check its dimensions.

2b. read_data(): How should we read the data in the csv file?
Because our visualization requires numerical computations, you are required to save the data in a *numpy array*.

- You will first have to provide the path to where your “data” folder is residing on your computer.
- Read all the batting data from all the files in the “data” folder.
- *In what mode will you open these files at this step?*
- You can use the python library named “*csv”* to read csv files if you want to. Since our data is quite clean, this isn’t necessary.

More information on how to use python csv library can be found here:
https://docs.python.org/3/library/csv.html
http://www.pythonforbeginners.com/csv/using-the-csv-module-in-python
https://pythonspot.com/reading-csv-files-in-python/

- You need to read in every row from the csv files, append it to a list and convert this huge list into a huge numpy array. You final data will be of type <*class 'numpy.ndarray'*>. You should double check to make sure this is the case.
- numpy requires that all the data in the array be the same type—since we have both words and numbers, what type should we make the values in our array?
- Make sure to ignore the headers when you read the data
- You need to return this numpy array to the main( ) function

See Appendix A for verifying the structure of your numpy array.

2c. Make sure all columns of our array have values, ensure that your data is of type string
After you have read in your data, take a look at some of the rows of your numpy array. All columns should have values and all columns should be of type np.str.

Make sure you understand how to access each column of your array using the variables in the BattingData object (provided).

When you need to treat your data as integers in later tasks, you will use the numpy .astype conversion function to help you turn it into type np.int32. (see the numpy document for more on this)

2d. Implement your User Interface (UI)
After reading the data the next step to would be to provide a user interface so that the user can choose what they would like to do with the given data!

Define a function named *get_menu_choice()* that takes a list of strings (the options that the user can pick between) and returns the choice entered by the user after making sure that it is a valid choice. This is very similar to the interface in Homework 7 - Calculator.

If the user enters an invalid choice, the menu should be displayed again until the user inputs a valid choice or enters a 0, which will exit the program.

In the final version of this project, the user will be able to choose a number and see the results of their choice (discussed in later tasks). After executing the functions required for their choice, this menu will be displayed over and over again until the user enters a “0” and exits the loop.

<u>At this point, you don’t need to have the functionality behind the options implemented, but your menu should otherwise work. We recommend using print statements to make sure that your program is correctly routing the user’s choice.</u>

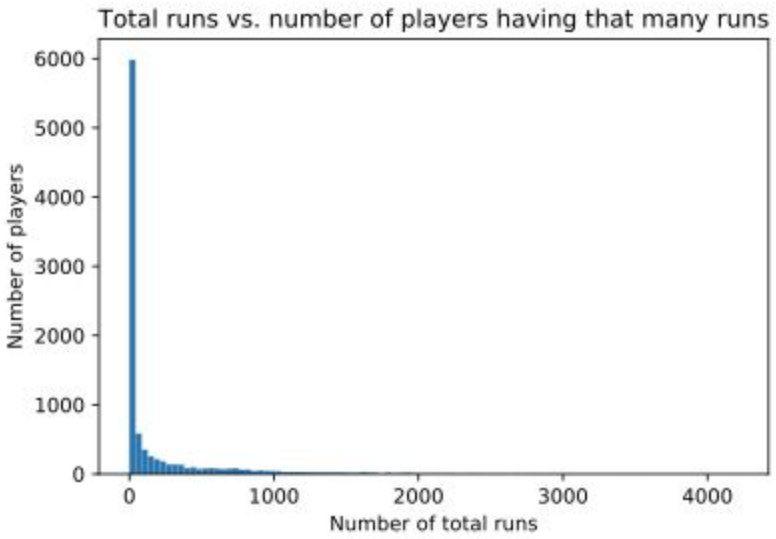
Example:

```
0: Exit
1: Histogram of runs scored in the lifetimes of all players (no cutoff, no
positions)
2: Histogram of runs scored in the lifetimes of all players (no cutoff)
3: Histogram of runs scored in the lifetimes of all players (cutoff = 100)
4: Graph team presence over time
5: Find the batters that had the best and worst seasons stealing bases
6: List the 10 best seasons that batters had by on base percentage
7: Plot on base percentage for the lifetimes of players with best seasons
8: Plot homeruns over time (percentiles)
9: Plot average team RBI over time (1950 - 1959)
10: Extra Credit

> 1
```

