Web Scraping Running2win.com

Introduction

Sean Kane, Jake Carlson

The main goal of the application was to be able to scrape data from the popular run-logging website, Running2Win.

There are three main python files, athlete.py, getusers.py, and r2w.py. “Athlete.py” contains a class structure for each username. The dunder method takes a username and will fetch the age, gender, age of the account, total miles logged, and PRs (in the 800, mile, 5000, half marathon, and full marathon). In order to get each of those the

The first step to the web scraping was to have a base of usernames. There is a page on running2win.com which shows the 500 most recent runs logged. The getusers.py logs into the website and uses the get() function to fetch the web page. The html page is then parsed using an html.parser function built within BeautifulSoup. The table that has the usernames are contained had two tags ‘align’ and ‘valign’ in the html document, which could be used to filter out only the usernames. The usernames were stored in a list and then compared to the usernames already in r2wUsers.csv, if the username is not in the csv file already, it is appended to the end of the list.

The r2w.py file contains all the methods for scraping info from the website. Running2win is not an open website and you have to be a user to look at the information of other users. The main method starts off by logging into the website with an account username and password made specifically for this process. From there, BeautifulSoup and requests can be used to navigate through the website and gather data from different links to the page. There are three pages that are used to gain all the information, view-member-running-log, view-member-profile, and the UserRaces pages. The first page is used to pull out the total mileage logged. The second page can be used to gather, age, gender, and account lifetime. The last page can gain the personal best times for 800, mile, five-kilometer, half marathon, and marathon. If the user has not logged a time for any of these, a zero is recorded in its place.

Athlete.py holds the file for the class structure for each username. This file is used primarily as a data structure for storing the values scraped from Running2Win using r2w.py. It also had many helper methods which served the purpose of formatting strings and converting values to strings or different units.

After all the scraping was done and the data was recorded for each user, matplotlib and statistics were used to analyze the data. The data, which was originally written into a CSV file, was read from that same file. Parsing the file by each individual line, allowed the each user’s

data to be read individually and placed into separate arrays denoting username, gender, total Miles, personal bests in multiple events, age, and account lifetime. Using the statistics library, mean and standard deviation was found for a runners personal bests in the 800 meter, Mile, 5 Kilometer, Half Marathon and Full Marathon.

Project Results

The scraping of the data was very successful and we were able to gather crucial data on close to 1000 users from Running2win, however we did not account for a majority of these users to log their personal bests. Roughly half of these users had a 5k best logged, and a third had the other four distances. For the plots, we removed a zero value (indicating the user did not have it recorded) and plotted only the self-recorded values. While this caused some problems when analyzing the data, it was not a hindrance when scraping.

By purely looking at the mean and standard deviation of each distance, it yields that most runners that are constantly logging runs are faster than the average runner, with the average PR for each event being rather quick. Figure 2 and 4 make the comparisons between age vs mile personal best and total miles vs. mile personal best respectively, but there is too much noise in all graphs to make any clear assumptions or generate any mathematical models. While the data was not as clear as we would have liked or expected, the main goal of our product was still achieved, scraping the data of users and representing this data graphically.

Scraped Data

66.16 % Male 33.84 % Female

Average age: 26.02	Age Standard Deviation: 11.04	
Average 800: 00:02:21.73	800 Standard Deviation: 00:01:43.33	Count: 261
Average Mile: 00:04:52.98	Mile Standard Deviation: 00:01:03.49	Count: 431
Average 5k: 00:17:25.58	5k Standard Deviation: 00:02:59.84	Count: 525
Average Half: 01:25:30.57	Half Standard Deviation: 00:16:19.04	Count: 244
Average Marathon: 03:04:02.90	Marathon Standard Deviation: 00:36:29.41	Count: 145

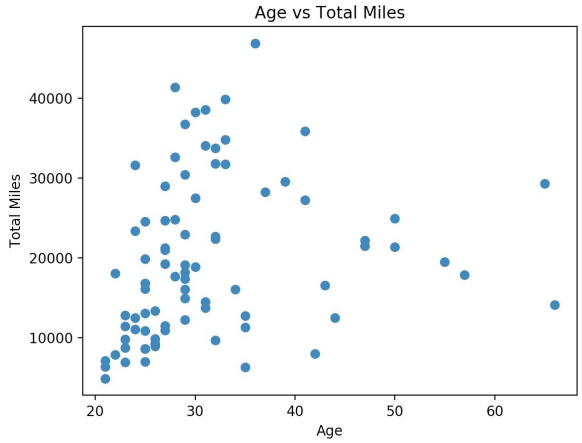


Figure 1. Age vs. Total miles of sample.

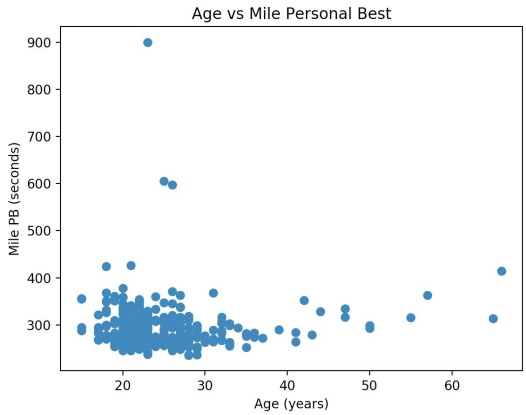


Figure 2. Age vs. Mile Personal Best.

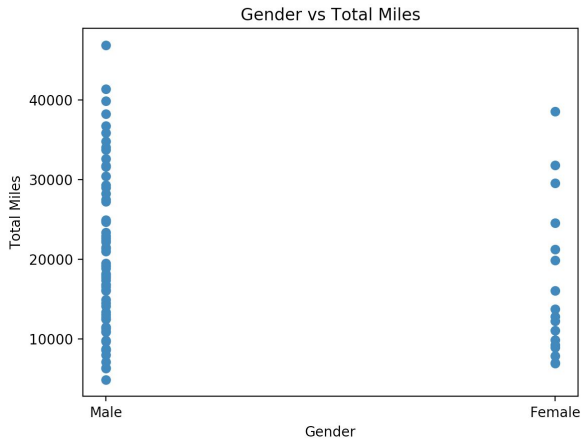


Figure 3. Gender vs. Total miles of sample.

