Data Analysis with Machine Learning and Natural Language Processing for Web Presentation: Language Choice

By: Justin Renneke

Capstone I

4/13/17

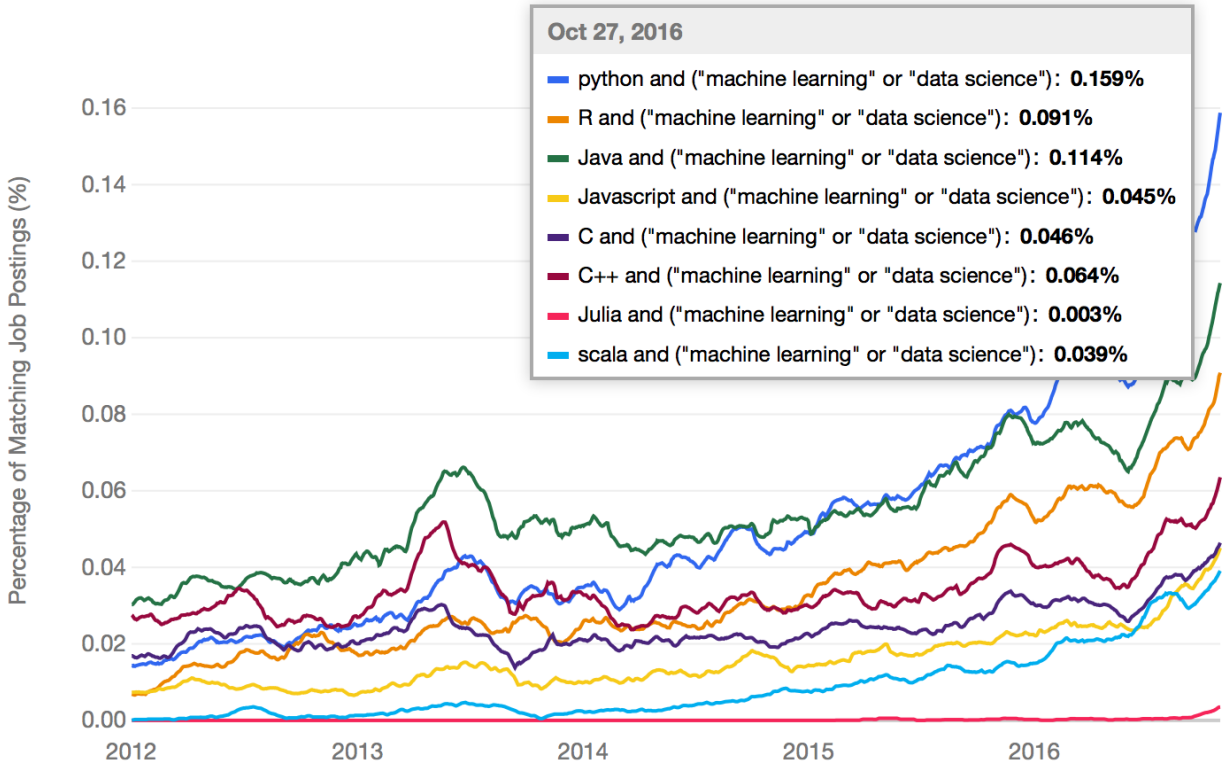
Choosing the right programming language for a project is like choosing the right construction material to construct a building. Wood and steel are both good building materials in their own way and both have been used to make impressive, well-engineered structures, but each has its own unique strengths and weaknesses. Choosing to build a skyscraper from wood instead of steel has the obvious downside of putting a relatively low limit on how tall it can be built. Building a family home from steel instead of wood has downsides such as higher initial cost and the requirement of more highly skilled and specialized contractors to build it, as well as more subtle issues that would only appear later in the home's life such as greatly reduced insulation capabilities due to the temperature conductivity of steel. However, by and large, a home is a home and a skyscraper is a skyscraper to the occupants inside. Similarly, the typical end-users of a well-designed software product will never know or care what language was chosen to construct it, but this choice has major implications for those who will code it and must live with its foibles and limitations. Just as in the choice of construction material, choosing a language well-suited to a project will reap many benefits for the developers of the project such as: facilitating the full realization of the project's scope by removing or avoiding limitations inherent in other languages; reducing time investment by accomplishing the same goal with a technology that is easier to use; or decreasing the cost of future maintenance. The problem addressed in this paper will be to choose the most suitable programming language for the back-end portion of a project involving the analysis of textual data for presentation in a web application.

