SMU Data Science Review

Volume 1 | Number 2 Article 7

2018

Data Scientist's Analysis Toolbox: Comparison of Python, R, and SAS Performance

Jim Brittain
Southern Methodist University, jbrittain@mail.smu.edu

Mariana Cendon
Southern Methodist University, mllamascendon@smu.edu

Jennifer Nizzi
Southern Methodist University, jnizzi@smu.edu

John Pleis
National Center for Health Statistics / CDC, gzp4@cdc.gov

Follow this and additional works at: https://scholar.smu.edu/datasciencereview

 Part of the Applied Statistics Commons, Other Computer Sciences Commons, Programming Languages and Compilers Commons, Software Engineering Commons, and the Statistical Methodology Commons

Recommended Citation

Brittain, Jim; Cendon, Mariana; Nizzi, Jennifer; and Pleis, John (2018) "Data Scientist's Analysis Toolbox: Comparison of Python, R, and SAS Performance," *SMU Data Science Review*: Vol. 1: No. 2, Article 7.
Available at: https://scholar.smu.edu/datasciencereview/vol1/iss2/7

This Article is brought to you for free and open access by SMU Scholar. It has been accepted for inclusion in SMU Data Science Review by an authorized administrator of SMU Scholar. For more information, please visit http://digitalrepository.smu.edu.

Data Scientist's Analysis Toolbox: Comparison of Python, R, and SAS Performance

Jim Brittain[1], Mariana Llamas-Cendon[1], Jennifer Nizzi[1], John Pleis[2]

[1] Master of Science in Data Science, Southern Methodist University
University 6425 Boaz Lane, Dallas, TX 75205
{jbrittain, mllamascendon, jnizzi}@smu.edu
[2] National Center for Health Statistics – Centers for Disease Control and Prevention

Abstract. A quantitative analysis will be performed on experiments utilizing three different tools used for Data Science. The analysis will include replication of analysis along with comparisons of code length, output, and results. Qualitative data will supplement the quantitative findings. The conclusion will provide data support guidance on the correct tool to use for common situations in the field of Data Science.

1 Introduction

All professionals need to utilize the best tools for their tasks. Veteran professionals incorporate the learning from the experiences of their careers while inexperienced individuals look for guidance.

Many articles offer preferences based on popularity, cost, ease of use, data handling, visual capabilities, advancements, technical/community support and career opportunities. The preferences are valid; however, the articles often include bias and qualifiers that are not measurable. In response, the research of this paper will focus on quantifiable and qualifiable attributes to offer comparison of multiple tools with a focus on performance.

The potential tools for a data scientist are numerous. The initial selection was based on public information as well as the tools in the Southern Methodist University (SMU) Master of Science in Data Science curriculum. The public information included research from renowned sites dedicated to data science and data analysis, Burtch Works and KDNuggets. An article by KDNuggets included Python,[1] R,[2] and SAS[3] in the top 4 tools for analytics and data mining [1]. In 2017, Burtch Works conducted a flash survey with over one-thousand data professional to assess the preferences for Python, R, and SAS [2]. As a result of the research, this paper will focus on these tools.

The motivation for this paper could be easily summarized by a quote by Tim O'Reilly in his article "What is Web 2.0" [3]: "Without the data, the tools are useless; without the software, the data is unmanageable." The findings of the paper will offer

[1] Python Software Foundation, [Online]. Available: https://www.python.org/.
[2] The R Project, [Online]. Available: https://www.r-project.org/.
[3] SAS Institute, [Online]. Available: https://www.sas.com/en_us/home.html.

comparative information to data driven individuals to provide insight into the comparison of the performance of three common data science tools. The performance comparisons will include data wrangling, visualization, and linear regression tasks with measurements on code complexity, computing time, and computing resources.

2 Literature Review

According to the 2017 Data Scientists Report by CrowdFlower, over 50% of time is spent collecting, labeling, cleaning and organizing data (Fig. 1 shows full time allocation) [4]. With the high percentage of time invested in the beginning of the process, the necessity to select the correct tool is paramount for the efficiency of a data scientist.

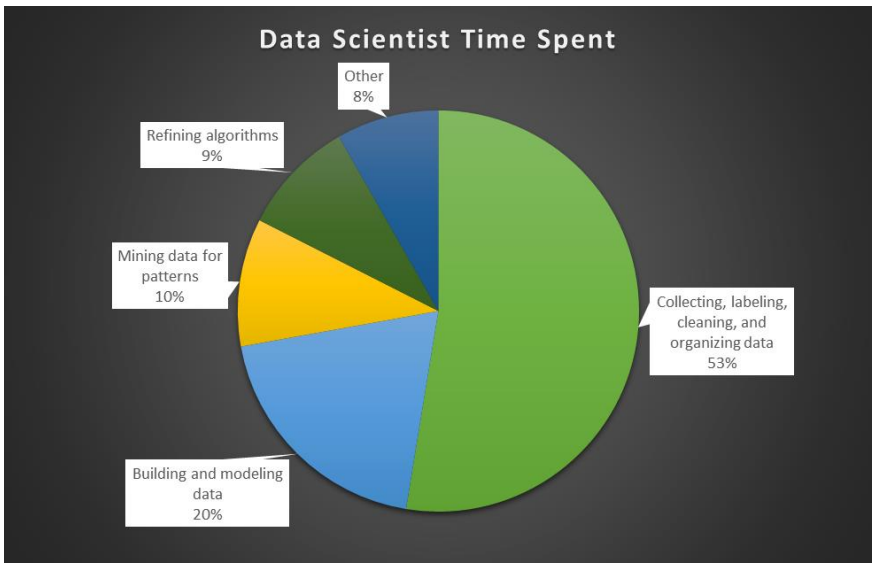

Fig. 1. Time allocation of data scientists based on a survey conducted in February and March 2017 including 179 data scientists globally representing varying companies. More than 40% of the companies represented were technology ones.

