Data Science Rosetta Stone

A Tutorial of and Translation between Data Science Programming Languages

Elaine Kearney
Advisor: Stanley Ahalt, Ph.D.

Honors Thesis Committee
Stanley Ahalt, Ph.D.
Gary Bishop, Ph.D.
Diane Pozefsky, Ph.D.

March 28, 2018

Approved:

Stanley C. Ahalt
Stanley Ahalt, Ph.D., Thesis Advisor

Gary Bishop, Ph.D., Reader

Abstract

Data Science and Machine Learning enable researchers both in academic and industrial fields to analyze data and consequently make important decisions as well as predictions. Different programming languages are used in the pursuit of Data Science. However, it is difficult for researchers to switch between languages or learn a new language which is needed in working across interdisciplinary fields and when working with colleagues from diverse backgrounds. This paper discusses a resource created to demonstrate the commonalities between four different programming languages commonly used in Data Science: MATLAB, Python, R, and SAS. The work for this project came out of research completed both at the Australia New Zealand Banking Group (ANZ) headquarters in Melbourne, Australia, and at the University of North Carolina at Chapel Hill (UNC-CH). This research resulted in an online set of tutorials, which we refer to as the Data Science Rosetta Stone, which demonstrates common data science tasks in the same progression for each programming language. We demonstrate that all of these languages achieve similar results, though with different syntax and/or simplicity of code, and efficiency of execution.

Table of Contents

1 INTRODUCTION 3

2 DEFINING DATA SCIENCE 4

3 MACHINE LEARNING 5

4 IMPORTANCE OF DATA SCIENCE 7

5 PRACTICING DATA SCIENCE 8

5.1 MATLAB 9
5.2 PYTHON 10
5.3 R 10
5.4 SAS 11

6 DATA SCIENCE ROSETTA STONE 11

6.1 MOTIVATION 11
6.2 INTRODUCTION 12
6.3 GOALS 13
A) Main Goals 13
B) Supplementary Goals 13
6.4 OUTLINE 14
6.5 DISCUSSION 15
1. Reading in Data and Basic Statistical Functions 15
2. Basic Graphing and Plotting Functions 15
3. Basic Data Wrangling and Manipulation 16
4. More Advanced Data Wrangling 17
5. Preparation & Basic Regression 17
6. Supervised Machine Learning 17
7. Unsupervised Machine Learning 19
8. Forecasting 19
9. Model Evaluation & Selection 19
6.6 APPLICATIONS 20

7 CONCLUSION & EVALUATION 20

REFERENCES 22

APPENDICES 24

A: MATLAB REFERENCES AND RESOURCES 24
B: PYTHON REFERENCES AND RESOURCES 24
C: R REFERENCES AND RESOURCES 25
D: SAS REFERENCES AND RESOURCES 26

1 Introduction

In March of 1962, John Tukey published his paper, *The Future of Data Analysis* in which he argues for a "reformation of academic statistics" to support the field of data analysis in a more direct way (Donoho, 2015). In this paper, Tukey defines "data analysis" to be:

> *…procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easier, more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data.*
>
> (Tukey, 1962)

Since this time, many other researchers have called for a reformation of statistics to expand beyond the theoretical and enter into a new field, or science, of statistics. In his 2001 paper "Statistical Modeling: The Two Cultures," Leo Breiman delineates the two cultures who use statistical modeling to draw conclusions from data:

> *One assumes that the data are generated by a given stochastic data model. The other uses algorithmic models and treats the data mechanism as unknown.*
>
> (Breiman, 2001)

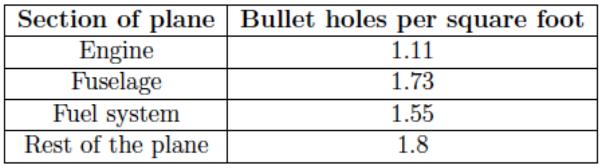
Breiman goes on to argue that the tactic of the statistical community to focus on the use of data models has led to

> *…irrelevant theory, questionable conclusions, and has kept statisticians from working on a large range of interesting current problems.*
>
> (Breiman, 2001)

This isn't to criticize the statistical community, but rather to demonstrate the importance of considering real world problems without seeking to make the data fit the model, but rather altering the tools, techniques, and models to fit the data.