The first of these two challenges - performing back-end data analysis on a large amount of text - has several key characteristics to consider that are specific to the nature of the

challenge. The first and most important factor to the success of this task is: the data analysis will involve statistical analysis and machine learning, but the developers do not have the time or expertise to create machine learning algorithms from scratch, so the language should have pre-existing machine learning libraries available. Second, the data will need to go through layers of cleaning and feature extraction using natural language processing, so support for this must exist. Finally, a huge amount of textual data will need to be processed, so performance is an important consideration. The second challenge - providing visualizations of the data analysis within a web application - will require dynamic interaction between the back-end and front-end systems in the project to allow for graphing and charting the results within an interactive web interface based on HTML, CSS, and JavaScript. In addition to these task-specific considerations, there are also some more general concerns to keep in mind. The first of these is that the developers are college students who have a relatively limited repertoire of language experience. Learning a difficult language such as C++ from scratch is not feasible for this project. In addition, the scope of this project is very ambitious considering the skillset of the developers, yet it must be completed within a timeframe of a few months.

Given the heavy focus on data analysis and machine learning, the initial language candidates can be narrowed down to three for in-depth analysis: R, Java, and Python. This decision is supported by an article on the IBM DeveloperWorks blog[1] which explored the question of which languages are the most popular to use for machine learning and data science

by searching indeed.com job postings and the results can be seen in the chart below.



An analysis of each of the factors integral to the success of the project as outlined above done within the context of these three candidate languages will reveal the best tool for the job.

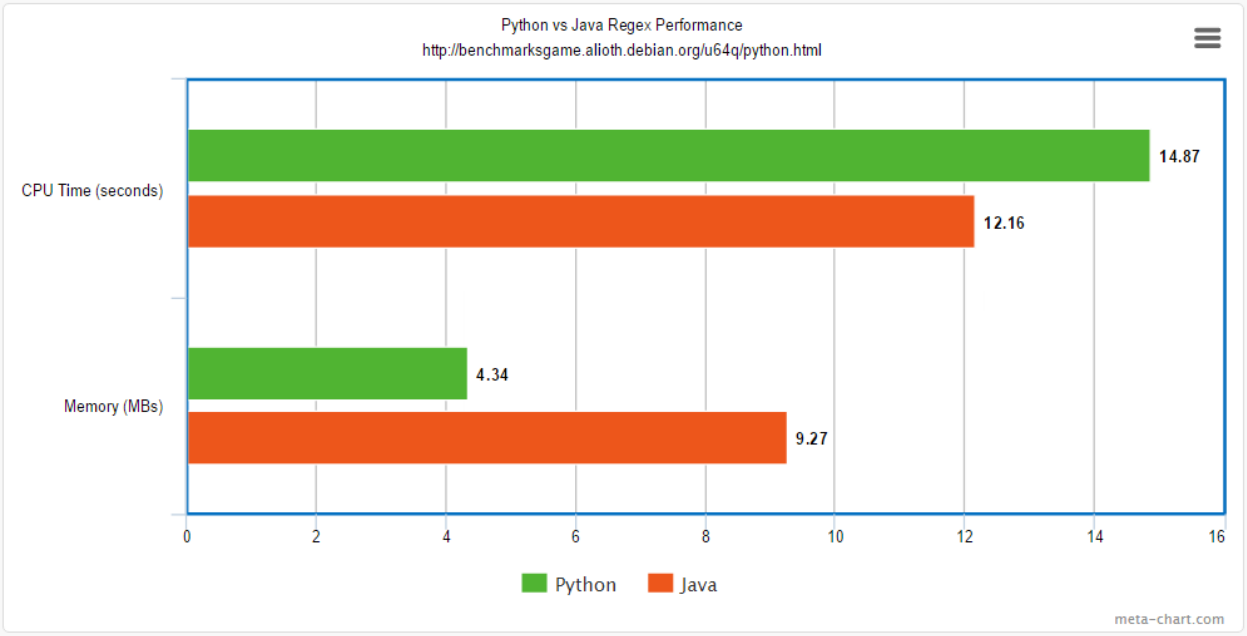
The first required capability - the availability of robust support for machine learning via libraries - is well supported by all three candidates as expected given that they were chosen based on this, the most important criteria for the success of the project. According to a highly upvoted post [2] on Stackoverflow that compares the machine learning libraries of R and Python, R has over 5000 (somewhat fragmented and overlapping) libraries that support machine learning and data analysis while Python has several comprehensive libraries such as Pandas, Scikit Learn, Scipy, and Matplotlib that offer similar capabilities. Java too, provides very good machine learning libraries ranging from Weka, which provides not only a machine learning