Over four decades ago, formulas were developed to measure the complexity of algorithms and languages. The pioneer of this field is Maurice H. Halstead credited with the metrics now known as the Halstead complexity measures. Robust research and testing were done on these measurements [5], [6], [7]. The number of operators (N_1) and operands (N_2) are identified along with the unique operators (n_1) and operands (n_2). Calculations are then performed on these numbers to provide the program vocabulary (n), program length (N), volume (V), difficulty (D), effort (E), and time (T).

$$n = n_1 + n_2 \quad \textbf{(1)}$$

$$N = N_1 + N_2 \quad (2)$$

$$V = N \times log_2 n \quad (3)$$

$$D = \frac{n_1}{2} \times \frac{N_2}{n_2} \quad (4)$$

$$E = D \times V \quad (5)$$

$$T = \frac{E}{18} \quad (6)$$

In this context, an operator has the ability to manipulate and check on the values of an operand; while an operand is either a numeric, text and/or Boolean values able to be manipulated.[5] There is not a strict convention as to what defines an operator and an operand therefore a single code script could return different counts of these attributes depending on the criteria used to select them [8].

The Cyclomatic complexity model, also known as McCabe's complexity, was also developed in the 1970s [9]. The cyclomatic complexity focuses on the number of edges (e), vertices (n_{cc}), and connected components (p).

$$v(G) = c - n_{cc} + 2p \quad (7)$$

Since the introduction of the models, criticism has been voiced on both models. One concern is that complexity of code may represent more than the complexity of the language. The complexity may represent the complexity of the tasks being performed by the code or less direct coding practices [10]. Another concern is the direct correlation between the complexity and the lines of code [11]. Despite the concerns, the complexity measurements by both Halstead and McCabe continue to be used.

3 Overview

3.1 Python

Python is an open source general purpose tool with applications[4] for web, Internet, and software development; education and academia; numeric and scientific, to mention a

[4] Python Software Foundation, "Applications for Python," [Online]. Available: https://www.python.org/about/apps/. [Accessed November 2017].

[5] webopedia, "operand," [Online]. Available: https://www.webopedia.com/TERM/O/operand.html [Accessed April 2018].

few. Python—created by Guido Van Rossum as the successor of the ABC language and officially released in 1991—relies on the contribution of its wide community of users and developers self-identified as PUGs (Python User Groups)[5] for its continuous evolution and growth. There is a scientific community of "well-established and growing group of scientists, engineers, and researchers using, extending, and promoting Python's use for scientific research" [12].

Python capabilities are extended through its robust collection of packages. As of today, PyPI, also known as the "Cheese Shop,"—the official package repository—has more than 100,000 packages stored[6]. Roughly explained, a package is a collection of modules that in turn contain definitions and statements to execute functions or determine classes.

In the field of data analysis some of the common packages [13] are: Pandas—ideal for data manipulation—; Statsmodels—for modeling and testing—; scikit-learn—for classification and machine learning tasks—; NumPy (Numerical Python)—for numerical operations—and SciPy (Scientific Python)—for common scientific tasks. A recent survey [14] found that Python's NumPy, and SciPy packages were among the most preferred ones for statistical analysis, while scikit-learn stood as a data mining favorite.

Python also provides an extensive list of Integrated Development Environments (IDE). According to DataCamp [15] among of the top ones for data science: Spyder, a cross-platform IDE distributed through Anaconda (a "freemium" open source distribution for large-scale data); PyCharm integrates libraries such as NumPy and Matplotlib and provides support for JavaScript, KTML/CSS, Node.js, making it a good interface for web development; and Jupyter Notebook, previously known as IPython, "offers an end-user environment for interactive work, a component to embed in other systems to provide an interactive control interface, and an abstraction of these ideas over the network for interactive distributed and parallel computing [16]."

3.2 R

The development of R was inspired by S with some programming influences from Scheme [17]. Two professors introduced the language to assist students with a more intuitive language, specifically lexical scoping which eliminates the necessity for global defining of variables [18].

Although the history R can find a foundation in FORTRAN, R is its own language. R is an interpreted language with code directly executed rather than compiled. Using a compiler, programmers can write interfaces for C, C++, and FORTRAN for efficiency.

R is part of the GNU Project, which focused on free software allowing users the ability to run, redistribute, and improve the program [19]. Although initially criticized, R upgraded quickly with collaboration from around the globe. Since R is open source,

[5] Python Software Foundation, "Diversity Statement," [Online]. Available: https://www.python.org/community/diversity/. [Accessed March 2018].

