Data Visualization Project — Baseball

Due November 14th, November 21st, November 29th, and December 6th at 1pm

(10,60, 50, 50 points)

Objectives:

- Use the skills that you have acquired thus far to design a program in a modular way
- Become familiar with graphing and data visualization in python
- Implement a large program that works with real-world data

Turn In:

- November 14th: `pseudocode.pdf`
  - Project plan with pseudocode or list of tasks.
  - Choice of partners (optional, maximum team size is 3)
  - pseudocode (Task 1)
- November 21st: `baseball.ipynb`
  - reading data; utility functions (Task 2)
  - **All partner choices at this point are final. Maximum team size is 3!**
    - If you have changed partners in any way between submitting Task 1 and Task 2, you **must** submit a change of partner request to soch7759@colorado.edu.
- November 29th: `baseball.ipynb`, pdfs of the 4 graphs that you produced
  - graphs 1 - 4 (Task 3)
- December 6th: `baseball.ipynb`, pdfs of the 4 graphs that you produced, plus pdfs of any graphs you produced for the extra credit.
  - graphs 5 - 8 (Task 4)
  - Extra credit

Turn these items into the corresponding links on canvas.

Your code must run without errors at each deadline. **You may not turn in any tasks late.** Your code must follow the guidelines set forth in previous homework in terms of style and structure.

You may work with a partner if you choose to. **You and your partners (if you have any) are expected to contribute equally to the project and all partners are expected to know how all code works, even if you were not the primary author of a particular section.**

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.

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,triples,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. You are **required** to use this object for all code that generates graphs when appropriate. This code appears in the starter notebook that we've provided.

```
# "BD" stands for "batting data"
class BD:
      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 (Friday, November 14th)

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. You should write your plan as pseudocode, with the exact format up to you (e.g., flowchart, text). You should be as detailed as possible. This plan must cover **all** tasks in this project through **task 4b**. It must be **at least** 1 page/ 50 lines long.

Task 2: Read the data and create the user interface (Friday, November 21st)

Note: you should keep your data as a single file in the same folder as your project.

2a. `main()` function

- Always begin your code by writing out the `main()` function.
- From your `main()` call a function named `read_data` that returns a list of lists, one list per row in your original file, excluding the headers. Parameters for this function are up to you.

2b. `read_data`: How should we read the data in the csv file?

- You will first have to provide the path to where your "data" folder is located on your computer.
- Read the batting data from the target file.
- *What mode will you open this file in at this step?*
- You need to read in every row from the csv file, convert its values to the appropriate type, and append it to a list of lists (each sublist should correspond to one row of the data).
    - Values that are numbers should be converted to ints
    - Empty values should be converted to the integer 0
    - Values that are strings (player ids, player names, team names, positions) should be left as strings
    - Warning! Make sure that you are not changing the order of the values in your rows!
- Make sure to ignore the headers when you read the data
- You need to return this list of lists from this function

2c. Make sure all columns of your array have values and ensure that your data is the correct types

- After you have read in your data, check its dimensions in your `main` function.
- All columns should have a value for each row.
- All columns that are numbers should be type int.
- All columns that are names, player ids, team names, etc, should be strings
- Copy + paste the appropriate provided function into your notebook to (help) verify that the structure of your data is correct.

Make sure you understand how to access each column of your array using the variables in the `BD` object (provided). You **will** lose points for future tasks if you access columns using integers rather than the provided batting data object.

2d. Implement your User Interface (UI)

After reading the data the next step will 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 no arguments, presents the user with a list of choices of the graphs they would like to generate and returns the choice entered by the user.

If the user enters an invalid choice, they should be re-prompted with the menu until they enter a valid choice.

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 a graph has been generated, the user should be re-prompted with the menu until they choose to exit the program.

At this point, you don't need to have the functionality behind the options implemented, but your menu should otherwise work. I recommend using print statements to make sure that your program is correctly routing the user's choice. (e.g., if a user selects "3", your code should print "Histogram …", and then show the menu again

Example (user input in **bold underlined**):