But, just as this new "science of data" field isn't just about the models and assumptions, it also isn't just about the data. Consider the famous example from World War II (WWII), which Jordan Ellenberg explains in his book, "How Not to Be Wrong: The Power of Mathematical Thinking." Ellenberg tells the story of Abraham Wald, a professor of statistics at Columbia University who made recommendations to the Allied forces during WWII.

In one particular situation, the military were faced with the issue of armoring the planes. This issue stems from the tradeoff between armoring, which can cause the plane to be too heavy and use more fuel, and not armoring, which can result in loss of life and loss of planes. The military wanted the Statistical Research Group (SRG), with whom Wald was associated, to investigate data about the distribution of bullet holes in planes so they could suggest a better method of armoring the planes. *Table 1* provides the data supplied by the military.



Section of plane	Bullet holes per square foot
Engine	1.11
Fuselage	1.73
Fuel system	1.55
Rest of the plane	1.8

Table 1: Data concerning the distribution of bullet holes on the planes of the Allied forces in WWII.
(Ellenberg, 2014)

It doesn't take a statistician long to examine the data in *Table 1* and make a suggestion about where to armor the planes. Odds are one will come to the same conclusion that the military came to and proposed to Wald for his approval. The military suggested that they armor the sections of the planes with the highest number of bullet holes per square foot, notably "Rest of the plane" and "Fuselage," but not "Engine." However, this was the exact opposite of Wald's conclusion. Wald asked the military about the *missing* holes, which he knew were on the *missing* planes, or in other words, the planes the military couldn't analyze for data because they didn't return to base after being hit. These were the planes that were hit in the "Engine" section. Therefore, Wald suggested armoring the "Engine" section. Ellenberg compares this phenomenon to that of soldiers in a hospital:

> *…you'll see a lot more people with bullet holes in their legs than people with bullet holes in their chests. But that's not because people don't get shot in the chest; it's because the people who get shot in the chest don't recover.*
>
> (Ellenberg, 2014)

This example demonstrates a critical component of the "science of data," which is the importance of using not only the data, but also a combination of common sense and discipline-specific knowledge to draw a conclusion.

Let us now delve deeper into this "science of data" field, and examine, more specifically, how it is different from statistics, computer science, information technology, and other similar fields.

2 Defining Data Science

In 1974, Peter Naur used the term "data science" multiple times in his book, Concise Survey of Computer Methods, defining it as the "science of dealing with data" (Press, 2013). In his 2001 paper, William S. Cleveland laid out six technical areas in which statistics departments should focus on in order to improve data analysis, and used "data science" in his title (Cleveland, 2001). Cleveland stressed that data analysis (data science), should focus on multidisciplinary approaches.

The Data Science Foundation defines "data science" to be:

> *Scientific study of the creation, validation and transformation of data to create meaning.*
>
> (Data Science Association, n.d.)

And, similarly, the association defines a “data scientist” to be:

> *A professional who uses scientific methods to liberate and create meaning from raw data…has a solid foundation in machine learning, algorithms, modeling, statistics, analytics, math and strong business acumen, coupled with the ability to communicate findings…*
>
> (Data Science Association, n.d.)

Moreover, David Donoho defines the new, “Greater Data Science” in his 2015 paper, *50 years of Data Science*, as the following 6 divisions:

1. *Data Exploration and Preparation*
2. *Data Representation and Transformation*
3. *Computing with Data*
4. *Data Modeling*
5. *Data Visualization and Presentation*
6. *Science about Data Science*

(Donoho, 2015)

While each one of these 6 divisions are uniquely important to data scientists, Donoho goes on to explain that in reality, “Data Exploration and Preparation” could require up to 80% of effort devoted to data science, while the representation of data science in academic settings is dominated by “Data Modeling” (Donoho, 2015).

The combination of these 6 divisions demonstrates the multidisciplinary nature of data science, which combines computer science (“Data Representation and Transformation” & “Computing with Data”) with statistics (“Data Modeling”), as well as with business intelligence and communication (“Data Visualization and Presentation”). The special sauce that makes data science in practice unique is the “Data Exploration and Preparation” aspect (Donoho, 2015).