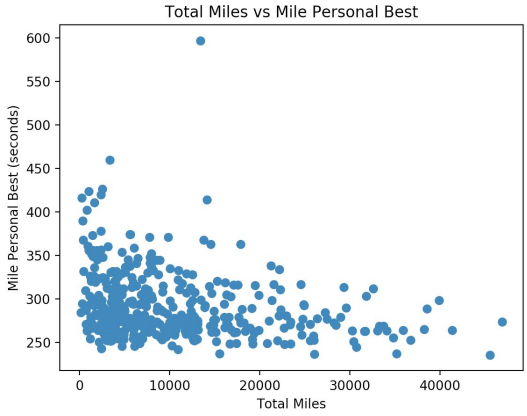


Figure 4. Total Miles vs Mile Personal Best

Division of Work

Jake Carlson	Sean Kane	Together
● Wrote BeautifulSoup functions for gender, and all the personal bests ● Wrote matplotlib functions that created all the plots ● Created athlete.py file and athlete class to streamline creating each athletes class	● Wrote BeautifulSoup functions for total miles, age, and account Lifetime ● Used statistics library to generate and print mean and standard deviation for each event ● Created getusers.py to scrape new usernames	● The data scraping took longer than initially expected (5-15 seconds per user), and both of us used our computers to get the data for half of the list.

Bibliography

"Beautiful Soup Documentation¶." *Beautiful Soup Documentation - Beautiful Soup 4.4.0 Documentation*, Beautiful Soup, beautiful-soup-4.readthedocs.io/en/latest/.

Crummy. (n.d.). Beautiful Soup Documentation. Retrieved April 20, 2018, from https://www.crummy.com/software/BeautifulSoup/bs4/doc/

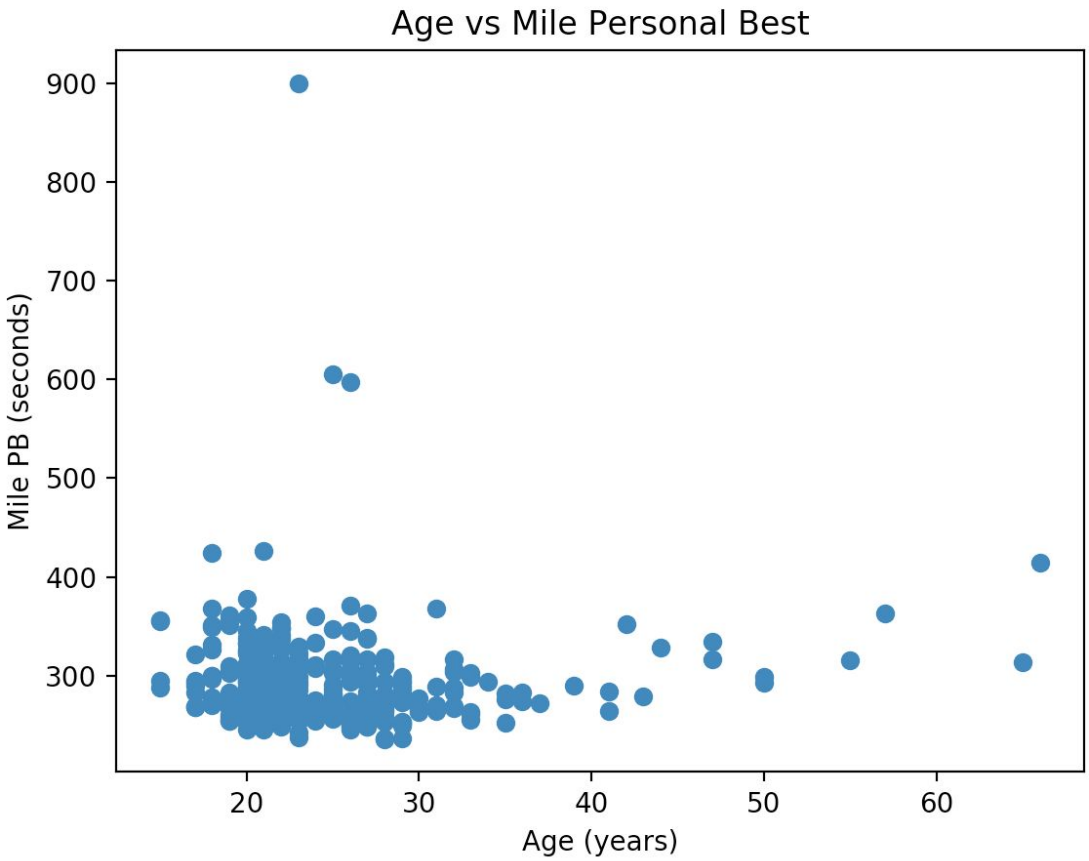
Hunter, J., Dale, D., & Droettboom, M. (n.d.). Matplotlib Documentation. Retrieved April 20, 2018, from https://matplotlib.org/contents.html