API but also a graphical user interface to facilitate experimentation, to Java-ML. This capability is essentially a tie across the three languages as each language provides library support for the full range of the data manipulation and machine learning analysis pipeline from pre-processing and feature generation to analysis using clustering, classification, and other techniques.

Secondly, the textual nature of the data being processed and analyzed requires that the language provide extensive support for natural language processing (NLP). This highly regarded reply [3] to a Stackoverflow question asking if Python or Java was a better choice for an NLP project states that both of these languages offer excellent support for this task. Python has a large and full-featured NLP library in the form of the Natural Language Toolkit (NLTK) library and Java offers multiple powerful NLP libraries such as the comprehensive OpenNLP library and the highly regarded and flexible StanfordNLP library. R, on the other hand, appears to lose out in this category. While R does offer support for NLP in the form of its Text Mining Package (tm), this blog post[4] from a data science course teacher suggests that “R's primary NLP framework (the tm package) [is] significantly more limited and harder to use [than Python’s NLTK].” Anecdotal evidence gathered from Google searches seems to support R’s inferiority in NLP. While there are a great many results for Python and Java in regards to NLP, there are relatively few for R NLP and those that do exist are lesser quality and scope, at the very least indicating that the support community for R NLP is not up to par.

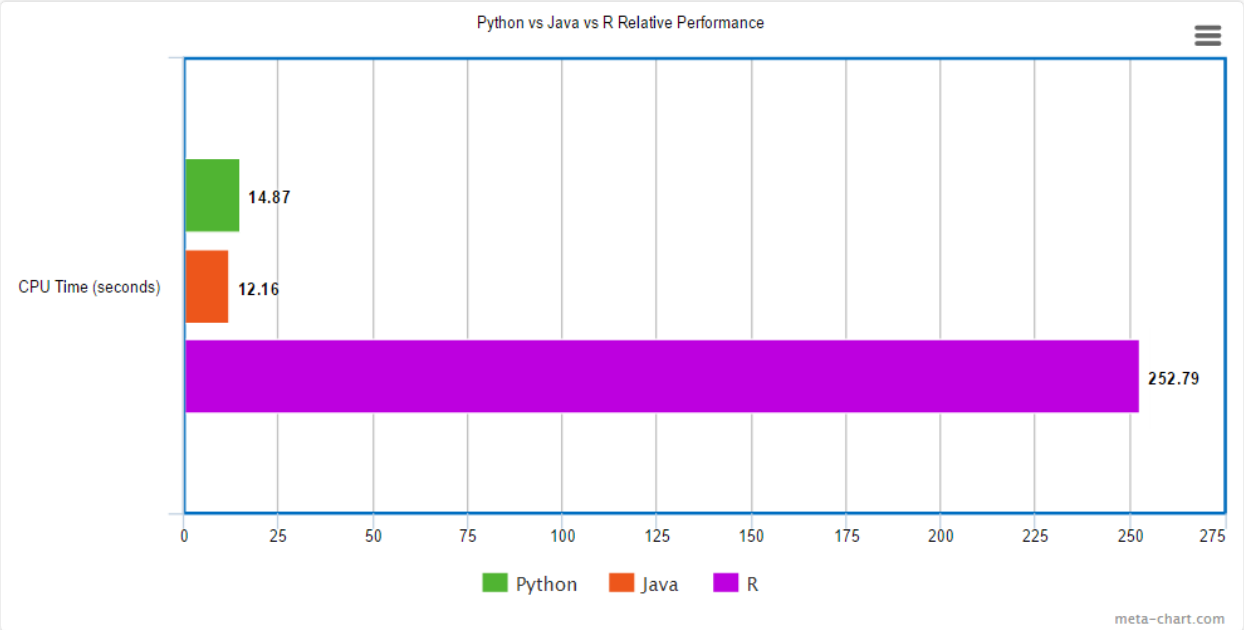
The third consideration, raw performance of the language when processing strings, is important due to the sheer amount of data that will need to be processed. This can be difficult to assess to a certain degree because it is highly dependent on the quality of effort spent to write optimized project code. It also depends on libraries leveraged - some languages have

libraries that interface with C code to do heavy computations, thus presenting the opportunity to achieve a significant performance advantage over the base language - and the type of computation, among other things. There is also the issue of choosing which metric to emphasize: processing speed, memory, or something else. The following chart from a website [5] comparing performance metrics across several languages shows the results from Python and Java when evaluating a Regex expression (lower numbers are better).



