**10/23/2017**

**Visualizing and Analyzing the FIFA World Cup**

**Project 2 Report**

***Braeden Brettin***

**Table of Contents**

[Table of Figures 3](#_Toc496100872)

[Background 4](#_Toc496100873)

[Data Visualization and Analysis 5](#_Toc496100874)

[Best Practices in Interactive Computing 12](#_Toc496100875)

[Advanced IPython Notebook Techniques 15](#_Toc496100876)

[References 16](#_Toc496100877)

# Table of Figures

[Figure 1. Excel Workbook 5](#_Toc496532548)

[Figure 2. Read in Data 6](#_Toc496532549)

[Figure 3. Other Pandas Functions 7](#_Toc496532550)

[Figure 4. Data Frame Plot 7](#_Toc496532551)

[Figure 5. Bar Chart 8](#_Toc496532552)

[Figure 6. Sorted Bar Chart 9](#_Toc496532553)

[Figure 7. %timeit 9](#_Toc496532554)

[Figure 8. New Excel Workbook 10](#_Toc496532555)

[Figure 9. Widget 10](#_Toc496532556)

[Figure 10. Poor Directory Structure 12](#_Toc496532557)

[Figure 11. Improved Directory Structure 13](#_Toc496532558)

[Figure 12. Markdown Notes 13](#_Toc496532559)

[Figure 13. Markdown Notes 14](#_Toc496532560)

[Figure 14. Markdown Notes 14](#_Toc496532561)

# Background

Imagine you had no programming experience and were asked to look at data produced by your client, a recent Software Engineering graduate from Mercer University. The graduate first shows you a line of Python code that contains every individual data point. All you can recognize is a sequence of numbers with a few keywords that stand out to you. The graduate then shows you a chart encompassing all this data. Which would you rather view? With absolutely certainty, I can say you would prefer the second option. Given your response, we can see the need for programmers to have a knowledge of data visualization techniques.

Having established a strong foundation in IPython notebooks during the last project, I now proceed to data visualization and analysis techniques that can be used throughout my career and any independent projects I elect to work on. Given the frequency with which programmers and software developers work with clients who are not versed in any programming languages, it is imperative that we learn how to create data that can be easily visualized and read by any person. For this project, this visualization will be accomplished using numerous Python libraries which produce charts, graphs, and other figures that can be examined within the IPython notebook. Throughout this project, I will address major techniques of IPython data visualization and analysis, common good practices for writing IPython code, and advanced techniques that the IPython notebook can perform.

# Data Visualization and Analysis

I begin by constructing a project with which I can implement the data visualization techniques I have previously learned. Given my passion for the game of soccer and the upcoming World Cup, I set out to find if I could objectively determine the best-performing country in the history of the competition. I found data on various statistics regarding countries that have participated before in the World Cup (*All-time table of the FIFA World Cup*). I then placed this data into an Excel workbook that will be read in using commands in the IPython notebook. As shown in Figure 1, below, this workbook contains data on the country, number of times it has participated in the world cup, total number of matches it has played, total wins, total draws, total losses, goals for, goals against, goal differential, total points, points per match, and best result.



Figure 1. Excel Workbook

The Pandas library in Python provides a plethora of useful functions for reading in and displaying data in tables. Pandas can read in a csv (comma delimited) file and display it in an easily-visualized table (Albon). I saved the excel workbook as a csv file and used the read\_csv() function to read in this file. As shown in Figure 2, below, simply calling the name of the newly created Data Frame object will display the data in a table that can be visualized by any non-programmer. Pandas also provides the ability to save raw data as a data frame object, a useful function, but not applicable in the case of this project.