```
1: Line graph of games played for one player over time
2: Line graph of games played for one player over time separated by team
3: Histogram of runs scored for all players in one year (no cutoff)
4: Histogram of runs scored for all players in one year (cutoff = 100)
5: Graph of team presence over time
6: Homeruns over time (percentiles)
7: [FILL ME IN]
8: [FILL ME IN]
0: Exit
> 123
1: Line graph of games played for one player over time
2: Line graph of games played for one player over time separated by team
3: Histogram of runs scored for all players in one year (no cutoff)
4: Histogram of runs scored for all players in one year (cutoff = 100)
5: Graph of team presence over time
6: Homeruns over time (percentiles)
7: [FILL ME IN]
8: [FILL ME IN]
0: Exit
> dog
1: Line graph of games played for one player over time
2: Line graph of games played for one player over time separated by team
3: Histogram of runs scored for all players in one year (no cutoff)
4: Histogram of runs scored for all players in one year (cutoff = 100)
5: Graph of team presence over time
6: Homeruns over time (percentiles)
7: [FILL ME IN]
8: [FILL ME IN]
0: Exit
> 3

[Display graph for number 3 here]

1: Line graph of games played for one player over time
2: Line graph of games played for one player over time separated by team
3: Histogram of runs scored for all players in one year (no cutoff)
4: Histogram of runs scored for all players in one year (cutoff = 100)
5: Graph of team presence over time
6: Homeruns over time (percentiles)
7: [FILL ME IN]
```