[6] Python Software Foundation, "PyPI - the Python Package Index: Python Package Index," [Online]. Available: https://pypi.python.org/pypi.. [Accessed November 2017].

the target audience is any user interested in statistical computing. R can be installed using Unix, Windows, or Mac.

R is available for download via the Comprehensive R Archive Network (CRAN). The master site is in Austria; however, mirrored sites throughout the world distribute the load on the network. In addition to the software, the CRAN hosts provide supporting documentation and libraries with add-on packages. The open-source add-on packages, which are groups of functions developed by other users, are available on CRAN. As on January 27, 2017, the CRAN hosted more the 10,000 packages which does not include packages from other vendors [20]. Although no warranties are given by R for any packages on CRAN, all the package contributions are reviewed by the CRAN team. Some packages in the libraries may restrict commercial use although the same packages may be openly available for education and research.

RStudio is an IDE using packages (knitr and rmarkdown) to develop composed documents with the code and output from the R language. In addition, RStudio is an editor for LaTeX which is a markup language to produce high quality documents. A 2011 poll rated RStudio as the most used IDE with only the basic R console more frequently used [21].

3.3 SAS

SAS is a proprietary comprehensive statistical and data management tool developed by the SAS Institute; used internationally by government, private industry, and academia. "94 of the top 100 companies on the 2016 Fortune Global 500® are SAS customers."[7] SAS is the largest privately-owned software company in the world.[8] Once an acronym for Statistical Analysis System; SAS has grown into much more than that and is no longer considered an acronym. It was originally created in 1966 for agricultural research work and later developed into a full-fledged system with the inception of SAS Institute. A study released in 2016 by MONEY and PayScale.com listed "Making Sense of Big Data" as the most valuable career skill now; with SAS as the top skill [22]. Current uses include business intelligence, and analysis of data in almost every business sector.

SAS has various components and products that can be licensed along with Base SAS which is the core procedures and data management tool. SAS is a static typed language that uses the proprietary SAS dataset as the main table style data structure with only 2 data types numeric and character.

SAS does not have the large user-written package library common to R and Python. There is however a huge user base that write and share code; it is just not included and centralized in the same way. SAS has its macro language which allows users to write code that can take various parameters and encapsulate code similar to functions in other languages. A user would then reference the code and call the macro. Unlike Python and R, this macro code is not compiled and does not get installed. It is SAS macro language

[7] SAS Institute, "SAS Institute (current) SAS – History," [Online]. Available: https://www.sas.com/en_us/company-information.html#history. [Accessed November 2017].
[8] "SAS Institute," 3 December 2017. [Online]. Available: https://en.wikipedia.org/wiki/SAS_Institute. [Accessed December 2017].

and Base SAS code that a user can read and modify as needed. A good source for this type of code is GitHub.[9] The SAS Global Forum, an annual conference for users by users, is a great source of SAS knowledge and code sharing. SAS provides an extensive "Knowledge Base" on the support section of their website[10] and has well-supported user support groups including the SAS-L list-serve hosted by the University of Georgia.[11]

The main IDE for SAS is referred to as Foundation SAS or generally known as PC SAS. SAS Enterprise Guide was released several years ago; mainly known as a more point-and-click method of coding in SAS. More recently, SAS developed the SAS Studio which is a platform agnostic alternative that is based in Java and runs in a web browser from a licensed SAS install. SAS also recently introduced Jupyter integration where SAS can be run from Jupyter with a licensed install on the machine running Jupyter.

4 Ethics

The ethics surrounding this paper are focused on the presentation of the facts and data of the tools without personal bias or influence. To eliminate potential bias in writing and results, all experiments and research was performed with a focus on the quantitative or qualitative data.

An additional ethical perspective is added when reviewing the code of conduct from the ACM (Association for Computing Machinery) [23]. The code of conduct defined the contribution to society and human well-being as well as creating opportunities for members to learn the principles and limitations of computer systems. The research to assist less experienced individuals entering the field is an effort to assist with guidance by providing all of the resources applied on this paper as an educational tool.

The source of any external code used in this paper was appropriately credited in the corresponding tool.

5 Methodology

The comparisons of performance needed to be quantified to provide the most unbiased substantive information. Some measurements of performance were not able to be fully quantified without the creation of rubrics which could be developed with bias. The experiment was separated into the quantifying and qualifying elements. The experiments were created to demonstrate the capabilities and limitations of tools and not to glean statistically sound data analysis.

[9] GitHub, "GitHub (current) Trending-SAS," [Online]. Available: https://github.com/trending/sas. [Accessed November 2017].
[10] SAS Institute, "SAS Institute (current) Knowledge Base," [Online]. Available: https://support.sas.com/en/knowledge-base.html. [Accessed December 2017].
[11] SAS Institute, "SAS Community (current) SAS-L," [Online]. Available: https://support.sas.com/en/knowledge-base.html. [Accessed October 2017].

5.1 Quantifiable Experiments

Two projects were identified to apply multiple traditional applications. The experiments were performed on two (2) machines. Each test run of the comparative programs were run with all extraneous applications closed. The application running the test was restarted in between program runs.

Table 1. Machine specifications.