2e: Histogram of runs scored in the lifetimes of all players (no cutoff, no positions)
Here, you will be creating a histogram with 100 bins for the number of runs scored in the lifetimes of all players. For each player in the dataset, you will need to sum their runs to calculate their "lifetime runs". Then, you will take all of the total run values, and give these to matplotlib's hist() function.

To accomplish this, you will find numpy's unique() and where() functions useful. Remember that when you are summing, you should be treating your data as type np.int32!



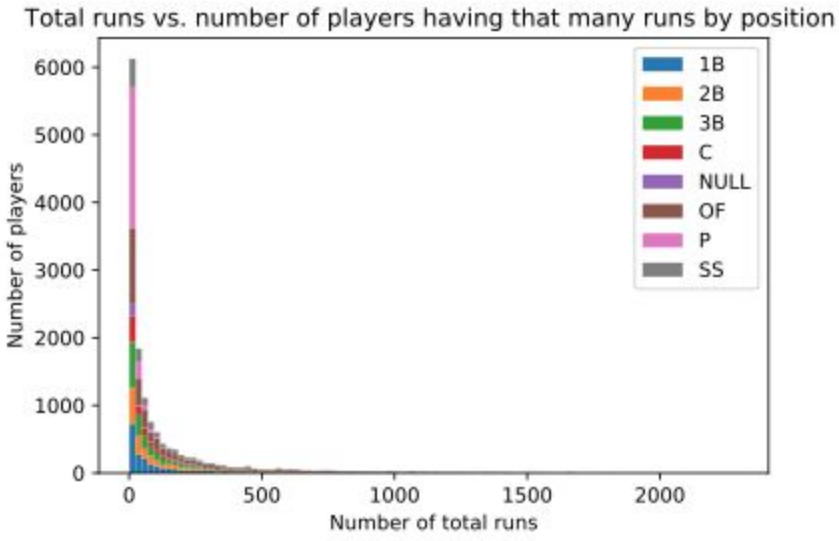
Task 3: Plot some graphs (Tuesday, April 10th at 11:55pm)

For this task, you will be implementing the functionality for options 2 through 4.

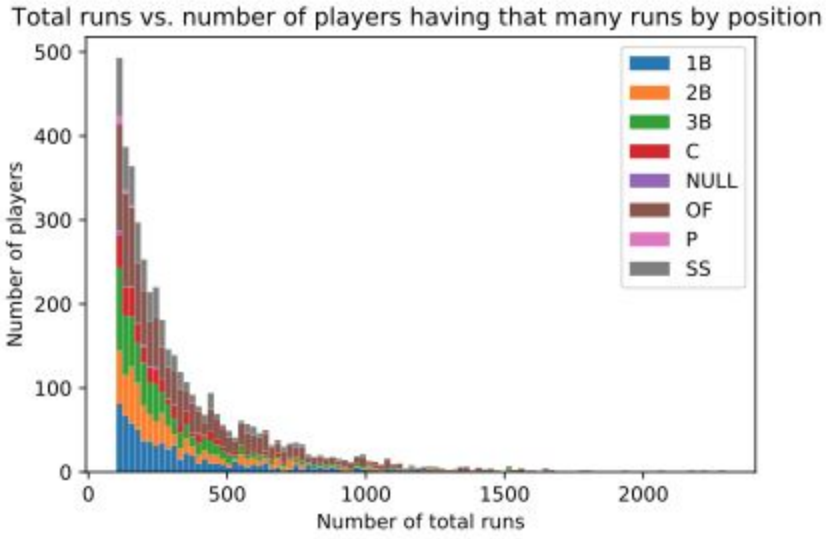
3a: Histogram of runs scored in the lifetimes of all players (no cutoff)
Your next task is the same as number 3a, but this time, we will separate the data based on the position of each player. We would like to group all Pitchers together, all Catchers together, all Short Stops together, and so on and so forth.