```
8: [FILL ME IN]
0: Exit
> 0
```

2e: Utility functions to deal with your lists of lists.

Lastly for this task, you will implement 4 utility functions that will help you greatly in the rest of your project.

- **`get_matching_rows`**: takes your list of lists, an integer column index, and a target value. Returns a new list of lists that contains all rows from your data where the given column's value matches the target value.
    - Example: get_matching_rows(data, BD.team_name, “Colorado Rockies”) should return a list of all the rows (sublists) for which the team name is “Colorado Rockies”.

- **`get_column_values`:** takes your list of lists and an integer column index. Returns a new list that contains the values of the target column from all of the rows.
    - Example: get_column_values(data, BD.home_runs) should return a list containing all the “home_runs” values.

- **`get_unique_values`:** takes a list of values. Returns a new list that contains all unique values in the original list.
    - Example: get_unique_values([1,1,1,2,2,3,3,10, 5, 10, 5, 10, 5]) should return a list containing [1,2,3,10,5] (the order is not important)

- **`get_unique_column_values`:** takes your list of lists and an integer column index. Returns a new list that contains the unique values of the target column from all of the rows. (you should use get_column_values and get_unique_values to accomplish this).
    - Example: get_unique_column_values(data, BD.year) should return a list containing each year (but each year should only be included once)

Task 3: Making your first graphs! (Monday, November 29th)

For this task you will implement the functionality behind the first 4 options in your menu, producing the first 4 required graphs.

For **all** graphs in this project

- All axes should be labeled
- Graphs should have appropriate titles
- Graphs should be saved to the computer as .pdf files with reasonable filenames

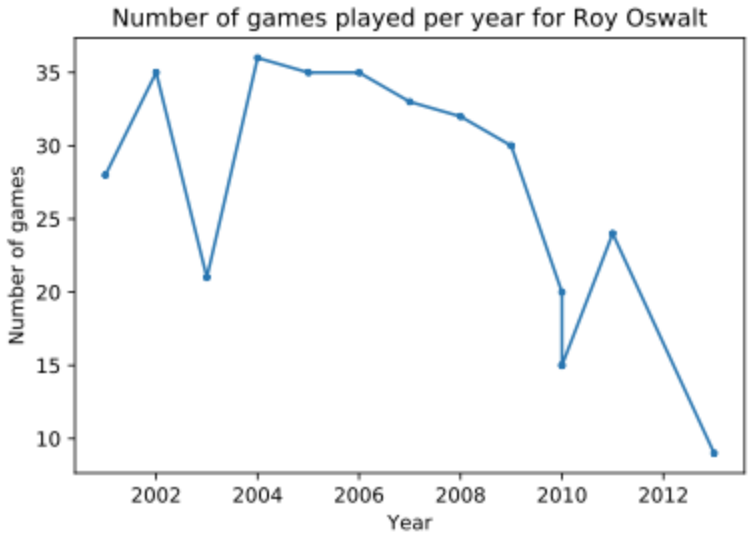
3a: Line graph of games played for one player over time

For your first graph, you will be creating a line graph of the number of games played per year for a particular player.

To begin with, go ahead and "hardcode" in a certain player's id (i.e., pick a player, such as “oswalro01”, and create the graph for them). You should use your get_matching_rows() function from Task 2. Once you have a graph that works, let the user pick between either:

a) entering the player id themselves OR
b) generating a graph for a randomly selected player

You may assume that the user will enter a correct player id.



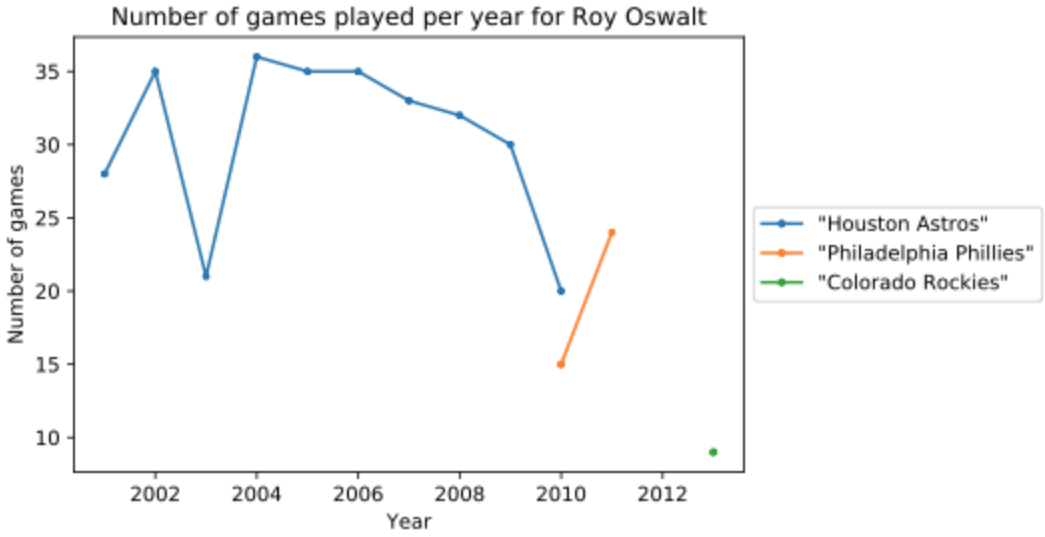
3b: Line graph of games played for one player over time, separated based on team

Next, you'll build a more complex graph based on the one that you previously produced. Graph the number of games played per year for a specific player, but plot a separate line for each team that they played for.

Follow the same strategy as in 3a:
To begin with, go ahead and "hardcode" in a certain player's id. Once you have a graph that works, let the user pick between either:

c) entering the player id themselves OR
d) generating a graph for a randomly selected player

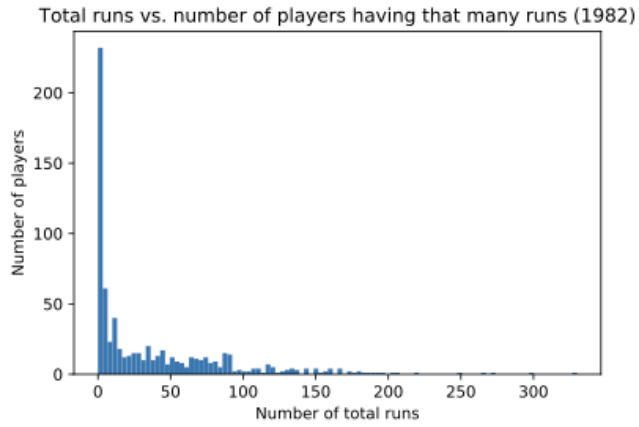
You may assume that the user will enter a correct player id.



3c: Histogram of total runs scored for all players for a specific year

Here, you will be creating a histogram with 100 bins for the number of runs scored in a specific year for all players. For the given year, for each player in the dataset, you will need to sum the runs from all rows belonging to that player to calculate their total runs.

This is because some players play multiple positions or for multiple teams in one given year, as we saw from the previous graphs!



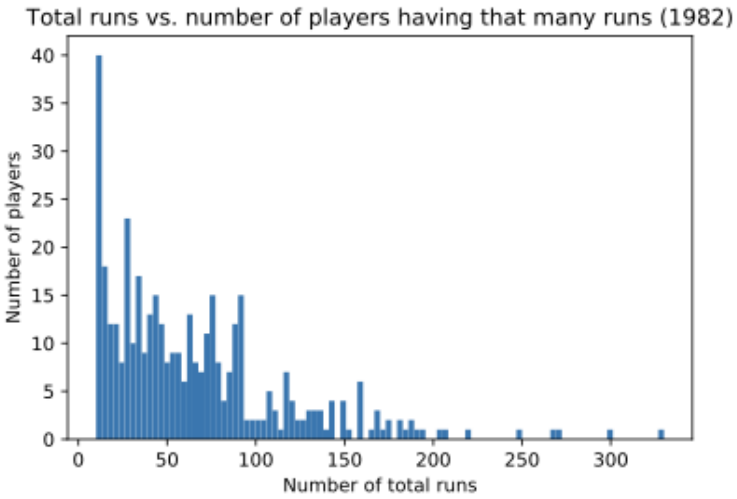
Then, you will take a list of all of the total run values, and give these values to matplotlib's hist() function.

To accomplish this, you will find the get_matching_rows and get_unique_column_values functions useful.

3d: Histogram of runs scored for all players in a specific year (with a "minimum" cutoff)

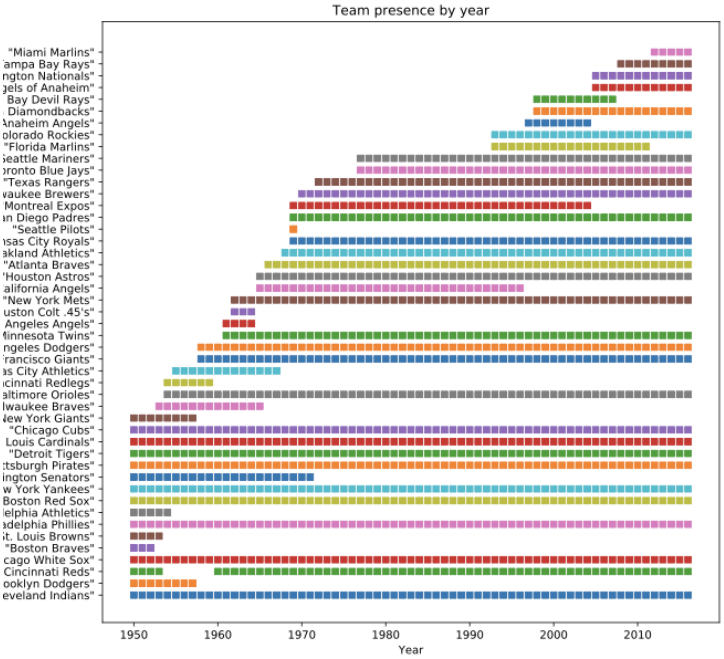
Your next task is the same as the previous task, but this time we will impose a cutoff, meaning that we will only be graphing the data from players whose total runs value is greater than the cutoff. We've shown you a graph with a cutoff set to 10, but you should design your function so that you can pass in *any* number as the cutoff.

For this task, you should modify the function that you wrote for the previous graph so that you can use it for either no cutoff (equivalent to cutoff = 0), or with a cutoff! Don't write a new function!



Task 4: Graphing the rest of your graphs! (Monday, December 6th)

Task 4a: Creating a time plot of when teams were active

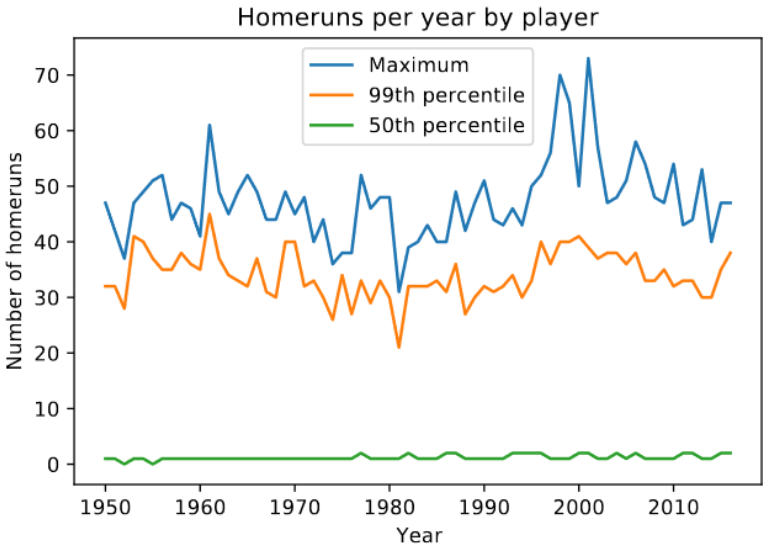


For this task, you will be creating a scatter plot showing when each team was active. The y-axis will be labeled with team names and the x-axis will be the years that they were active. A team is considered to be active during a given year if there is any player from that team with statistics for that 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.

4b: Plot homeruns over time (percentiles)

For this task you will create a bar graph with the following (graphed in this order) : the maximum number of home runs scored every year, the 99th percentile of # of home runs scored every year, and the 50th percentile of home runs scored every year. Take a moment to think about what this data tells us.



4c: Explore the data and choose two more statistics to create graphs of.
Add these to your menu as options corresponding to choices 7 and 8.

- +2.5 points (up to +5) for each new type of plot you make (as in, other than line, bar, or scatter plot)
    - E.g., look up boxplot, violin, imshow, pie or others (see gallery for inspiration: https://matplotlib.org/gallery/index.html)