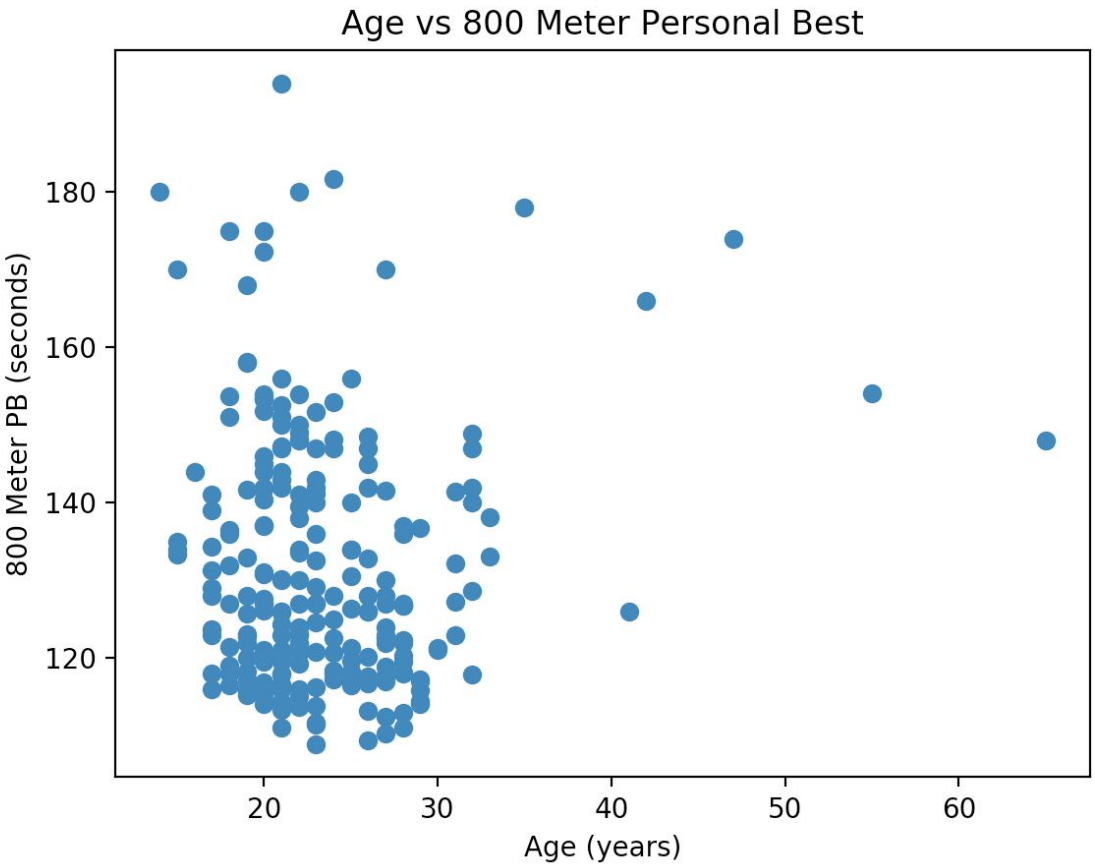
Python. (n.d.). Requests: HTTP for Humans. Retrieved April 20, 2018, from http://docs.python-requests.org/en/master/