Unfortunately, similar data was not available for R, but research into R's performance relative to other languages shows a general consensus among data scientists that R suffers in comparison. A companion website [6] to the R textbook *Advanced R* states that "R is not a fast language. This is not an accident. R was purposely designed to make data analysis and statistics easier for you to do. It was not designed to make life easier for your computer." Another article [7] from a blog dedicated to discussion of the R language compared R's performance with Python's when calculating prime numbers and found that R was 17 times slower than Python in

this case. Extrapolating this statement to the CPU time metric in the graph above with the understanding that this is only for purposes of visualization due to the significant difference in context, we see the big difference in the chart below.



Based on these results, Python and Java have similar relative performance, with Java slightly edging out Python in CPU time, but losing out on memory consumed. R would appear to be a distant third.

Next, since the end results of all of this data analysis is meant to be displayed in a web application, the languages' ability to interface with a web application must be considered. Each of the three candidate languages has support for this to some degree. R has a handful of web frameworks including the Shiny framework that supports the development of interactive web pages and is particularly strong where visualization of data is concerned. Python has many web frameworks from the extensive Django to the lighter weight Flask, and many in between. Java, being the oldest and most heavily used enterprise language, has a wide variety of web

frameworks including Spring MVC, Grails, and innumerous others. R's Shiny provides truly impressive visualizations, but seems to be limited in scope compared to the others as it is mainly used to create dashboard-type setups, providing little interactivity and has a smaller community for support. This project already plans to use Javascript's D3.js library for most front-end data visualizations due to team members' pre-existing experience with the technology, so the main selling point for Shiny is not as important in this case. Python web frameworks such as Flask have a reputation for being relatively easy to set up, for being flexible and facilitating fast development, and for being well-suited to small to medium sized applications developed by small teams, but not as ideal for large teams working together to develop modular, high-performing applications. Java web frameworks such as Spring MVC are generally well-regarded for providing high performance, a truly thorough toolkit given the massive enterprise-level library support, and providing good support for very large teams to work together, but require large investments of time and skill to set up and have high learning curves. These findings are summarized in the table below. R's Shiny, Python's Flask, and Java's Spring MVC are used as general reference points.