For each player having each position, you will do as you did in 3a and sum their runs over their lifetimes. Once you have all the lifetime runs for all the different positions, you can once again use matplotlib's hist() function to create a stacked histogram. You should continue to use 100 bins.

You will see that some players have position "NULL". These are season/player combinations for which we are missing the data. You don't need to worry about this.



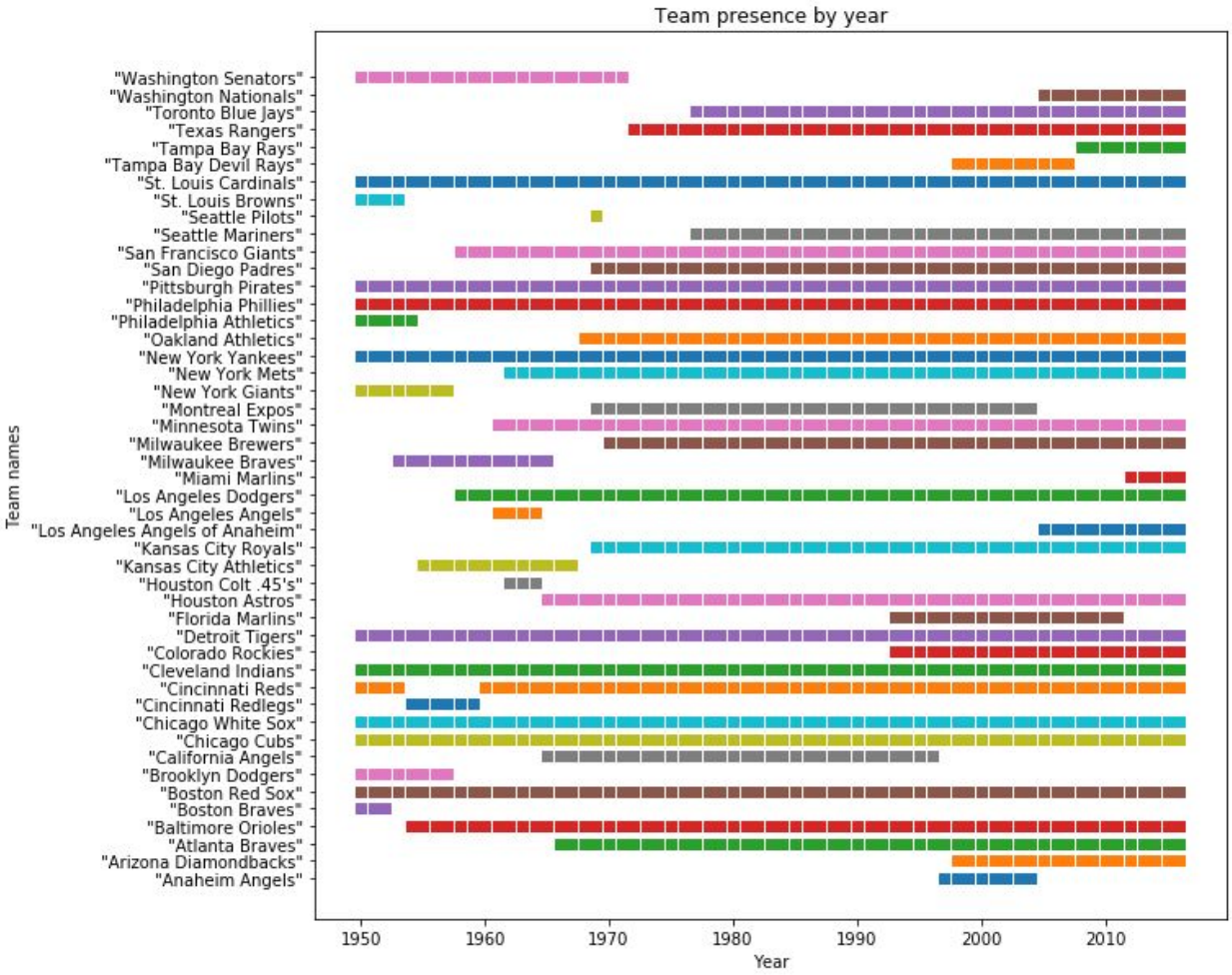
3b: Histogram of runs scored in the lifetimes of all players
Your next task is the same as number 3b, but this time we will impose a cutoff, meaning that we will only be graphing the data from players whose total lifetime runs value is greater than the cutoff. We've shown you a graph with cutoff set to 100, but you should design your function so that you can pass in any number as the cutoff.