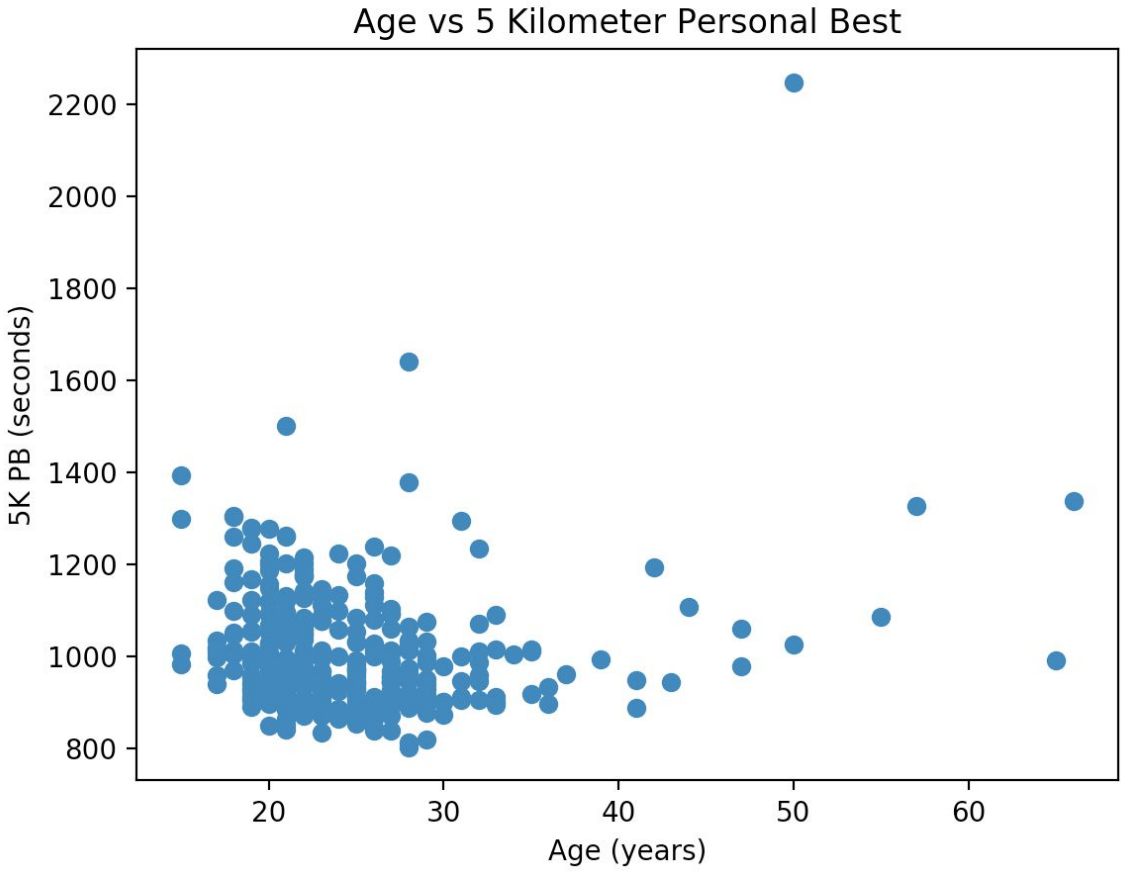
Code Appendix

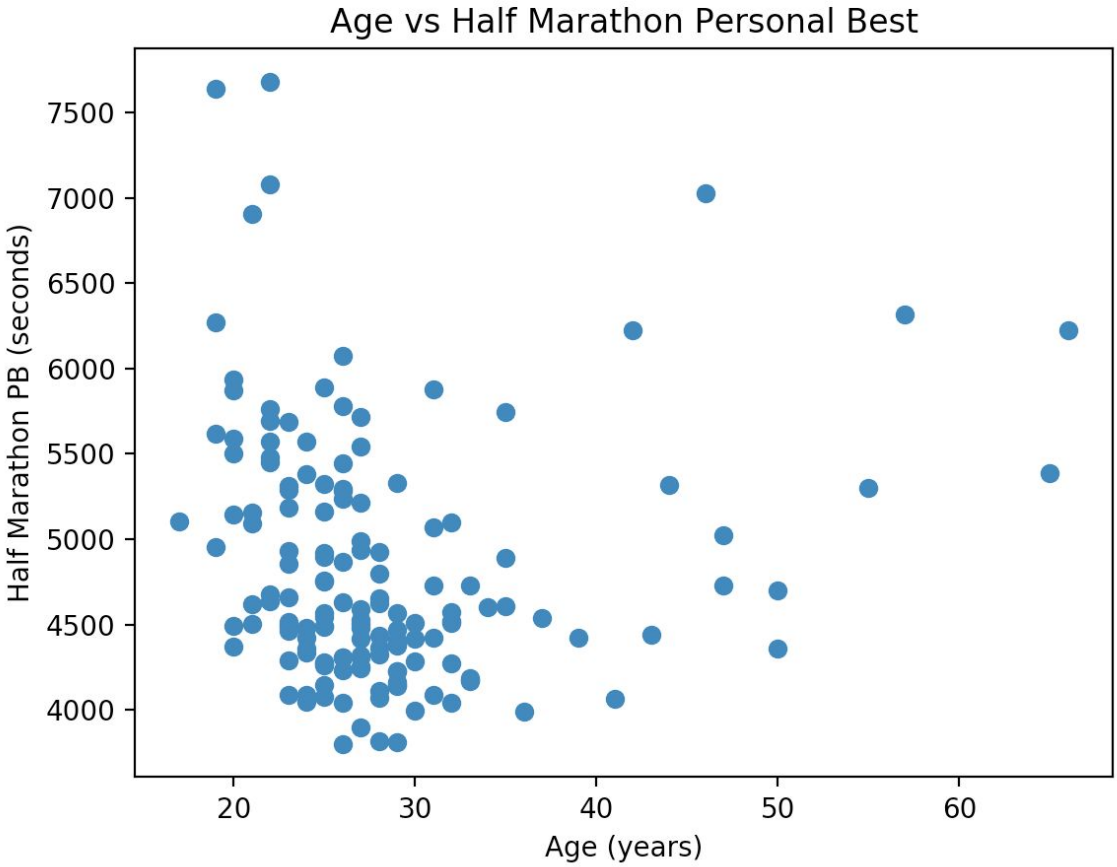
The project was hosted on Github.

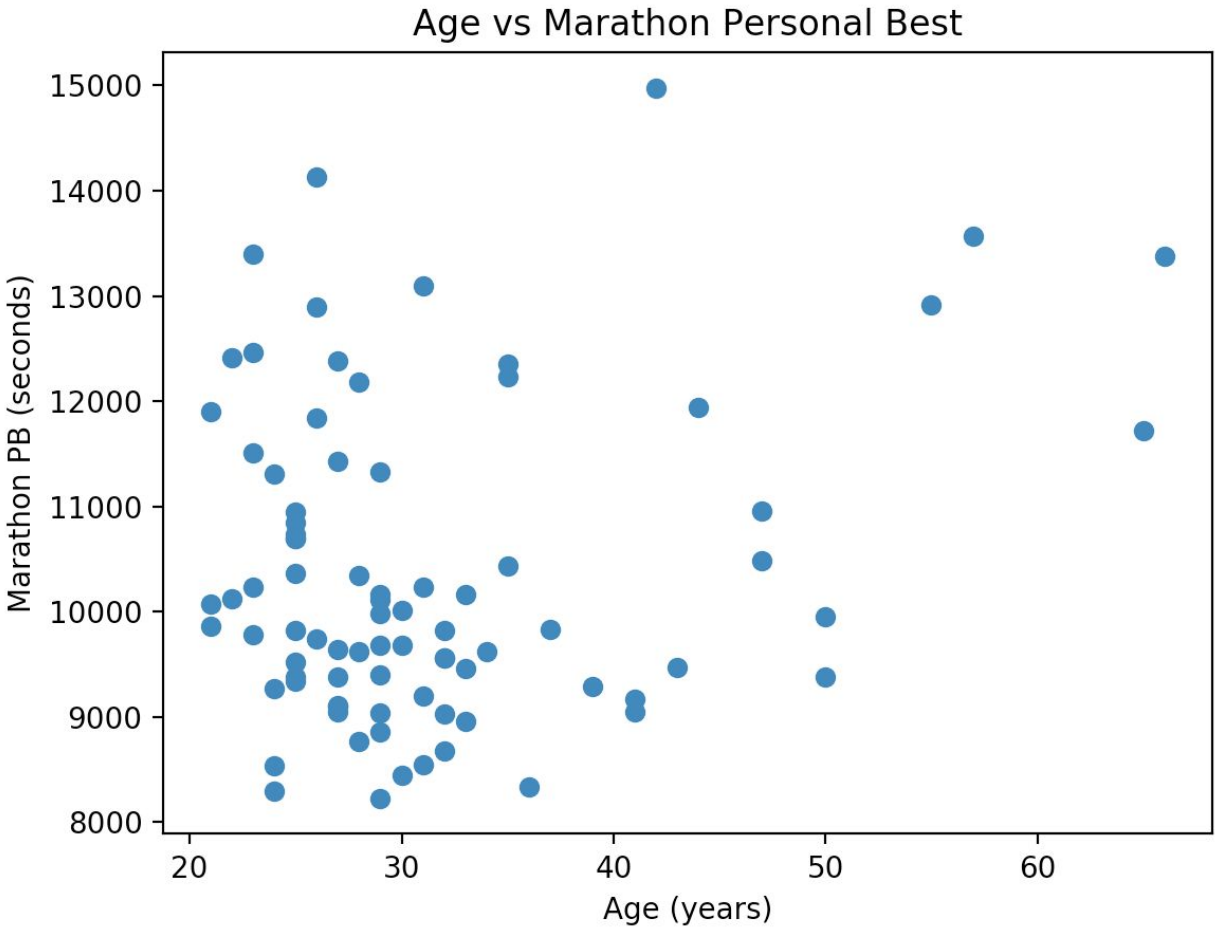
https://github.com/spkane31/webscrape-running