3 Machine Learning

There is a wide range of definitions of machine learning, just as we have shown the varying definitions of data science. These terms have evolved over time. It is important, though, to try to parse out the difference between data science and machine learning. SAS Institute Inc. (SAS) defines machine learning as

> *…a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that machines should be able to learn and adapt through experience.*
>
> (SAS Institute, 2018)

At first glance it appears that machine learning points to artificial intelligence (AI), after all, this definition does state that “machines should be able to learn and adapt through experience.” Does this mean on their own? Are we living in a world with AI? Not exactly, but we are close. Many describe machine learning as a subfield of AI, such that we give machines data to train a model, and eventually the machine "learns" on its own. On the other hand, AI is the broader idea of a machine being able to carry out tasks on its own. In machine learning, a human is still expected to provide the data to the model, and some parameters on which the model should be "learning" (Marr, 2016). Stated another way, machine learning focuses on an AI problem that can be described in

> *…discrete terms (e.g. out of a particular set of actions, which one is the right one), and given a lot of information about the world, figure out what is the "correct" action, without having the programmer program it in.*
>
> (Granville, 2017)

Therefore, the machine has a lot of guidance about the choices it can make as the "correct" action. A machine coming up with the "correct" action on its own would be a more general case of AI.

One doesn't have to look far to find an example of machine learning in daily life. Consider recommendations on Amazon and Netflix, the self-driving car, fraud detection, computer virus detection, etc. These are all well-known examples of machine learning currently in practice (SAS Institute, 2018).

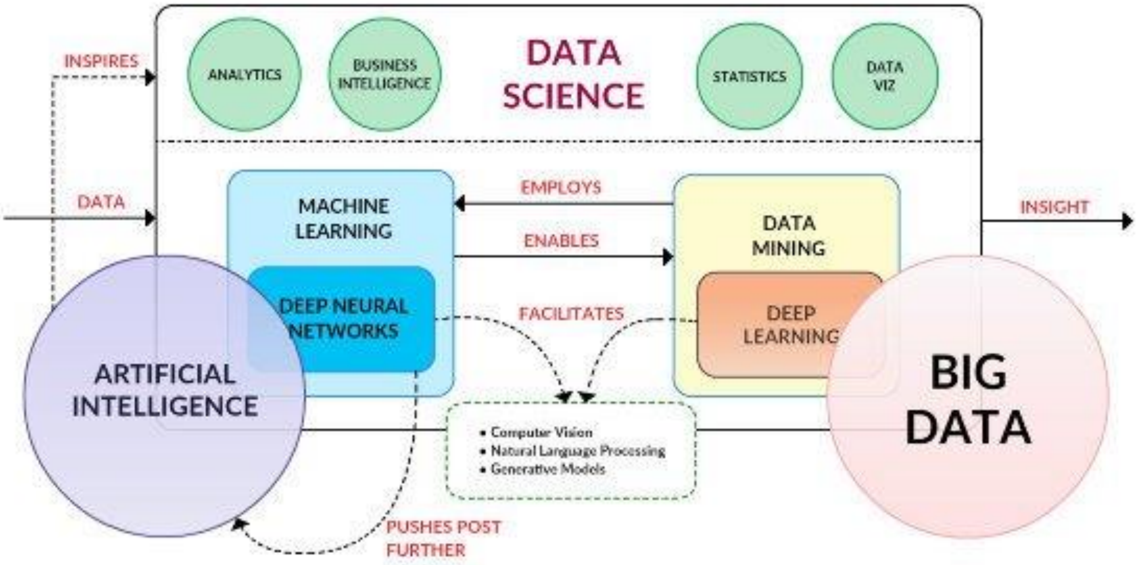


Figure 1: A concept map of the interactions between the fields of Data Science, Machine Learning, Artificial Intelligence, and Big Data.
(Yadav, 2017)

Why discuss machine learning in a paper about data science? Just as machine learning is a subcategory of the general term "AI," machine learning can also be thought of as a subcategory of a more general “data science” term. Particularly, machine learning is a field of algorithms that involve a “machine” (i.e. computer) “learning” (i.e., through an algorithm). Data science, as discussed previously, also encompasses information surrounding data that does not have anything to do with a machine. This includes business intelligence (BI), data visualization, data engineering, and others (Granville, 2017). *Figure 1* demonstrates one approach to explaining the interactions between all of these different terms and concepts, though this is not to suggest this is the “right” way to understand their interactions, but just one approach.