Specification	Machine1	Machine 2
Processor	AMD A8-7410 APU 2.20 GHz	Intel Core™2 Quad CPU Q6600 @ 2.40 Ghz
HD Size	921 GB	C: 75 GB / D: 931 GB
HD Free Space	664 GB	C: 6 GB / D: 318 GB
RAM	8 GB	8 GB
OS	Windows 10	Windows 10

Table 2. Software specifications for both test machines.

Tool	Version	Installation
Python	3.6.4	Local PC
R	1.1.4	Local PC
SAS	9.4 (M3)	Local PC

Project 1 – Mortality Analysis. The first project for our testing represents a data wrangling task prepping data for further analysis. Mortality data was obtained from the National Center for Health Statistics / CDC website[12]. The mortality data contains both demographic information as well as cause of death information for all reported death certificates in the United States. Cause of death information is a complex structure including various coding and multiple contributing factors for the cause of death. The files were downloaded to the local machine for the experimental program performance runs. This data is relatively large with approximately 2.5 million records per yearly file for all of the U.S. states. There are also files for the U.S. territories that are approximately 30,000 records per year. We used the territory files for 2010 – 2016 as test data since it is larger although not as large as the full data for the United States. The project was then to use the full United States files.

The objective of the project was to focus on data wrangling without drawing conclusions. The initial code was developed in SAS and then replicated into R and Python. The data wrangling was sequential when necessary although some deviations occurred when they did not affect the outcomes.

The experiment started with the reading of the files and consolidated into a single dataset. The consolidated data contained 117 variables with 219,613 records. The data wrangling included replacing variable names with human readable values or variable labels. The codes were replaced with human readable value labels or formats for both race and one of the cause of death variables. Two variables to indicate if diabetes or

[12] Data can be retrieved from https://www.cdc.gov/nchs/nvss/deaths.htm

hypertension were a contributing cause of death were created based on the values of 20 different contributing factors to the cause of death. Frequency tables were then run to show the deaths per year by gender, cause of death (ICD-10 code recoded to 39 classification of causes), cause of death (ICD-10 code recoded to 113 classifications of causes), race, diabetes, and hypertension.

Project 2 – Accident Fatalities Analysis. The second project represents a more traditional data analysis task performing various analytical tasks. Accident data was obtained from the Fatality Analysis Reporting System (FARS) of the National Highway Traffic Safety Administration (NHTSA)[13]. A single file focusing on accident data for the year 2015 was used. The original file contained 32,167 observations and 52 variables. The data was preprocessed and cleaned before usage. Variables with less than half of the observations recorded were dropped, and unknown values were converted to actual missing values.

The objective of the project was the exploratory data analysis and multiple linear regression for fatalities. Initially, a model considering more than thirty (30) regressors was defined to understand the relation between variables and the number of fatalities involved in a vehicle accident.

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + \varepsilon \quad \textbf{(8)}$$

The purpose of the model, however, was to compare its performance against other tools and not its prediction power. Therefore, its accuracy was not taken into consideration to fulfill the task at hand. The initial code was developed in Python and then replicated into R and SAS.

The experimentation started with the loading of the data. Multiple bar graphs and a boxplot were generated to visualize the data. A correlation matrix was developed, and a heatmap was created for visualization of the correlation. A cross tabulation was performed and graphed to show drunk drivers and the number of fatalities. To begin the linear regression, dummy variables were created for the categorical variables with several levels: weather, day of the week, and lighting condition. A linear regression was performed on 34 variables. The statistically significant variables were selected based on ordinary least squares. For consistency every experiment used the same eight (8) variables selected from the Python analysis. The final model with eight (8) variables was trained using 80% of the data and tested with the remaining 20% of the data. The mean absolute error, mean squared error, root mean squared error, and variance were calculated to provide the analysis of the model.

5.2 Code Complexity

The guidelines defined by the Halstead Metrics were applied to the code for all three programs. The guidelines were originally defined for analysis C, so adaptations needed to be made. Although programs exist for computing the metrics, the existing applications do not apply consistently to all three tools.

[13] Data can be retrieved from ftp://ftp.nhtsa.dot.gov/fars/2015/

To ensure consistency in the counting of operands and operators, the counting was done by two or more individuals with consensus on all assignments. The code was reviewed line by line with each expression being identified individually. The total operators and operands were consolidated to provide a total. The unique operators and operands were then identified. All calculations for other complexity measurements were based on the operand and operator counts.

5.3 Qualifiable Research

There are certain considerations to be taken into account when selecting the right application for executing data analysis tasks: A programming language for data analysis should be easily writeable and readable by humans not by computers only, able to handle various data types whether those are arbitrary in nature, have different options to manage missing values properly, and provide at least basic mathematical and statistical functions such as the ability to generate random numbers and probabilistic distributions, as well as high-level visualizations [24]. To evaluate the selected tools, a matrix with these attributes was developed. All variables were included except the attribute simplicity. Simplicity was excluding because coding is a creative activity that may vary in length and style. Also, previous programming experience would influence the perception of simplicity.