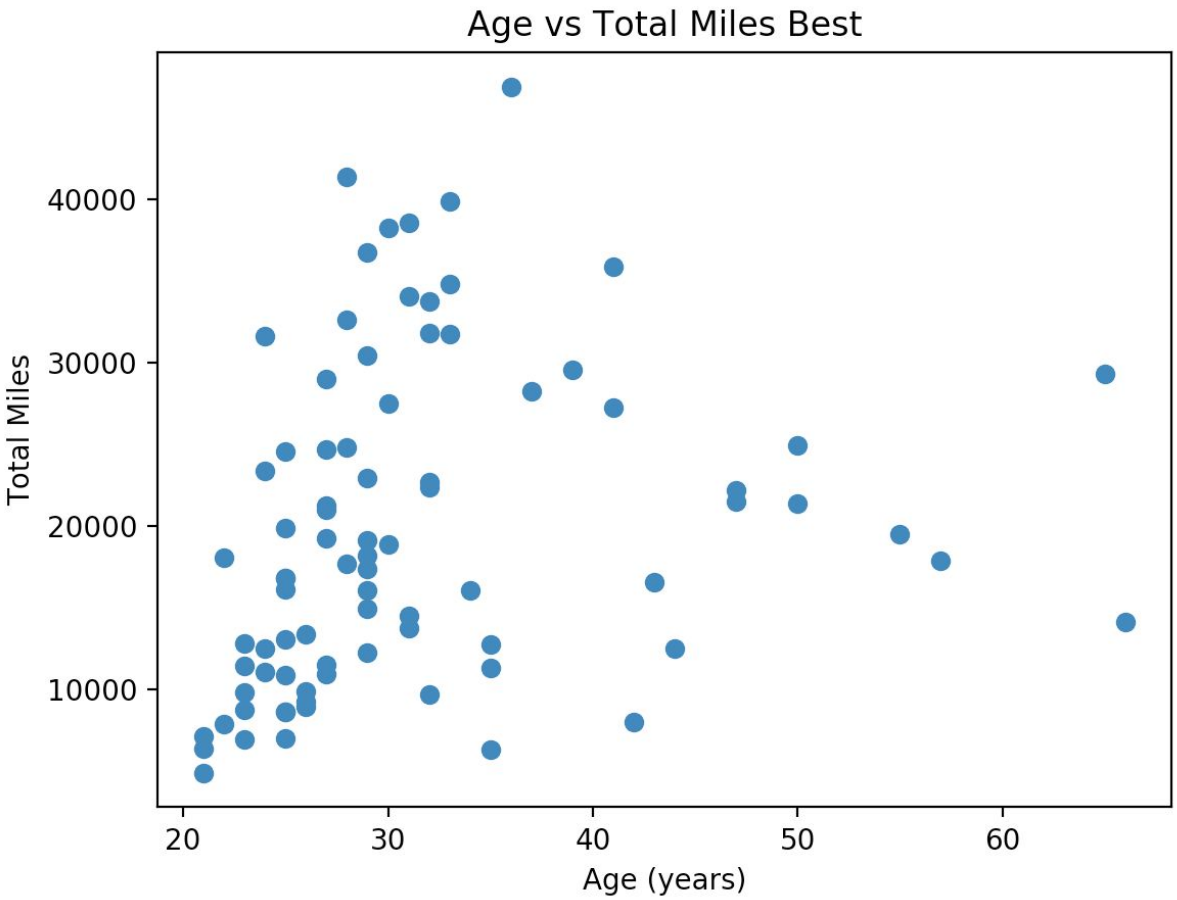

Age vs Mile Personal Best
Mile PB (seconds)
900
800
700
600
500
400
300
20
30
40
50
60
Age (years)