4 Importance of Data Science

Countless examples of success stories demonstrate the importance of data science. Consider the example, given above, of Wald and the WWII airplanes (Ellenberg, 2014). Or the example of the ship captain in 1601 who placed the crew on one of his four ships on a regimen of three teaspoons of lemon juice a day. He found that all of his crew on that ship survived, while almost 40% of the crews on the other ships died. This ship captain’s small experiment and collection of data paved the way for the discovery for the cure for scurvy (Gillam et al., 2009).

Today, the fields of finance and marketing are always leveraging data science. In finance, data science is used to display relevant information about financial resources to stakeholders in a BI framework. Also, data science can be used to predict stock market trends by analyzing data from the past and relevant information about present times, and there is big money in that business. Consider personalized ads on Google, Facebook, and Amazon, for example. These are the result of organizations using user data that is likely provided subconsciously (browsing history, location data, and other metadata) and data science to produce a targeted advertisement (Livingstone, 2017).

Organizations have been formed around and focused on the “#data4good” (or Data-for-Good) movement. One such organization, DataKind, matches top data scientists with social change organizations centered around the notion that there is potential for change, and “close collaboration between data science and social sector experts” is the way to achieve this change (DataKind, n.d.). Projects that DataKind has been a part of include forecasting water demand in California and using satellite imagery to find remote villages in need of assistance (DataKind., n.d.).

Another example of a socially-motivated success-story of data science is a partnership that began between the White House Police Data Initiative and researchers at the University of Chicago and led to a collaboration between the Charlotte-Mecklenburg Police Department and the Data Science for Social Good Summer Fellowship. This project works to take historical data about police officers and apply a machine learning model to the data to assign a “risk score” to each police officer so that departments can implement interventions to prevent adverse events, such as, in extreme cases, police brutality. Ultimately, the project has been very successful, and has even resulted in the founding of Flag Analytics, a company that will commercialize the research so that it can be used by police departments across the country (Mitchum, 2016).

An example more specific to North Carolina is the Coastal Emergency Risks Assessment (CERA) model which uses data science to predict storm surges. This model and website, sponsored in part by the Renaissance Computing Institute (RENCI), are particularly helpful during hurricane season and inform citizens about their risk during storms.

On another note, data science isn't always inherently "good," and doesn't always save lives outright. Consider the somewhat-humorous, somewhat-unnerving story of Target's statistician, Andrew Pole, discovering a "teen girl was pregnant before her father did" (Hill, 2012). Pole had assigned Target shoppers a "pregnancy prediction" score so that the company could send coupons to potential mothers for pregnancy supplies, scheduled according to the predicted timing of the pregnancy (Hill, 2012) (Duhigg, 2012).

Given that data science can be used to solve a wide variety or problems in nearly all disciplines, the job market for data science is steadily growing. In October of 2012, Harvard Business Review deemed "data scientist" as the "Sexiest Job the 21st Century" (HBR, 2012). According to GlassDoor, the average base salary for a data scientist in 2018 is just over $120K (GlassDoor, 2018).

5 Practicing Data Science

A common question among aspiring data scientist is: "What programming language should I learn to do data science?" A slightly more nuanced version of this question, "What programming language should I learn to get a machine learning or data science job?", is what KD Nuggets calls the "silver bullet question" (KD Nuggets 2017). There's not an answer to this question that will cover all situations, because invariably different companies use different programming languages. This is for several reasons.

For one, companies often re-use legacy code that was written in a language that may not be as popular now or is only industry-specific. Moreover, since data science is such a new field, most practicing data scientists have degrees in other fields other than specifically data science. These other fields include computer science, business administration, statistics, mathematics, and physics (Swanstrom 2015). Each of these fields has a preference towards one programming language over others, and data scientists work with individuals from all these different fields, and more. So, the question still remains: "What language should I learn?"

First, to clarify, a data scientist generally isn't expected to know all the programming languages and produce the best data visualizations but is expected to adapt quickly. Therefore, it is our belief that just knowing one programming language, with a general understanding of other languages, is sufficient to do data science, and get a job in data science. Sure, there are some organizations that will state in a job description that programming experience in Programming Language A is required. But, as professor of computer science at UNC-Chapel Hill, Dr. Gary Bishop, says,

> *The answer to the question, "Can you program in ______?" is "Yes! (I may need a manual but I know how to program and programming is programming until you get to really weird stuff like Prolog and such)." Mastering the idioms for a language is important but certainly affects time to write code and code quality, but*

> *they are generally easy to pick up by reading code.*

(Bishop 2017)

This is not to suggest ignoring programming language requirements on a job description, but instead to provide an alternative view of "knowing" and "learning" how to program.