General Comparison of Web Frameworks on a Scale of 1 to 5 (5 is best)

	R Frameworks	Python Frameworks	Java Frameworks
Ease of Use	4	3	1
Flexibility	1	4	5
Scope	2	4	5
Speed of Development	4	5	2
Support Community	2	4	5
Total Score	13	20	18

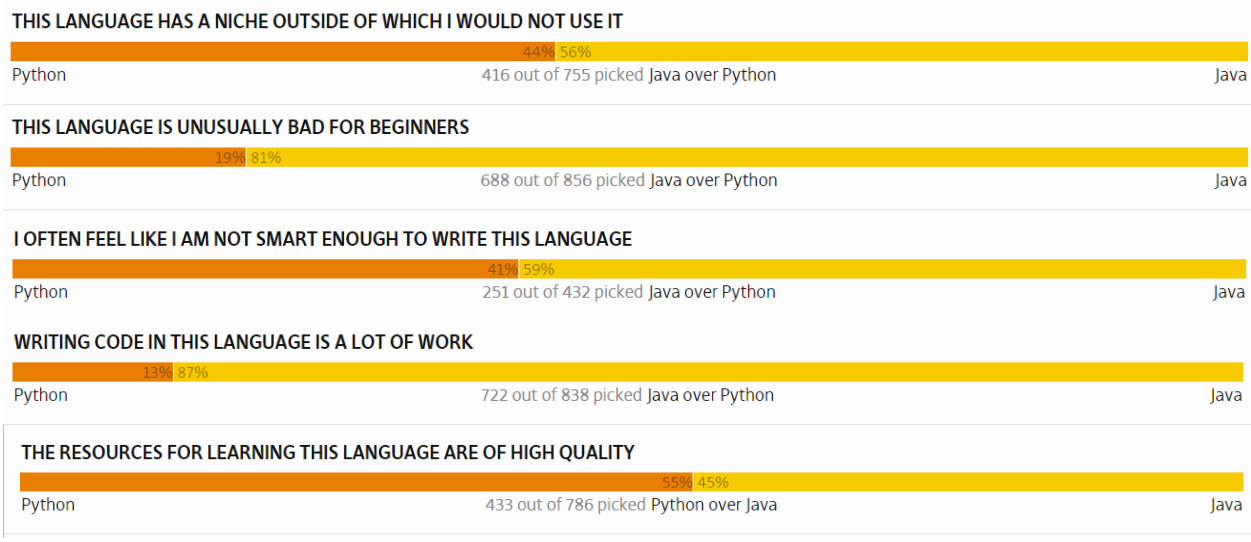
A discussion of Python vs Java web frameworks on Quora [8] supports many of these arguments. In this project's case, Python's web frameworks are the best fit.

Finally, the circumstances under which the project will be developed must be considered. Given that the developers have a limited range of experience with different programming languages, the ease of learning the candidate languages should be considered since some team members will have to learn the language from scratch no matter which is chosen. In addition, this ambitious project must be completed within a few months while the developers are learning as they go, so a robust ability to support rapid development will be essential. R is viewed as a language designed for statisticians more so than for traditional developers. This has led to relatively intuitive syntax but also some counterintuitive handling of traditional programming structures due to this fact. The general consensus is that R is a very specialized language that, while not particularly difficult to get started with, is not as easy to jump into as a more traditionally designed language for people who already know other programming languages and have a pre-existing understanding of program design and logic flow. Also, while R's specialized nature makes it well-suited for machine learning and statistical analysis, that same specialization limits its capabilities in the wider functionality required by the project. Java lies on the opposite end of the spectrum from R. Java has a massive amount of libraries providing developers with a huge range of toolkits. However, it is a highly structured and rigorous language that demands a solid understanding of programming principles from those who would use it. It provides high performance and one of the most extensive object-oriented design paradigms, but inherently has a high learning curve to achieve this. Java is very verbose, has a high overhead to start with, and requires a thorough understanding of object