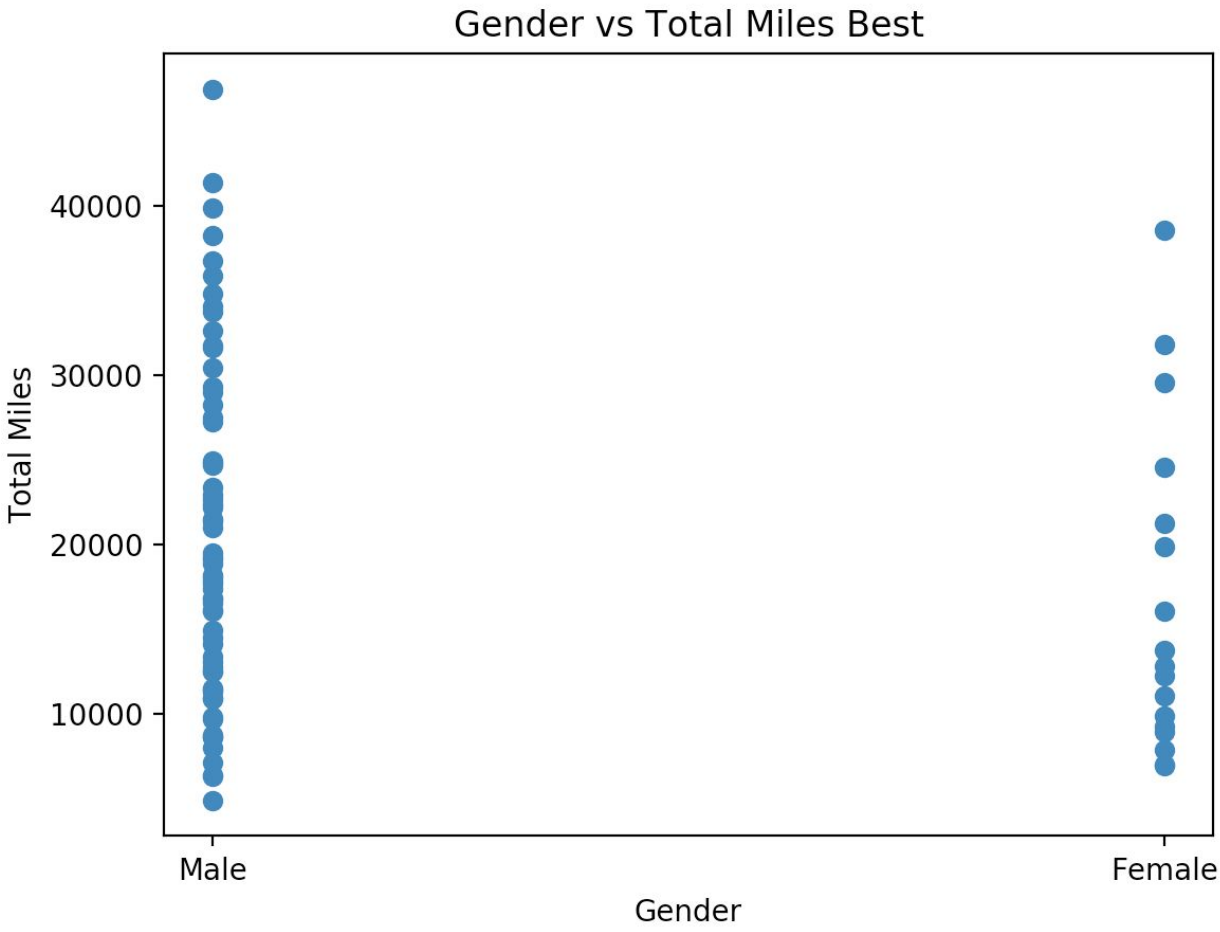

Age vs 800 Meter Personal Best
800 Meter PB (seconds)
180
160
140
120
20
30
40
50
60
Age (years)

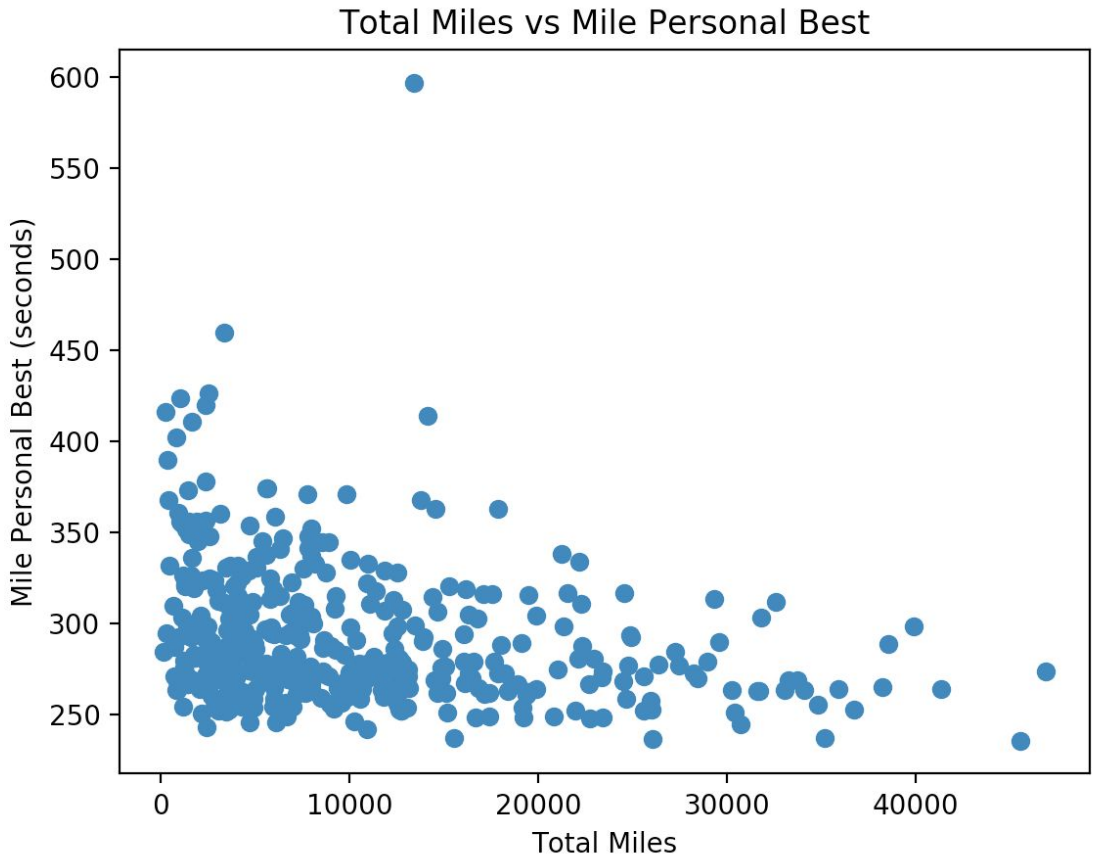

Age vs 5 Kilometer Personal Best
5K PB (seconds)
2200
2000
1800
1600
1400
1200
1000
800
20
30
40
50
60
Age (years)