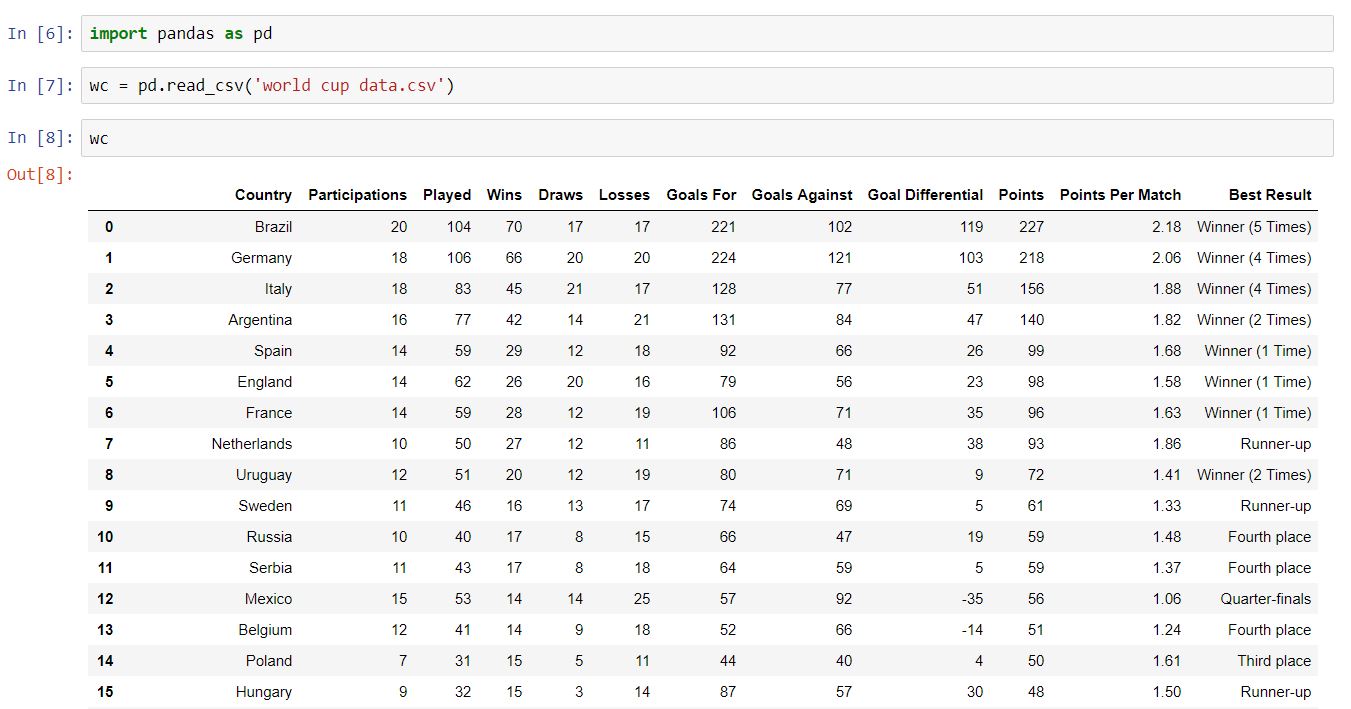


Figure 2. Read in Data

The Pandas library also provides many other useful functions, such as the head() function, which displays the first few rows of the data, and the tail() function, which displays the last few rows of the data. Possibly the most useful function provided by Pandas is the describe() function, which provides a statistical analysis of the data. Figure 3, below, utilizes all three of these functions.