6 Results

The experiments and research provided both quantifiable and qualifiable research as defined previously. Key aspects of the experiments and research were identified to offer comparisons of the performance. Common measurements in the experiments were number of lines in the code. The count eliminated all comments and white space and only focused on the necessary code to perform the activities identified. The selection of lines and words was to illustrate how the tools compare when looking at the amount of code that is required to perform a task. Program running time measured at various points and overall were measured in "wall clock" time, the time it takes the process to complete on the given hardware. Machine specs affect the performance in different ways depending on the tool. Using 2 machines allowed us to give a more balanced performance score of average machines. We considered running also on a Mac, however SAS is not available in native form or local install for the Mac. Typically Mac users use Citrix or other similar virtual machine to run SAS; in which case the code is actually processing on a remote machine and this would not be a good comparison across tools.

Many results were obtained from the experiments. The summary below highlights key results for the tools. The one test that we performed was to run the Mortality data on the full U.S. state file with ~2.5 million records per year. When we attempted this, SAS was the only one to complete the job. Python and R both threw errors once the data exhausted the RAM on the machine. This happened on both test machines. This is due to the way data is stored while processing. Python and R both use RAM to store the data in a work space while processing. SAS, in comparison, uses the Hard Drive

(HD) for work space data storage. This typically results in slower processing but the ability to handle more data on a typical machine.

6.1 Overall Time Performance

The overall processing time is a key consideration when selecting an analytical tool. The processing time analysis was performed using wall clock time on all three (3) tools with both experiments. The times from both machines were averaged for a final processing time.

No specific tool had a clear advantage or disadvantage (Fig. 2). While SAS offered the best performance when working with the large dataset in the mortality analysis, it was not the fastest performance when doing the accident analysis. The deviation between the time performance was larger with the accident data where many graphics were rendered rather than the mortality data which had a large dataset.

As mentioned above, the experiment could not be completed in Python or R for the full dataset. SAS was the only tool that was able to process the full set of mortality data for 2010 – 2016 combining all U.S. state and territory files for analysis, taking an average of 52 minutes to process on the given hardware. R threw errors once a memory vector was exceeded and Python overloaded the RAM to the point that the machine rebooted itself. The errors were due to the difference in data storage handling.

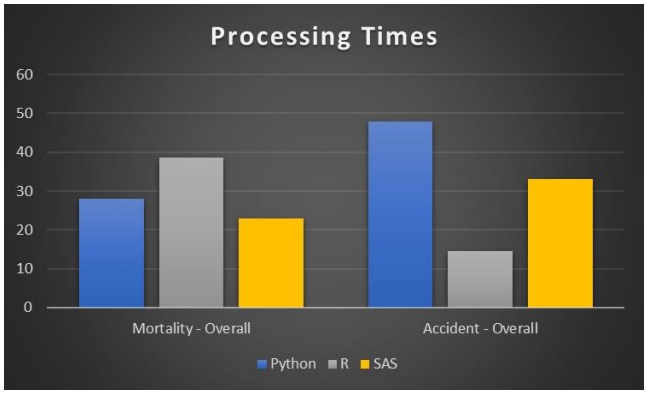

Fig. 2. Comparison of processing time (*wall clock time*).

6.2 Mortality Code

The code for the mortality data experiment was analyzed. Three key areas were identified: loading the data, creating the variable flags, and producing frequency tables. The lines of code for each activity was tallied and graphed (Fig 3). Python and R both required user defined functions to create the variable flags which SAS had an integrated procedure.

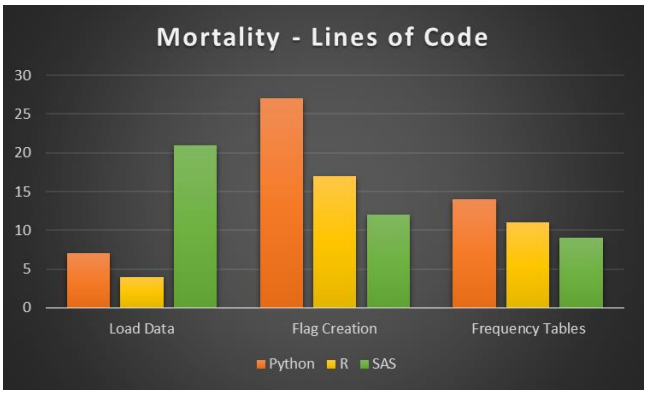

Fig. 3. Comparison of code lines for Mortality Analysis experiment.

In addition to line analysis, the overall activities of the code were measured. A summary table provide the details (Table 2). While the lines of code for specific operations were more efficient for SAS, the overall code requirements to complete the entire analysis were more extensive in both code lines and words. Finally, Python utilized four (4) packages to more efficiently complete the task while R did not require packages and SAs does not have any available.

Table 3. Data from the mortality analysis experiment.

Measured Activity	Python	R	SAS
Processing Time (seconds)	25.9	38.5	22.0
Lines of Code	162	214	296
Total Words	1147	1063	1329
Packages Used	4	0	0

6.3 Accident Fatalities Analysis

The code for the accident fatalities data experiment was analyzed. Several key areas of the code were identified to illustrate the capabilities of the tools. The comparison of the lines of code were tallied and graphed (Fig. 3). Most activities required a comparable amount of code to complete the tasks. However, the results of the heatmap, dummy variable creation, and linear regression provided results showing deviations between the tools.