3c: Graph team presence over time

For this task, you will be creating a scatter plot showing when each team was active. The y-axis will be labeled with team names and the x-axis will be the years that they were active. A team is considered to be active during a given year if there is any player from that team with statistics for that year. It doesn't matter to us what order the teams are in, but the data that you graph should be correct.

Feel free to use whatever marker you like best in your scatter plot. Make sure to look at the examples for creating scatter plots in the matplotlib document.



Task 4: Plot the rest of the graphs (Tuesday, April 17th at 11:55pm)

For the final task, you will be implementing the functionality for options 5 - 9 and implementing any of the extra credit.

4a: List the 20 best seasons that batters had by on base percentage

To retrieve this information, you will calculate on base percentage for the players in our data set. On base percentage is: (hits + walks + hit by pitch) / (at bats + walks + hit by pitch + sacrifice flies). We will only consider players that both have more at bats than the 25th percentile of all players/seasons for at bats and more games played than the 25th percentile of all players/seasons for games played.

Once you have calculated on base percentage for each player, you will need to use numpy's argpartition function to find the indices corresponding to the players with the best stats. We're asking you to find the 20 best seasons, but you should design your function such that you could use it to find the 10 best seasons or the 50 best seasons as well.

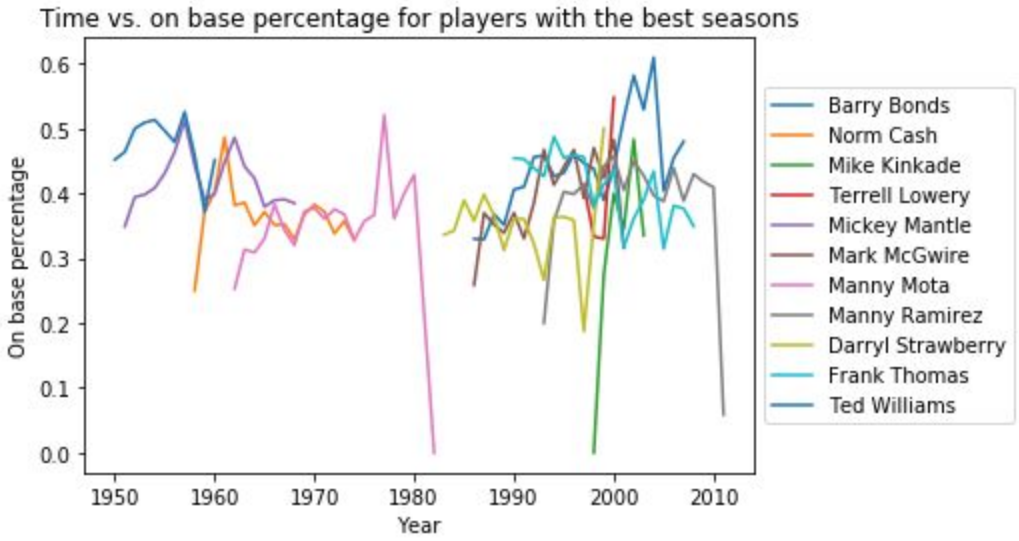
Think about how you can debug your code and look at the graph for task 4b to determine whether you are finding the correct players!