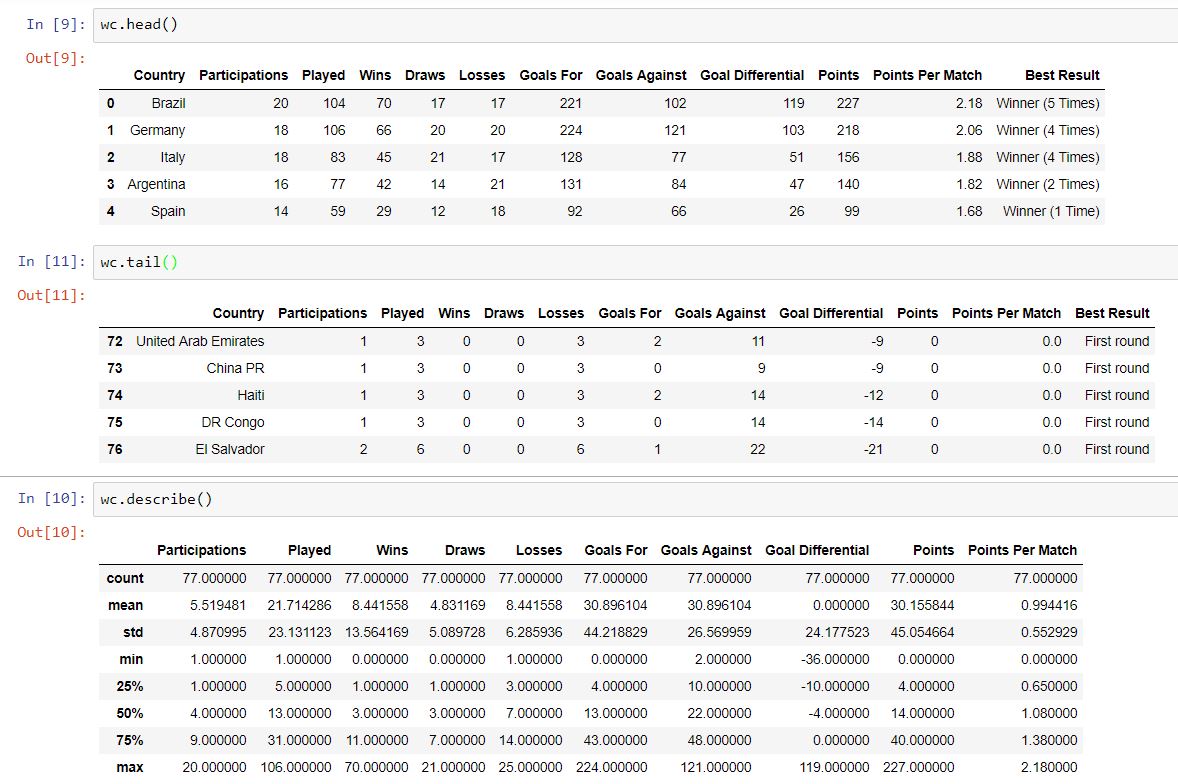


Figure 3. Other Pandas Functions

Of course, most people would prefer seeing data in a graph rather than a table. For this purpose, one can use the Matplotlib library. If we were to plot the data using the plot() function, as shown in Figure 4, below, we should see a negative correlation for each of the sections, since the data has been sorted by total points already; teams with more points in the competitions usually have more appearances and regularly perform better in the competition.

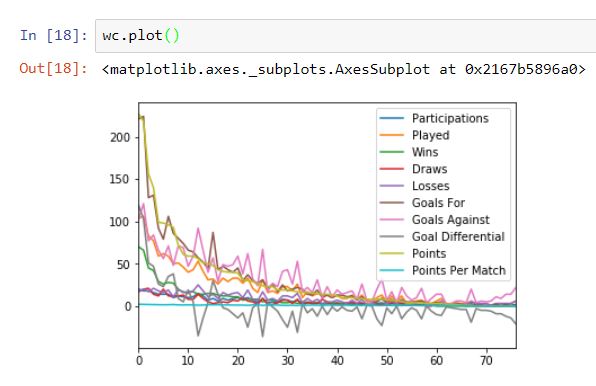


Figure 4. Data Frame Plot

Unfortunately, this graph does not allow one to easily visualize any individual country’s performance in the competition. Fear not, for the Matplotlib library provides functions to plot specific countries and sections of data. For instance, I want to see a bar chart showing the total number of points for the top five countries in the competition (*Matplotlib Bar chart*). To achieve this, I will incorporate the NumPy library. This library provides useful functions for data analysis using multi-dimensional arrays, which is what the data frame object is stored as. After saving the top five countries and their point totals to two separate arrays, I then used the bar() function to display a bar chart of these arrays, as shown in Figure 5, below.