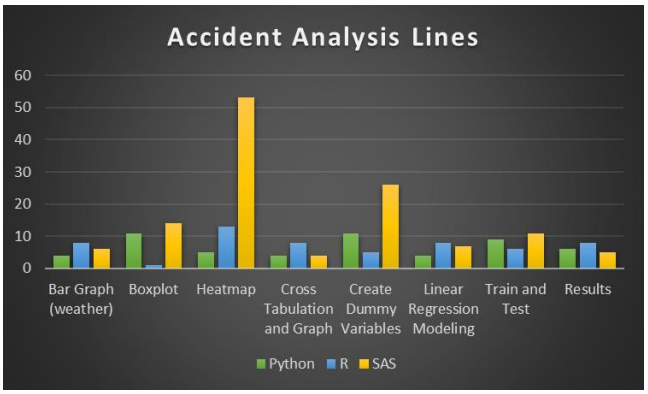

Fig. 3. Comparison of code lines for Accident Fatality Analysis experiment.

The correlation matrix and heatmap required additional lines of programming in SAS compared to both Python and R. However, SAS is in the process of a new release which will include a procedure to create a correlation matrix.

The creation of dummy variables is necessary when working with categorical variables and linear regression. Both Python and R have packages developed in the user community to create the dummy variables. In SAS the creation of a dummy variable is performed using the data step because a more automated method could not be found.

The lines of code for the linear regression were similar for all tools; however, the processing time showed deviations. While Python and R processed the linear regression in 1.15 seconds and 1.27 second respectively, SAS required 15.75 seconds to develop the same model.

A summary table of the analysis of the code from the experiment was compiled. Again, the overall line and word count of SAS exceeds both Python and R. In the accident experiment, Python used 12 packages while R used 7. Many of the packages allowed for reduced code due to functions within the packages.

Table 4. Data from the accident fatality analysis experiment.

Measured Activity	Python	R	SAS
Processing Time (seconds)	39.5	21.3	33.5
Lines of Code	103	88	197
Total Words	446	379	806
Packages Used	12	7	0

During the linear regression modeling, some variations were identified during the selection of the statistically significant variables. However, the final statistics were the same except a possible rounding variation with the RSME (Table 4). Also, SAS did not have the MAE available through the summary statistics for linear regression. The manual calculation was omitted from the experiment.

Table 5. Results from linear regression.

Goodness of Fit Measurements	Python	R	SAS
MAE (Mean Absolute Error)	.17	.17	N/A
MSE (Mean Squared Error)	.14	.14	.14
RMSE (Root Mean Squared Error)	.38	.38	.37
Variance (R2)	.02	.02	.02

Despite the variations in the tools, the experiments were able to be duplicated in all tools including graphics. Below are the correlation heatmaps. Although each has a slightly different appearance, all three provide the same insight to the correlation of variables.

Each tool involves a different level of coding. In Python, a correlation matrix can be calculated with a single line of built-in code then passed to the library Matplotlib to create a figure that can be displayed as a heatmap using the library Seaborn. R also uses a built-in command to generate the correlation matrix then uses a package called reshape2 to manipulate the data and pass the data to the package ggplot to create the graphical output. SAS has plans to include output for a correlation heatmap into PROC CORR in a future release based on an article on SAS Blogs [25]. In the meantime, the blog shares code that can handle this process with a few inputs. The steps include: generate the correlation matrix using PROC CORR, rearrange the data using the Data step, define the graph "shell" then render the graphic using PROC SGRENDER.

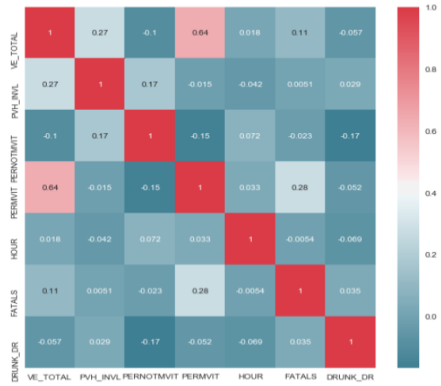

Fig. 4. Correlation heatmap from Python

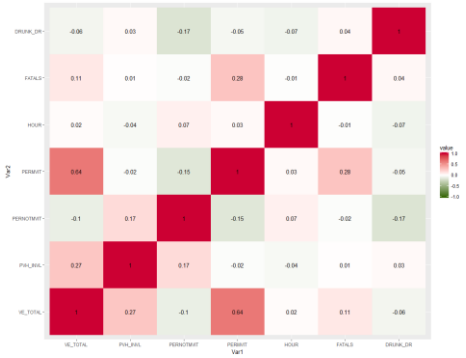

Fig. 5. Correlation heatmap from R

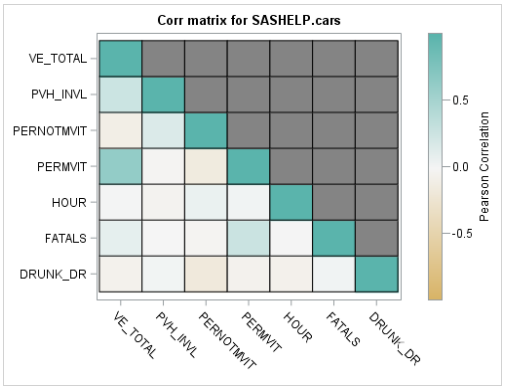

Fig. 6. Correlation heatmap from SAS

6.4 Halstead metrics

The results of the Halstead metrics showed SAS with the lowest vocabulary, length, calculated program length, and volume for both experiments. SAS also had the lowest effort and time with the accident project. With the mortality project, Python had the lowest effort and time scores. On both projects, Python and R were the lowest difficulty. The difficulty scores of these tools were comparable on both projects while SAS was considerably higher.