In a 2014 KD Nuggets Poll, 49% of responders reported using R, compared to 36.4% using SAS, and 35% using Python (see *Figure 2*). The next highest was 30.6% reporting using SQL (which can be used in Python, SAS, and R). In a similar poll, responders were asked about their primary programming language for analytics, data mining, data science tasks. In 2015, 51% reported R as their primary language, compared to Python at 29% and "Other", which included SAS, at 17%.

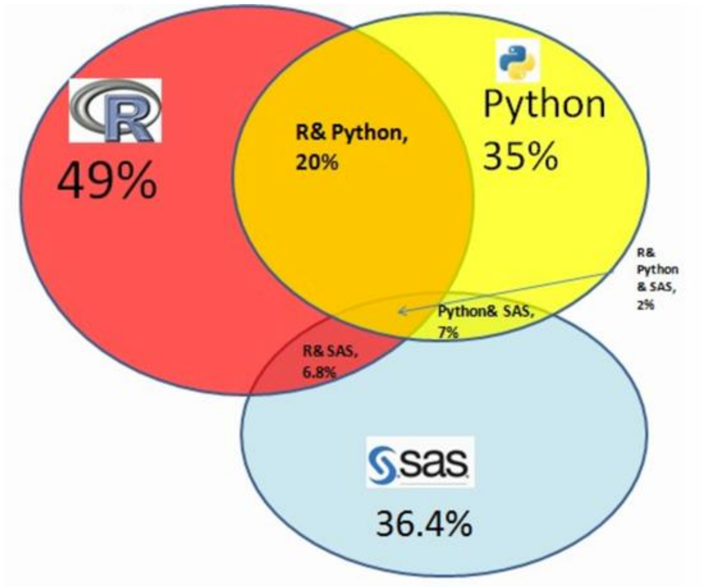


Figure 2: Results from a 2014 KD Nuggets Poll on the use of different programming languages for data science.
*(*Piatetsky, 2014*)*

While there are other programming languages that are of course used by data scientists, varying around the world and throughout different industries, Python, R, and SAS are the main three, and will be discussed throughout the remainder of this paper. MATLAB will also be added to this list as it is often used in academic settings and in science and engineering fields.

5.1 MATLAB

MATLAB is a proprietary software that combines the MATLAB application, which integrates visualization and programming, with the MATLAB scripting language. The development of MATLAB started at the end of the 1970s by Cleve Moler, the chair of the computer science department at the University of New Mexico at the time. The language soon became widely popular among university students, and Jake Little realized the potential of the language. This led to Jack Little, Steve Bangert and

Cleve Moler joining hands to rewrite MATLAB in the C language (Cleverism, n.d.). The MathWorks, which is the organization responsible for the further growth of the MATLAB "technical computing environment" was founded in 1984 in California by J.H. Wilkinson, George Forsythe, and John Todd (Moler, 2004). Today, MATLAB is still very popular in education.

5.2 Python

The programming language, Python, was conceptualized in the late 1980s. At the time, Guido van Rossum was working on a project to develop a distributed operating system, called Amoeba, at Centrum Wiskunde & Informatica. In the early 1980s, van Rossum had worked on a team to develop the programming language ABC, which is a general-purpose programming language and programming environment. Given his experience in developing ABC, van Rossum decided to design a simple scripting language that contained the better properties of ABC without some of its problems, and out of this, Python was created (Klein, n.d.).

Python version 0.9.0 was first published in February of 1991, and 1.0 was released 3 years later. Now, the most commonly used versions are Python 2.0 (released in October of 2000) and Python 3 (released in 2008) (Klein, n.d.). Today, Python is no longer written by van Rossum, but is an open source programming language, which means all the source code is available to the general public for use and/or modification, and the code is improved through a collaborative effort. Python can be run in the command line by simply typing

```
>>> python
```

or, as many prefer to exploit the interactive nature of Python, you can use Anaconda-Navigator to access other applications, such as Jupyter Notebook and Spyder for running and writing Python code.