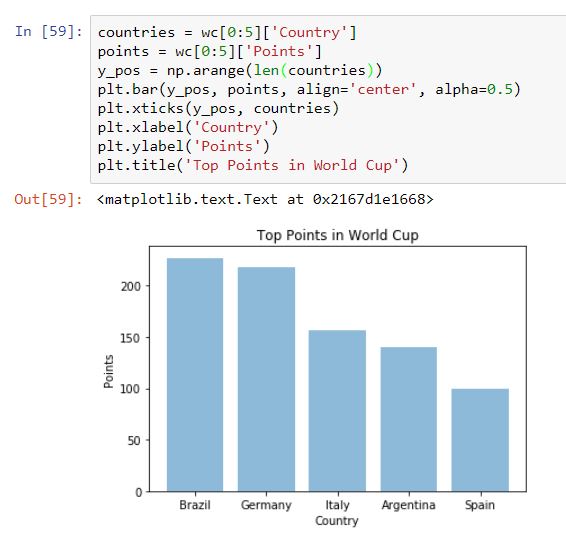


Figure 5. Bar Chart

The code to display the top point earners is easy enough, given that the countries have already been organized by total number of points earned. What if I wanted to display the top countries in terms of points per match, however? Because not all countries have played the same number of matches, the table may be different when sorted by points per match. Thankfully, Pandas allows the user to sort a table by any column desired (*pandas.DataFrame.sort\_values*). I used the sort\_values() function to sort the table by ‘Points Per Match’ and displayed the data in another bar chart, as shown in Figure 6, below.

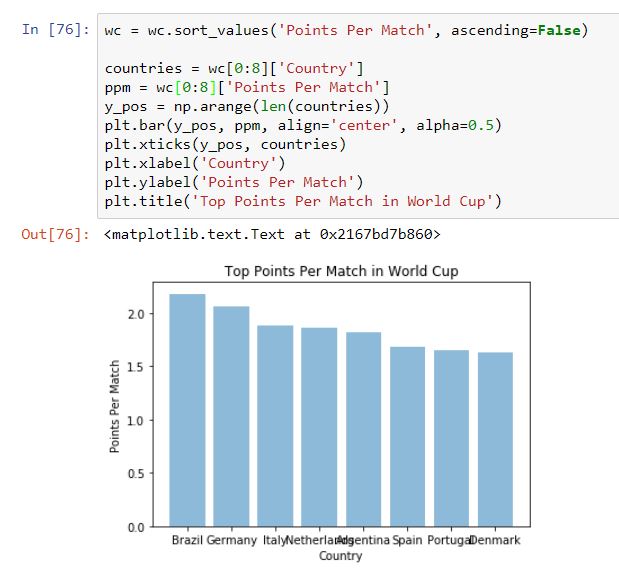


Figure 6. Sorted Bar Chart

Efficiency in algorithms should be of the utmost importance to developers, as it could mean the difference between a program taking several hours or several seconds to run. To aid in developers determining the efficiency of their algorithm, IPython provides a handy magic command, ‘%timeit’, that calculates the amount of time it takes to accomplish a loop of the code block that is run. Gone are the days when developers needed to include system time calls in their code to debug and improve an algorithm. Simply adding the ‘%timeit’ command before a function call will display the time the function ran for, as shown in Figure 7, below. I can use this command to refactor my existing code to hopefully reduce its run time.

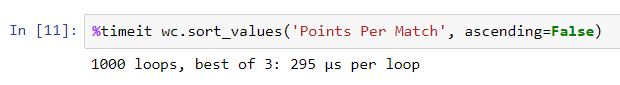


Figure 7. %timeit

Among the major visualization techniques addressed in Chapter 1 is the use of interactive widget. With a drop-down widget, a user can quickly select from any number of input sources or links, rather than having to rewrite the code in the IPython notebook every time a new set of data is analyzed (*Widget List*). For the purposes of this project, I have created an additional excel workbook containing data on the top scorers in the competition’s history (*FIFA World Cup top goalscoreres*), as shown in Figure 8, below.



Figure 8. New Excel Workbook

I then added a drop-down button widget that allows me to select from either the original data or this new data set, as shown in Figure 9, below. For this new data set, I simply need to use the read\_csv() function to read in the new data set. Upon doing so, I can carry out a plethora of functions to better visualize and analyze the data.