Halstead Metrics	Accident Data			Mortality		
	Python	R	SAS	Python	R	SAS
Program Vocabulary	223	211	95	422	306	303
Program Length	730	725	448	1,415	1,272	1,062
Calculated Program Length	1,544	1,446	530	3,520	2,385	2,198
Volume	5,695	5,598	2,943	12,340	10,503	8,754
Difficulty	58	56	65	20	27	149
Effort	332,757	313,010	192,276	241,663	288,273	1,300,851
Time	18,486	17,389	10,682	13,426	16,015	72,269

Fig. 7. Halstead Metrics

6.5 Qualitative

The table below summarizes some of Huber's requirements of a statistical and data analysis tool. The first six relate to general aspects of the languages, whereas the last three correspond specifically to data analysis.

According to Huber, a tool dedicated to data and statistical analysis should be extensible. R and Python take advantage of the library of packages they sport to extend their capabilities beyond their base language. Having users contributing in the creation of packages is what keeps these tools in constant evolution. However, caveats include rapid deprecation of packages, and lack of quality control in the development and deployment processes of the package creation. SAS has these extensions built-in in the base code avoiding the need to install and run an external application.

Data handling capabilities in the case of this experiment are measured by the type of computational devices employed to perform this experiment, so in this context Python and R's capabilities are limited by the amount of RAM. SAS is not constrained by this feature as it runs directly on the hard drive.

The three tools provide both on-line assistance to the user whenever an error occurs, indicating the type of error and the exact location of the error in the code. Some packages in Python due to their design even provide alternatives to recover from an error. In this experiment, during the data reading of the mortality files in csv in Python, an alert due to the overpassing of the default memory limit was issued along with a way to fix the error by changing the `low_memory` option from `True` to `False`.

Python, R and SAS can handle numbers, characters, logical, complex, and arbitrary data types. All of them can be operated from either the command line or through one of their multiple IDEs.

All three tools offer differ options to handle missing values. Python's package fancyimpute includes methods such as SimpleFill, MICE (Multivariate Imputation by Chained Equations) and SoftImpute; R offers the MICE, Amelia, Hmisc, mi libraries, while SAS deals with this through the procs MI and MIANALYZE. All three tools have methods that allow for the transformation or replacement of missing values by a test statistic or specific value as well as the dropping of rows or columns that contain unknown values in them.

The three tools can compute linear algebra functions, generate random numbers, do probabilistic distributions, and have high-level customizable visualization packages in the case of Python and R, while SAS has this feature integrated.

Table 6. Qualitative attributes.

Attribute	Python	R	SAS
Packages Available	133,915	10,000	Integrated
Data handling	RAM	RAM	Hard Drive
On-line Error	Yes	Yes	Yes
Numbers & Text	Yes	Yes	Yes
Interactive & programmed	CLI & IDE	CLI & IDE	CLI & IDE
Complex data structures	Yes	Yes	Yes
Missing Values	Yes	Yes	Yes
Linear Algebra	Yes	Yes	Yes
Graphics	Yes	Yes	Yes

7 Conclusion and Future Work

The comparative analysis does not identify a single best tool for all circumstances. The experiments did show situations where each tool performed better than the others with strengths and weaknesses for various activities. All three tools completed the analysis for the experiments; however, some coding required less eloquent brute force methods.

None of the tools stood out as requiring less code overall. Although less code is often considered preferable, it may also make the code less readable and more difficult to understand. Example of this would be positional parameters used in functions.

Python performed well in most respects. The use of a dozen different open source packages demonstrates the strength of the Python community in developing enhancements to the Python capabilities. Although not explored, Python being a general-purpose tool encourages participation from users outside the Data Science community which enhances package availability.

In the experiments, R had the highest time performance on the smaller dataset which can be attributed to the holding of the data in RAM. R also has a large user community providing packages as demonstrated in the experiments. Many users in the scientific and academic community continue to migrate to the use of R.

SAS the clear preferred tool for handling large datasets with moderate computer resources. Since R and Python hold the data in RAM, they were not able to analyze the full mortality data. SAS uses hard drive space to hold the data and process. While generally slower due to the access speed of a hard drive compared to RAM; most PCs have ample hard drive storage and significantly less RAM. Python and R do have methods available to utilize HD space for processing data storage; but that was outside the scope of this paper, which was intended to keep the coding to a typical or standard coding methods.

Depending on the tool chosen a user could build a machine more scaled to the tool. For instance, if Python or R were the primary tool chosen, a machine could be built with a very large RAM capacity to facilitate the use of large data. SAS on the other

hand can benefit from the use of a Solid State Drive (SSD) allocated for the work space. A SSD has a read-write speed that is several times faster than a traditional magnetic HD.

