Gina O'Riordan, Marisa Roman, Justin Saslaw
Professor Paul Miller
DS64502
13 December 2017

R & Python in Exploratory Data Analysis

The exploratory data analysis (EDA) process can be broken down into four general steps according to Hadley Wickham: import, tidy, understand, and communicate (Wickham ix)[1]. The import stage brings all data for analysis into the programming environment where it is then tidied by ensuring "each variable must have its own column…each observation must have its own row…[and] each value must have its own cell" (Wickham 149)[2]. The understand step includes transforming the data into various datasets and dataframes for further analysis, modelling it to gain insight into trends or points of further interest, and visualizing the data in different types of graphs. Communicating this final analysis of hypotheses and/or insights can then be shared to a wider audience.

Both the R statistical and Python general programming languages can satisfactorily accomplish each step of an EDA. Each have specific packages and libraries, respectively, that can ease the data scientist's task of managing the data analysis; each have graphing and publishing capabilities to ease finalized report sharing as well. Though both languages are capable of enabling EDA, each have individual strengths and weaknesses in each step of the process, including their coding environments. Based on the team's experiences analyzing the National Longitudinal Survey of Youth (1979), there are advantages and disadvantages to both languages. Python is an object-oriented, dynamically-typed language whose packages are best used in the beginning of the EDA process, while the R ggplot and statistical packages make exploring and communicating data seamless. This paper will review the balanced strengths and weaknesses of R and Python to demonstrate an optimal combination of both languages when performing an exploratory data analysis.

The import step is easiest when using the numpy ("numerical python") library in Python. The pandas read_csv function ensures a smooth data upload and has a built in usecols argument to minimize the columns imported to only those needed and thus avoid extraneous memory usage (first code line, 'Income,' Appendix 2). On the other hand, R's load function only allows for file upload (first code line, 'Income,' Appendix 1); column filtering and column selection must be done separately (first code line, 'Joining the Data into a New Dataset,' Appendix 1). The tidying phase of the EDA is also most efficient when performed in Python. By using masking techniques, dropna(), aggregations such as max() and mean(), and groupby() functions, the data can be cleaned quickly and with minimal coding lines ('Truncation of Income Data', Appendix 2). Comparatively in R, it takes multiple steps to do the same task. The pipe function (%>% in code, Appendix 1) that feeds dataframes from one step to the next can avoid naming a variable for each function's line of code, but it does not minimize the amount of steps to properly tidy the data ('Truncation of Income Data', Appendix 1).

Understanding the data by transforming, visualizing, and modelling can be managed most efficiently in R although both R & Python accomplish this step well. The pipe function (%>% in code, Appendix 1) in R allows for quick exploration of data by enabling the data scientist to manipulate a dataframe progressively and in a continuous workflow environment ('Truncation of Income Data', Appendix 1). For instance when looking for summary statistics you can easily filter an initial dataframe, manipulate the dataframe for specific variables to explore, and create an output tibble with this data neatly displayed. This is a more intuitive process than Python, which requires multiple lines of code to achieve the same functionality along with the explicit definition of variables and incremental dataframes

---

[1&2] Wickham, Hadley, and Garrett Grolemund. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data.* O'Reilly, 2017.

('Truncation of Income Data', Appendix 2). Although Python requires more code to explore the data, it does have a very user-friendly summary statistic package using the .describe() function.

Once data is imported, cleansed, and analyzed, visual outputs help users both technical and non-technical draw conclusions regarding the problem at hand. While Python has numerous visualization libraries, this can also serve as a drawback as at times; it may not be clear what visualization tool is ideal for the given output. The visualization packages in R are made for statistical data analysis and offer very powerful alternatives including the ggplot suite whose arguments take into consideration a plethora of graphing options. Though both languages boast robust and readily available plotting packages, the ggplot package in R has advanced customization features that aren't available in the matplotlib package used in Python. For example, generating the same plot against multiple series of data requires the addition of one line of code in R (code preceding 'Mean and Median Income by Years of Education Completed, 2008-2014' plot, Appendix 1), whereas in Python, the plotting code must be repeated for each year being analyzed (code preceding 'Mean and Median Income by Years of Education 2008 – 2014' plot, Appendix 2).

The Integrated Development Environment (IDE) plays an important part of the data scientist's experience with R & Python as well as the ability to communicate results as the final step in the EDA process. The IDE is a coding tool used to write, test, and debug code during the data analysis process; R most frequently uses the IDE 'R Studio' and Python has no clear IDE as it is a general-purpose language. R Studio provides a dashboard for the data scientist; an accessible list of variables appears in the 'environment' pane and dedicated panes for file management and plot viewing put file management and different graphical outputs at the data scientist's fingertips. In contrast, the main Python IDEs Jupyter Notebook and Spyder do not display saved variables and users have to go outside the IDE in order to reach the file storage.

The benefits of the Jupyter Notebook come at the output and communication stage of an EDA, since it is a viable standalone presentation tool that supports markdowns and HTML components with a quick conversion to PDF format. Notes and headers can easily be added with two keystrokes (Esc + M) for easily marking up an EDA to present to viewers. Jupyter Notebooks also allow for Python data visualization libraries, including Matplotlib & Seaborn, to present graphs concurrent with code that can be exported to HTML or PDF files. On the other hand, the R Studio IDE needs R files to be "knit" to export in a user-friendly format, such as HTML or PDF. To include markdown language for presenting an EDA, the file needs to be saved as a specific R notebook file type, allowing for graphs to appear in line and for additional text formatting options seen in the Jupyter Notebooks.

Python is the choice for data import and cleaning that requires minimal lines of code to get into a usable format, while R is more cumbersome when importing and organizing data. Python's simplistic syntax makes it easy to write, read, and maintain the code throughout a project, minimizing time and effort spent working with it. Once a clean data set is created, R has a more robust graphical package with dedicated statistical analysis tools as compared to Python. As a language with data science roots, R has powerful visualization tools, with cutting edge packages designed solely for analyzing large data sets & drawing conclusions. As a general-purpose language, Python has an easier learning curve for those with a programming background and includes both libraries and IDE's focused on the data science community. Due to the significant benefits and efficiencies gained by using Python for importing and tidying data, and those gained by using R for exploring and visualizing data, we recommend the adoption of both languages so that our employees will have the best options at their disposal for performing all steps of an exploratory data analysis.

Appendix 1: R "Final Project Part C: Explore Factors Affecting Income in NLSY '79 Data"
Appendix 2: Python "Final Project Part B: Explore Factors Effecting Income in NLSY '79 Data"