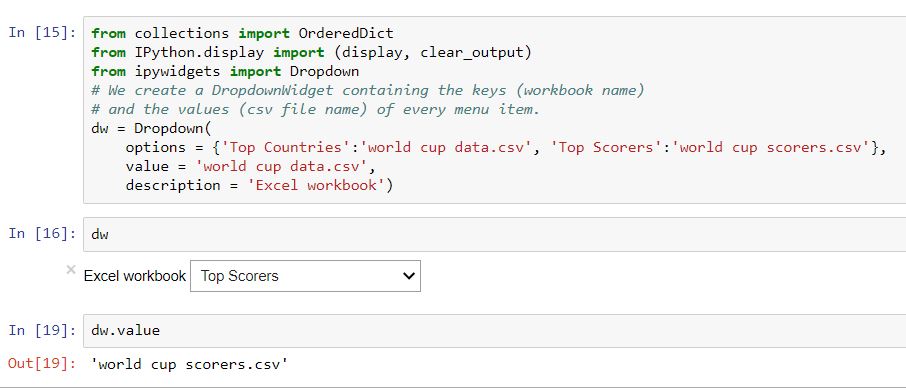


Figure 9. Widget

The most notable characteristic of IPython is its interactivity. Any user with access to the notebook can change a code block and quickly observe how that change affects the data. I have made the notebook accompanying this project interactive and have included it in my FTP folder alongside this report.

# Best Practices in Interactive Computing

When creating an IPython project, it is important to remember that the project should ideally be efficient and reproducible. Efficiency improves the speed at and ease with which the project runs. Reproducibility requires documenting ones work so that other developers can easily build against the current project. To achieve this ideal, Rossant has provided several tips that developers should be mindful of when writing code and creating IPython notebooks (59). I will address a number of these tips and demonstrate them in practical use.

First, Rossant recommends that the developer organize his or her directory structure in such a manner that any person searching for a notebook or script within the directory can quickly and easily locate it. This involves employing good naming conventions and storing notebooks and scripts in their correct location. Before I utilized this tip, all the notebooks I had previously worked on in this class were in one folder, as shown in Figure 10, below. This will present problems as I continue to develop more IPython notebooks and store them in the same location. To fix this poor directory organization, I created specific folders and subfolders for each project I have worked on in the class, as shown in Figure 11, below. Now, any user can easily navigate the notebook in question with little to no guidance.



Figure 10. Poor Directory Structure



Figure 11. Improved Directory Structure

In normal high-level languages, comments give the user and other people looking at the code the ability to quickly understand what the developer intended to accomplish with a segment of code. The IPython notebook, however, provides developers the option to write notes in Markdown and other text languages. These notes can be more extensive than regular comments and can be formatted to be bolded, italicized, and more. These additional formatting options can be utilized to make a specific code segment stand out for other users. As shown in Figures 12 through 14, below, I went through my notebook and added Markdown notes to better explain specific code segments.