The Halstead metrics is a quantifiable result; however, the values may be subjective. The disparity of programming experience may influence the results. In the experiments, the SAS code was written by a veteran programmer while the Python and R were written by less experienced individuals. Further validation of the metrics could be done by each researcher performing the coding in each of the languages. The metrics would then be mitigated by the diversity of experience.

A good Data Scientist should not focus on being an expert in any single tool. Rather the focus should be on learning the strengths and weaknesses of all tools. With the many resources available for coding expertise, the more important decision is choosing the right tool for the analysis.

References

[1] G. Piatetsky, "Four main languages for Analytics, Data Mining, Data Science," 18 August 2014. [Online]. Available: https://www.kdnuggets.com/2014/08/four-main-languages-analytics-data-mining-data-science.html. [Accessed 10 November 2017].

[2] Burtch Works, "Burtch Works," Burtch Works, 19 June 2017. [Online]. Available: https://www.burtchworks.com/2017/06/19/2017-sas-r-python-flash-survey-results/. [Accessed 11 November 2018].

[3] T. O'Reilly, "O'Reilly Media," 30 September 2005. [Online]. Available: http://www.oreilly.com/pub/a/web/archive/what-is-web20.html. [Accessed 12 March 2018].

[4] CrowdFlower, "2017 Data Scientist Report," CrowdFlower, San Francisco, 2017.

[5] L. M. Ottenstein, V. B. Schneider and M. H. Halstead, "Predicting the Number of Bugs Expected in a Program Module," Purdue University, West Lafayette, 1976.

[6] N. Bulnut, M. H. Halstead and R. Bayer, "Experimental Validation of a Structural Property of Fortran Algorithms," Purdue University, West Lafayette (Ankara, Munich), 1974.

[7] N. Bulnut and M. H. Halstead, "Impurities Found in Algorithm Implementations," Purdue Univiersity, West Lafayette, 1974.

[8] T. J. McCabe, "A Complexity Measure," IEEE Transactions on Software Engineering, Fort Meade, 1976.

[9] A. Measday, "npath - C Source Complexity Measures," GEONius, 25 June 2016. [Online]. Available: http://www.geonius.com/software/tools/npath.html.

[10] G. Jay, J. E. Hale, R. K. Smith, D. Hale, N. A. Kraft and C. Ward, "Cyclomatic Complexity and Lines of Code: Empirical," Scientific Research, Tuscaloosa, 2009.

[11] J. K. Millman and M. Aivazis, "Python for Scientists and Engineers," *Computing in Science & Engineering,* vol. 13, no. 12, pp. 9-12, 2011.

[12] W. McKinney, "Chapter 1- Preliminaries," in *Python for Data Analysis*, Sebastopol, O'Reilly Media, 2013, p. 3.

[13] P. Barlas, I. Lanning and C. Heavey, "A survey of open source data science tools," *International Journal of Intelligent Computing and Cybernetics Information ,* vol. 8, no. 3, pp. 243-245, May-June 2015.

[14] P. H. Vasconcellos, "Top 5 Python IDEs For Data Science," 22 June 2017. [Online]. Available: https://www.datacamp.com/community/tutorials/data-science-python-ide. [Accessed 11 November 2017].

[15] F. Perez, B. E. Granger and J. D. Hunter, "Python: An Ecosystem for Scientific Computing," *Computing in Science & Engineering,* vol. 13, no. 2, pp. 13-21, 2011.

[16] "What is GNU?," 29 09 2017. [Online]. Available: ttp://www.gnu.org/. [Accessed November 2017].

[17] R. A. Becker, "R: A brief history of S.," AT&T Laboratories, New Jersey.

[18] C. A. Gomez Grajales, "Created by statisticians for statisticians: How R took the world of statistics by storm," 19 November 2015. [Online]. Available: http://www.statisticsviews.com/details/feature/8585391/Created-by-statisticians-for-statisticians-How-R-took-the-world-of-statistics-by.html. [Accessed 11 November 2017].

[19] D. Smith, "CRAN now has 10,000 R packages. Here's how to find the ones you need," 27 January 2017. [Online]. Available: http://blog.revolutionanalytics.com/2017/01/cran-10000.html. [Accessed 31 October 2017].

[20] "R GUIs you frequently use," April 2011. [Online]. Available: https://www.kdnuggets.com/polls/2011/r-gui-used.html.. [Accessed 31 October 2017].

[21] K. A. Renzulli, C. Weisser and M. Leonhardt, "The 21 Most Valuable Career Skills Now," *Time.com,* 16 May 2016.

[22] Communications of the ACM, "ACM code of ethics and professional conduct," May 1992. [Online]. Available: http://link.galegroup.com/apps/doc/A12219969/ITOF?u=txshracd2548&sid=ITOF&xid=ec3eb919. [Accessed 15 March 2018].

[23] P. T. Huber, "Languages for Statistics and Data Analysis," *Journal of Computational and Graphical Statistics,,* vol. 9, no. 3, pp. 600-620, 2000.

[24] Hemedinger, "How to build a correlations matrix heat map with SAS," 2013. [Online]. Available: https://blogs.sas.com/content/sasdummy/2013/06/12/correlations-matrix-heatmap-with-sas/ .

[25] B. Baesens, "Commercial Versus Open Source Software for Analytics," *BigData Quarterly,* Summer 2016.