You can choose the format to print this information out in, but you must include enough information to uniquely identify the player/season combination.

4b: Plot on base percentage for the lifetimes of players with best seasons
The first step of this task will be, given the output of 4a, find the unique players that these seasons correspond to.

Then, create a timeseries graph where, for every player who had one of the top 20 seasons, you graph a line with one data point for every year that they were active. Your x values will be years and your y values will be on base percentages.

Warning: some players switch teams part way through a season. You didn't need to worry about this for previous graphs, but for this one, if a player has more than one row in a given year, we want to calculate their on base percentage as follows: (sum(hits) + sum(walks) + sum(hit by pitch)) / (sum(at bats) + sum(walks) + sum(hit by pitch) + sum(sacrifice flies)). Where sum indicates that we are adding all values together from the different rows.



4c: Find the players that had the best and worst seasons stealing bases
Here, for all players that have played more games than the 25th percentile of players/seasons, we'll be calculating two things: their "steal ratio" (number of bases stolen / number of games played) and their "failed steal difference" (number of bases their were caught stealing - number of bases successfully stolen). Once we've calculated each of these, we'll take the player/season with the maximum "steal ratio" and declare them the best stealer, and the player with the maximum "failed steal difference" and declare them the worst stealer.

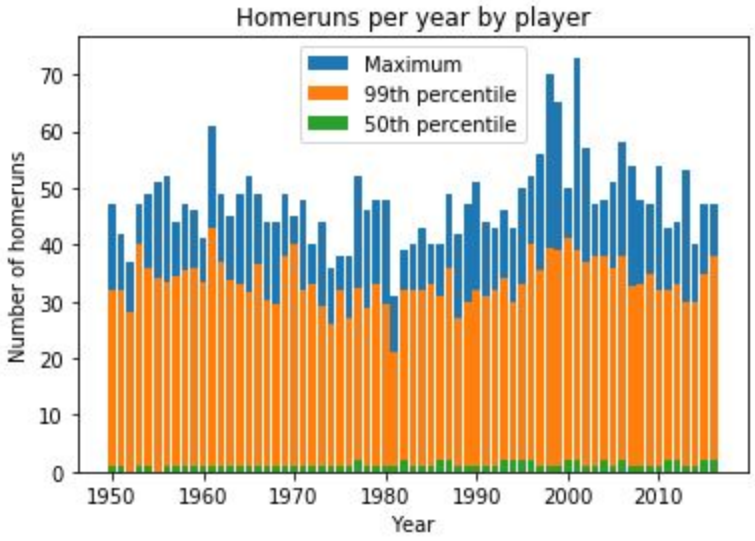
You will find numpy's argmax function useful here.

Again, think about ways that you can debug this yourself. How can you test to see if you are finding the correct players?

You can choose the format to print this information in, but you must give information about who the player is, what season you are talking about, how many games they played, how many bases they successfully stole (for best and worst), and how many bases they were caught stealing (required for worst).