- Both graphs must plot statistics that you have not already plotted in this project.
- Both graphs must be more different than graphs you have already produced than simply changing the column that you are graphing and the type of graph.
- Questions about new graphs are welcome on piazza and in David's office hours!

Here are some places that you might start:

- Look at the columns of data that you have not yet used! (rbi, stolen_bases, doubles, triples, etc)
- Look up other standard stats used to help analyze baseball statistics
    - make sure to comment on any sources that you consulted in the code that you turn in!

Extra Credit (up to 20 points, due on Monday, December 6th, alongside Task 4):

Some of the extra credit questions should be answered in a separate document (`extra_credit.pdf`), some in your code. Each question indicates where you should answer it.

1) (`extra_credit.pdf`) **Does it make sense to have the first two graphs be line graphs? why/why not? If not, what would a more appropriate type of graph be?**

2) **Create a new graph that visualizes a statistic and separates players based on position.**
    a) (`code`) Implement this as option "9" in your menu.
    b) (`extra_credit.pdf`) Put a copy of the graph you produced in this write-up. What conclusions can you make from your graph?
    c) (`extra_credit.pdf`) Explore the data to discover the source of what you described in part (b). Why is the graph the way that it is?

d) (`code`) Ensure that your code would work correctly if I gave you a new data set that had positions encoded differently or had a different number of positions altogether.

3) (`code`) **A player's "on base percentage" is a metric often used to measure how good a player's season is.**

Implement this as option "10" in your menu.

On base percentage is calculated according to: (hits + walks + hit by pitch) / (at bats + walks + hit by pitch + sacrifice flies)

Consider one row in your data set to be equal to one season. (for the purposes of this question, we will say that players who switched position or team mid-season had "multiple" seasons). Find the 20 best seasons according to on base percentage.

Next, find the 20 best seasons if you limit the data so that you only consider seasons that are at at least the 25th percentile for at bats and at least the 25th percentile for games played. How does this change your answer?

Write at least 3 sentences in the comments explaining the results from these two calculations and how they changed. Which list of seasons do you think should be used when considering best seasons?

4) (`extra_credit.pdf`) **Analyze the graphs that you created for task 4. What conclusions can you make from these graphs? What sort of additional evidence might you want to support these conclusions? How could you improve these graphs to make them more easily readable?**

Acknowledgements:

This project was originally developed by Felix Muzny, and has been since adapted by Dr. David Wooten.