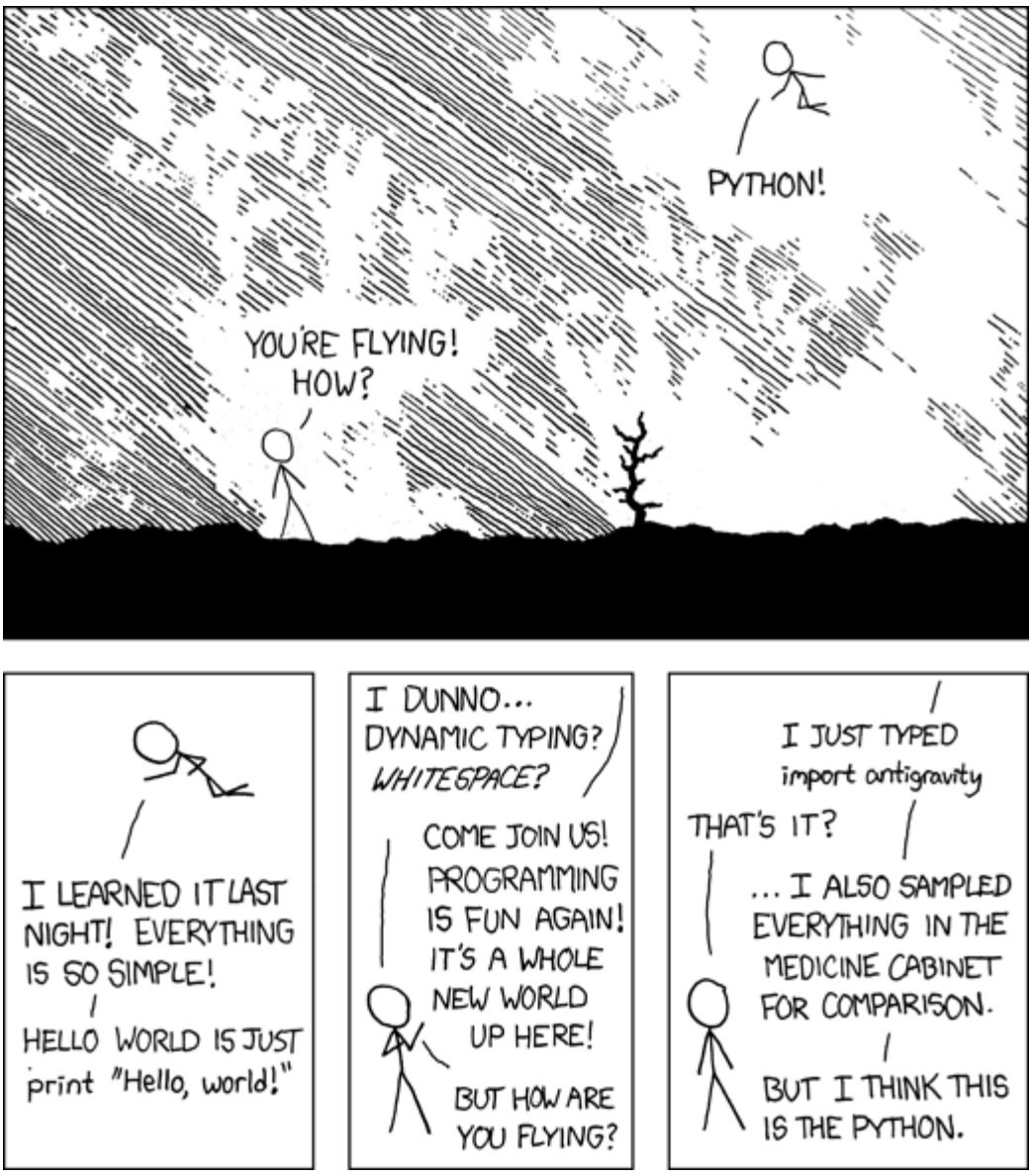
oriented design structures to make use of it. Python, in contrast, is widely regarded as one of the easiest programming languages to learn due to its friendly syntax coupled with traditional programming structures and is famed for allowing developers to accomplish tasks very quickly due to an extensive collection of powerful and easy to use libraries. Users opinions regarding these metrics can be found in the surveys provided here. [9] Some key comparisons can be seen below.