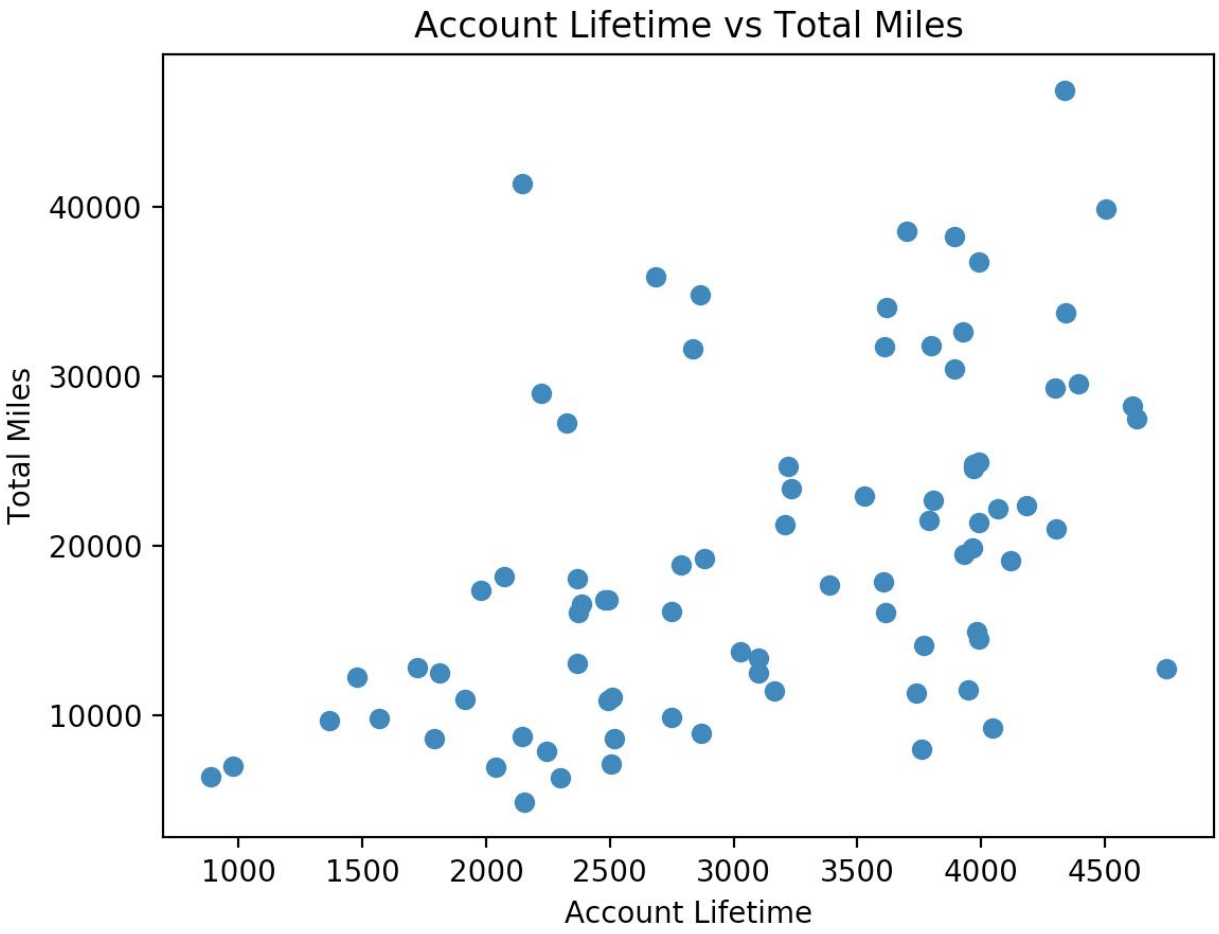

Age vs Half Marathon Personal Best
Half Marathon PB (seconds)
7500
7000
6500
6000
5500
5000
4500
4000
20
30
40
50
60
Age (years)