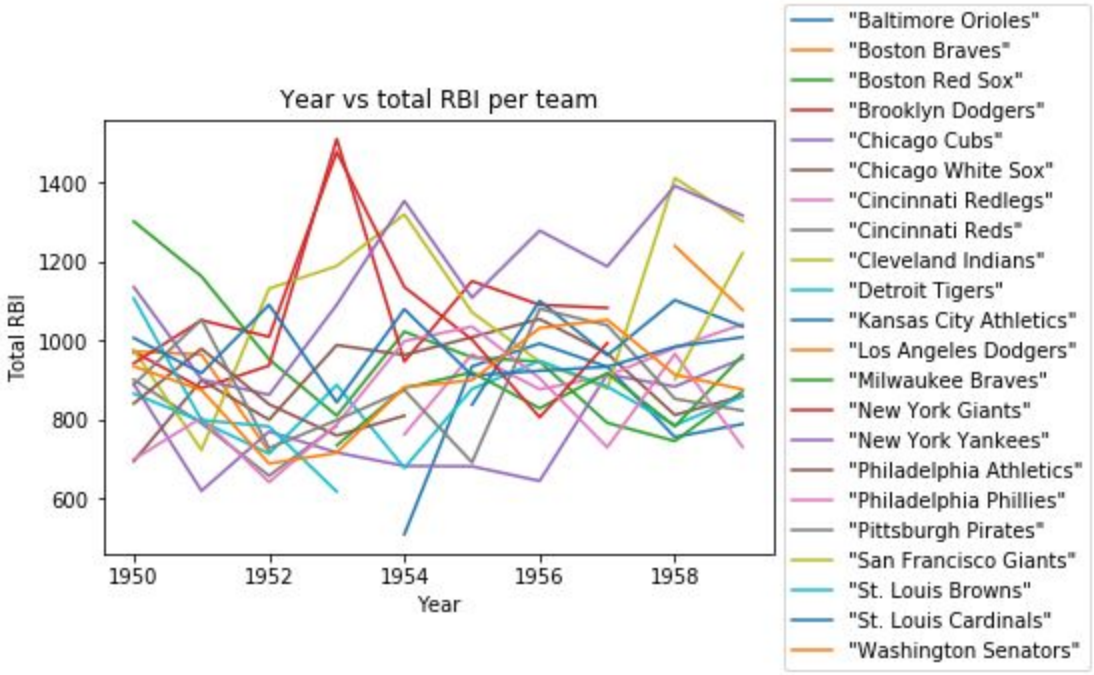
4d: Plot homeruns over time (percentiles)

For this task you will create a bar graph with the following series (graphed in this order) : the maximum number of homeruns scored every year, the 95th percentile of # of homeruns scored every year, and the 50th percentile of homeruns scored every year. Take a moment to think about what this data tells us.



4e: Plot total RBI for teams over time

For this task you will look at only the data from one decade. You will calculate total rbis for a team for each year as the sum of the rbis of the individual players on that team in that decade.



Extra Credit (up to 30 points):

1) (up to 15 points) Analyze your graphs. From each graph, what conclusions can we make? For each graph, is there anything that we could do to improve readability?
2) (up to 5 points) Modify your program so that the user can choose what decade they would like to analyze. For example, if they are analyzing the 1960s, all graphs and statistics (including graph 9, total RBI for teams over time) should only show information about that decade.
3) (up to 5 points) Make the 9th graph more legible. You can do this by modifying the graph parameters, by restricting the graph to only plot a certain number of teams, etc. The choice is yours. Make sure to comment every change that you make.
4) (up to 5 points) Define your own visualization task. Include in the comments the significance of this task. Make sure that this is sufficiently different from all other visualizations.

Task 5: Reflect on the project (Thursday, April 19th at 11:55pm)

Objectives:

- How did you prepare for this project?
- Did you complete Task 1 by writing a plan? Was it useful? How?
- Was working with a partner helpful? Why or why not?
- Reflect on how you could have done better, or how you could have completed the project faster or more efficiently

What you need to do:

Write a one-page essay answering the following question, in the context of the Data Visualization Project for this class:

Did your group/team have any false starts, or begin down a path only to have to turn back when figuring out the strategy/algorithm for your project? Describe in detail what happened, for example, what specific decision led you to the false starts, or, if not, why do you think you progressed smoothly, and give a specific example. If you worked alone, the question still applies.

Finally, if there is something that you would change about the project if you could, what would it be?

Note: all reflection papers should be individual. This is not a team assignment, even if you worked with a partner.