Python vs R:

Python vs Java:



Python's benefits appear to especially shine given the real world constraints of development for this project. The following graphic does a good job of summarizing the communal consensus. [10]



In conclusion, the most suitable language for the back-end data analysis involved in this project is Python. The other two languages considered fall short in some key ways. R is a great language if you are focused on statistical and machine learning-based analysis of numerical data to create visualizations, but appears to be lacking where NLP is concerned. It is also limited

in scope by its lower performance and limited support for web frameworks in addition to being slightly harder with which to work. Java is a very structured, high-performance language with a vast collection of powerful libraries covering everything from machine learning and NLP to web deployment, but these characteristics also cause it to have a high development overhead, giving it a steep learning curve and making it difficult to work with and a struggle to quickly prototype and accomplish goals for the inexperienced. On the other hand, Python provides extensive support for machine learning and natural language processing-based data analysis with libraries such as Scikit Learn and NLTK. Python also provides good performance in comparison to the other candidate languages as well as flexible and relatively easy to use web frameworks for communication with the front-end of the project. Finally, Python is the clear first choice given the time and skill constraints of the development team - it is a relatively easy language to learn and allows developers to quickly accomplish goals with numerous easy to use libraries. R and Java are powerful languages in their own way and the project may even benefit from using some of their specialized libraries which have been discovered over the course of researching this issue. Even then, it turns out that Python wrappers already exist for the most popular of the R and Java libraries, so it seems developers can have their cake and eat it, too, with Python.

Sources:

[1] Puget, Jean Francois. "The Most Popular Language For Machine Learning Is ... ." IBMDeveloperWorks. N.p., 19 Dec. 2016. Web. 17 Apr. 2017. <https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Language_Is_Best_For_Machine_Learning_And_Data_Science?lang=en>.

[2] Binga. "Python vs R for machine learning." Data Science Stack Exchange. N.p., 3 Jan. 2017. Web. 17 Apr. 2017. <https://datascience.stackexchange.com/a/339>.

[3] Alvas. "Java or Python for Natural Language Processing." Stack Overflow. N.p., 12 Apr. 2017. Web. 17 Apr. 2017. <http://stackoverflow.com/a/22905260>.

[4] Markham, Kevin. "Should you teach Python or R for data science?" Data School. Data School, 18 Apr. 2016. Web. 17 Apr. 2017. <http://www.dataschool.io/python-or-r-for-data-science/>.

[5] "The Computer LanguageBenchmarks Game." Python 3 vs Java. N.p., n.d. Web. 17 Apr. 2017. <http://benchmarksgame.alioth.debian.org/u64q/python.html>.

[6] Wickham, Hadley. "Performance." Performance · Advanced R. N.p., n.d. Web. 17 Apr. 2017. <http://adv-r.had.co.nz/Performance.html>.

[7] Simmering, Jacob. "How slow is R really?" R-bloggers. N.p., 28 Jan. 2013. Web. 17 Apr. 2017. <https://www.r-bloggers.com/how-slow-is-r-really/ >.

[8] Mittal, Chandan. “Which is better for web applications, Java or Python?” Quora. Web. 18 Nov. 2017. <https://www.quora.com/Which-is-better-for-web-applications-Java-or-Python>

[9] "Programming languages." Programming Languages | Hammer Principle. N.p., n.d. Web. 17 Apr. 2017. <http://hammerprinciple.com/therighttool>.

[10] Munroe, Randall. Xkcd: Python. N.p., n.d. Web. 17 Apr. 2017. <https://xkcd.com/353/ >.