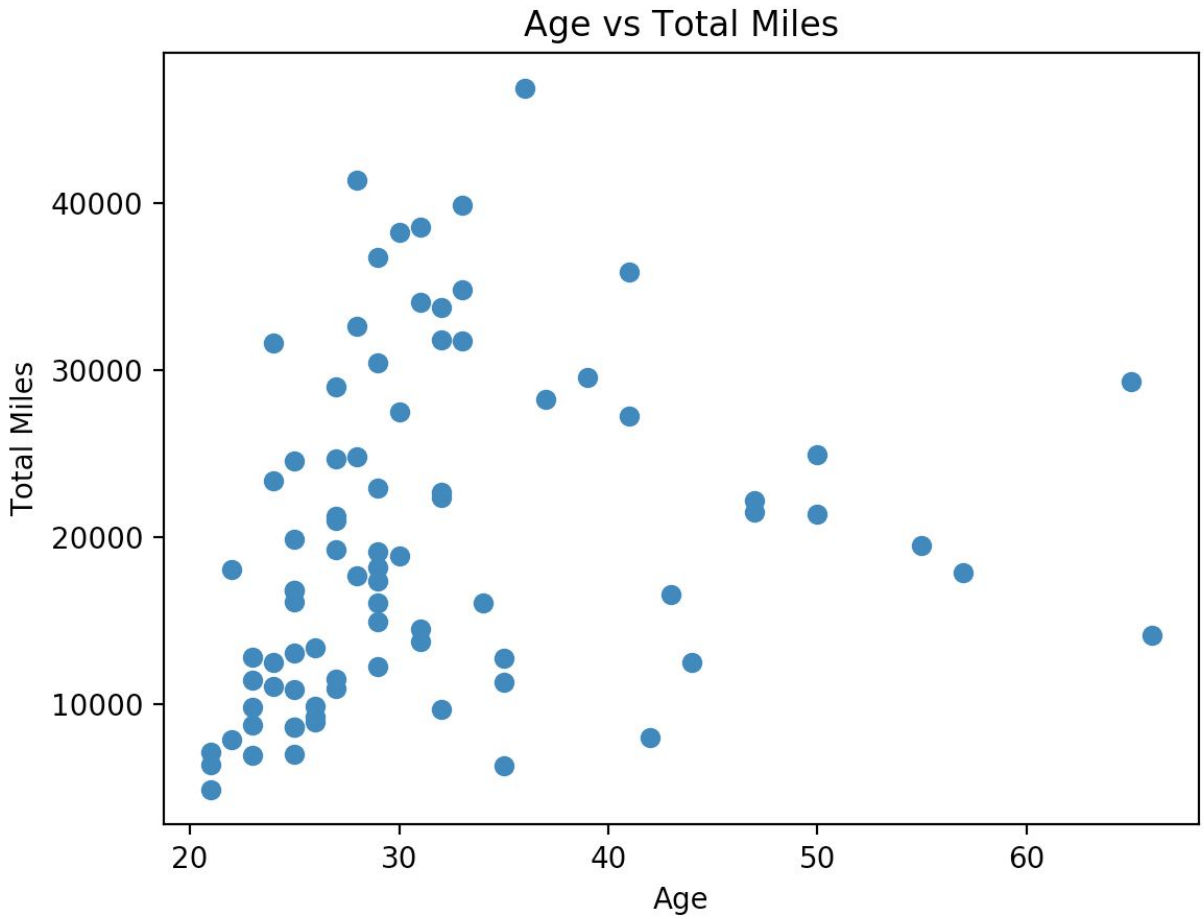

Age vs Marathon Personal Best
Marathon PB (seconds)
15000
14000
13000
12000
11000
10000
9000
8000
20
30
40
50
60
Age (years)

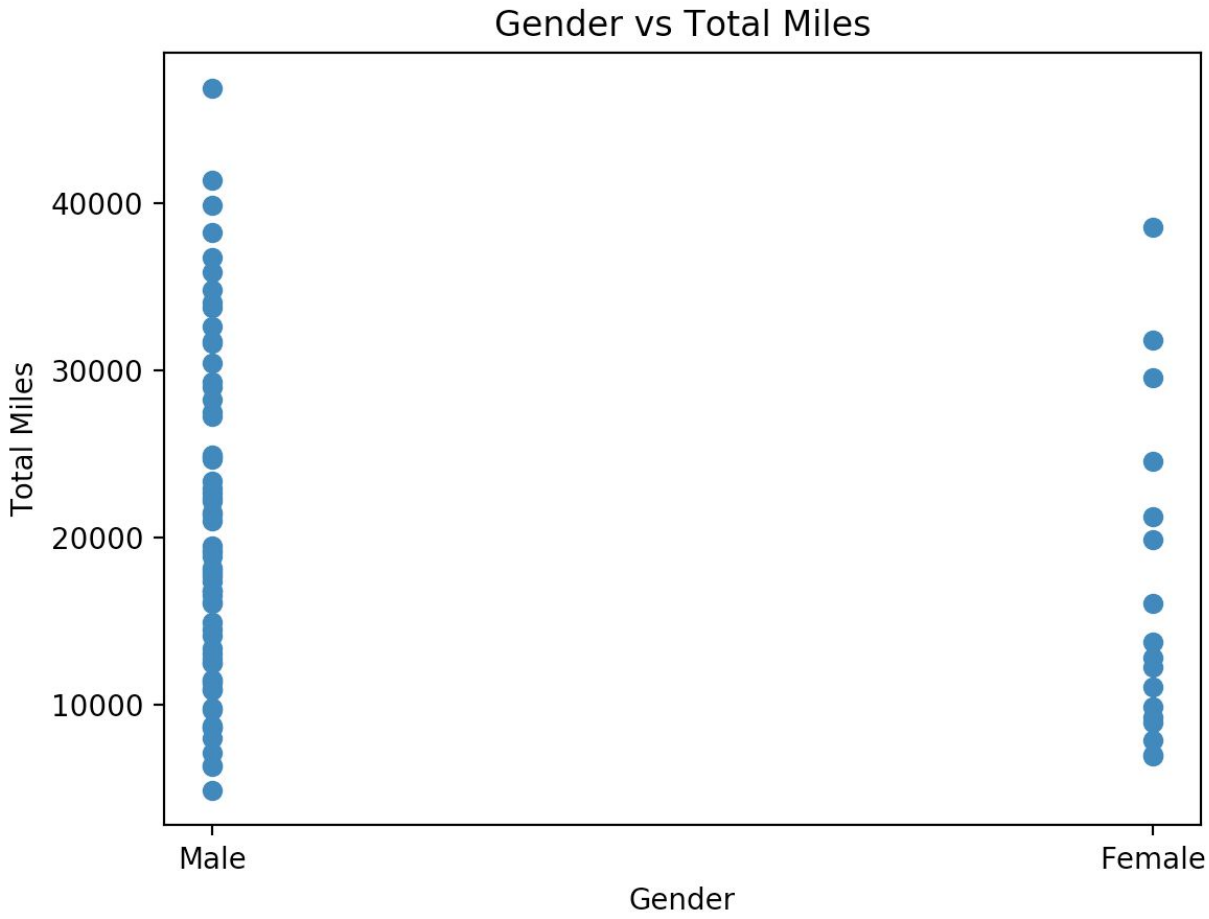

Age vs Total Miles Best
Total Miles
40000
30000
20000
10000
20
30
40
50
60
Age (years)


Gender vs Total Miles Best
40000
30000
20000
10000
Total Miles
Male
Female
Gender


Total Miles vs Mile Personal Best
Mile Personal Best (seconds)
600
550
500
450
400
350
300
250
0
10000
20000
30000
40000
Total Miles


Account Lifetime vs Total Miles
Total Miles
40000
30000
20000
10000
1000
1500
2000
2500
3000
3500
4000
4500
Account Lifetime


Age vs Total Miles
Total Miles
40000
30000
20000
10000
20
30
40
50
60
Age


Gender vs Total Miles
Total Miles
40000
30000
20000
10000
Male
Female
Gender