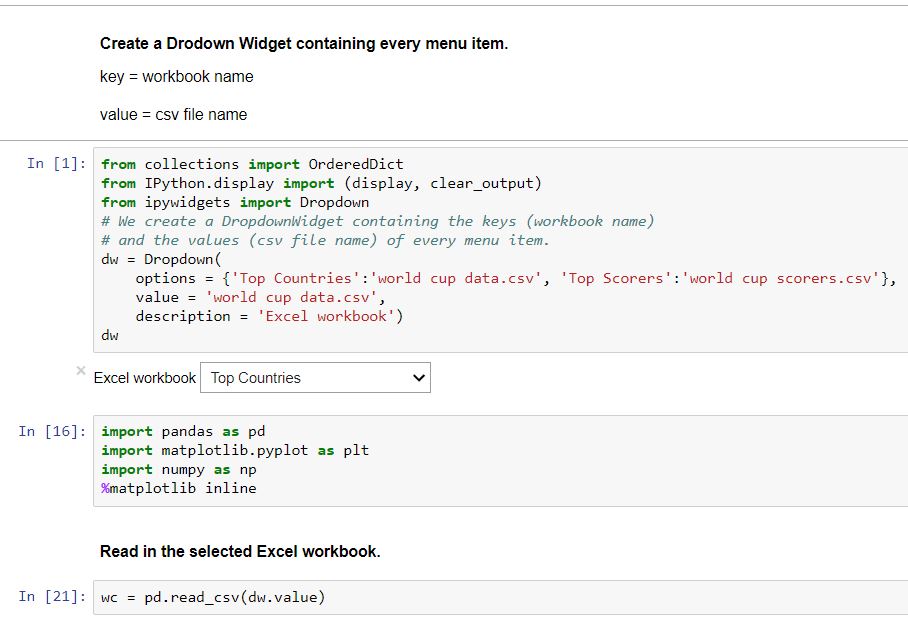


Figure 12. Markdown Notes

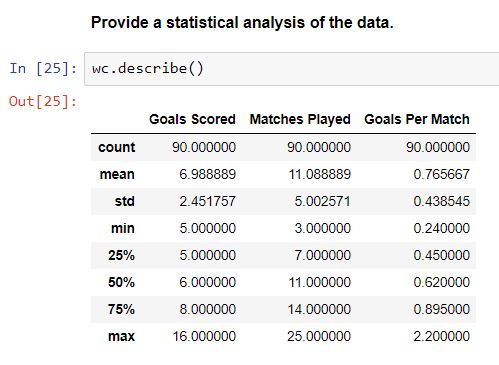


Figure 13. Markdown Notes



Figure 14. Markdown Notes

As with any project, run time is a crucial factor in algorithm efficiency and construction. If one has to wait several seconds for an algorithm to finish while debugging, he or she is most likely running the algorithm on too large of a data set. To reduce this run time and allow debugging to move more quickly, algorithms should be run on smaller sets of data before being run on full data sets. For the purposes of my project, I initially ran the project on a set of about eighty data points. I then ran the project on the full data set, after adding additional data points to each excel workbook. The code ran successfully and produced the same output for the top performers at the competition.

# Advanced IPython Notebook Techniques

Chapters 1 and 2 of this book dealt primarily with introductory IPython notebook functions, such as low-level data display and analysis. Chapter 3 moves on to more advanced techniques in which the IPython notebook is capable. By having a strong foundation in these lower-level functions, it should be much easier to understand these more complicated functions and techniques. For several of these high-level techniques, I will run code to accomplish the function and demonstrate it in use.

Every IPython notebook is saved as a JSON text file, which is a “text-based, open standard format that can represent structured data” (Rossant 89). This file contains all the code, text, outputs, and figures of the notebook. While JSON is a widely-used file format that can be parsed in many languages, it is sometimes beneficial to convert the notebook to another format, including, but not limited to, raw text, Markdown, HTML, and PDF. For the purposes of this project, I would like to convert my notebook to a raw text file which can be displayed using the Notepad application.

Use nbconvert to convert to raw text (ch3)

Custom controls

# References

Albon, C. (n.d.). Loading A CSV Into Pandas. Retrieved October 15, 2017, from https://chrisalbon.com/python/pandas\_dataframe\_importing\_csv.html

Rossant, C. (2014). IPython interactive computing and visualization cookbook: over 100 hands-on recipes to sharpen your skills in high-performance numerical computing and data science with Python. Birmingham (UK): Packt Publishing.

All-time table of the FIFA World Cup. (2017, October 14). Retrieved October 14, 2017, from https://en.wikipedia.org/wiki/All-time\_table\_of\_the\_FIFA\_World\_Cup

FIFA World Cup top goalscorers. (2017, October 22). Retrieved October 23, 2017, from https://en.wikipedia.org/wiki/FIFA\_World\_Cup\_top\_goalscorers

Jupyter Widgets. (n.d.). Retrieved October 19, 2017, from http://ipywidgets.readthedocs.io/en/stable/examples/Widget%20List.html

Matplotlib Bar chart. (n.d.). Retrieved October 14, 2017, from https://pythonspot.com/en/matplotlib-bar-chart/

Pandas.DataFrame.sort\_values. (n.d.). Retrieved October 18, 2017, from http://pandas.pydata.org/pandas-docs/version/0.19.2/generated/pandas.DataFrame.sort\_values.html#pandas.DataFrame.sort\_values