5.3 R

The statistical programming language, R, had its binaries first released on the *s-news* mailing list in 1993 (Ihaka, 1998). Originally, R was developed by Robert Gentleman and Ross Ihaka at the University of Auckland as an implementation of the S programming language. R is so-named after the first names of the first two authors of R, and as a play on the name of the S programming language (Hornik, 2017). After receiving some feedback about the first binaries, Gentleman and Ihaka decided to make the source code for R available under the terms of the Free Software Foundation's GNU general license (Ihaka, 1998).

Since 1997, there has been a group of developers, named the "R Core Team," who can modify the R source code (Hornik, 2017). R users can also develop their own R packages and release them for use by others. One such package that is used often in R is the knitr package that builds on the Sweave system within in R to produce reproducible reports using LaTeX (RStudio Support, 2017).

In fact, knitr was used to create the tutorials discussed further here, Data Science Rosetta Stone, by knitting the code to an html format. R is mostly used for statistical computations and graphics, and many users choose to use R Studio, a graphical user interface, to write R code (Hornik, 2017).

5.4 SAS

Unlike Python and R, which are open source software programming languages, SAS is proprietary software, like MATLAB. SAS has its origins in analyzing agricultural data, starting through a research project at North Carolina State University. In 1976, the SAS company was founded and its name originally stood for "Statistical Analysis System," though now the name stands on its own. Headquartered in Cary, North Carolina, SAS employs more than 14,000 across the world, and has customers in 149 countries. SAS now assists customers in all types of industries, "from pharmaceutical companies and banks to academic and governmental entities" (SAS Institute, 2018). Jim Goodnight, one of SAS's co-founders, is still the CEO of SAS.

An interesting characteristic of SAS, especially within the United States, is the monopoly it possesses on clinical trial data formatting, as the U.S. Food & Drug Administration (FDA) archives files in SAS transport format and expects that clinical trial statistical reports are run through SAS software (U.S. Food & Drug Administration, 2017).

SAS software includes many different products. The most commonly used product is the Base SAS environment in which users can write and run SAS code.

The remainder of this paper will focus on these four languages and how they can be used in data science related tasks by introducing the Data Science Rosetta Stone tutorials.

6 Data Science Rosetta Stone

6.1 Motivation

The work for this project came out of research completed both at the Australia New Zealand Banking Group (ANZ) headquarters in Melbourne, Australia, and at the University of North Carolina at Chapel Hill (UNC-CH). In 2017 at ANZ headquarters, data scientists were seeking to form an internal group to support employees throughout the company with managing data, analyzing and making predictions on data, and generating data visualizations for presentations and marketing. Essentially, this group's goal was and still is to provide data science as a service to the rest of the organization. What this group found is that employees in different departments were using different programming languages, and that they were having difficulty sharing code to do the same job simply because the code was written in different languages. For example, the statisticians prefer R, the computer scientists Python, and risk managers SAS. In order to address this challenge, the leader of the ANZ data science group suggested the creation of an online set of tutorials to translate between different programming languages, specifically, Python, SAS, and R, for the ANZ employees. A by-product of these tutorials would be resources for learning a brandnew programming language, including the basic tasks of data science. Through a summer intern project, this set of tutorials was created, and a version was published online, internally at ANZ. As a continuation of this research at UNC-CH, MATLAB was added, and the tutorials were published externally, as will be explained further in this paper.

6.2 Introduction

In 1799, French army engineers of Napoleon Bonaparte's Egypt campaign discovered a stone slab while making repairs to a fort near the town of Rashid (Rosetta). The texts on the stone included writings in hieroglyphics, ancient Egyptian demotic, and ancient Greek. Ultimately this stone led scholars to deciphering and understanding hieroglyphics and the history of Egypt (Nix, 2015). Today, "Rosetta Stone" is most widely known as "the best way to learn a foreign language," through the loads of resources this company offers to help you learn a new language by translating between a language you know and a new language (Rosetta Stone, n.d.).

Given all that has been discussed prior in this thesis, including the importance of data science, and the importance of knowing how to program, and the wide variety of programming languages used, we sought to create a resource that would make it easy to translate between the popular data science programming languages. We chose to name this resource "Data Science Rosetta Stone," since, similar to the Rosetta Stone company, this would provide translations and resources to learn new languages, but instead of *human* languages, *programming* languages. And it turns out, this idea, and even more interestingly, this name, are not unique. A quick Google search of "Data Science Rosetta Stone" reveals several different resources focused on this topic, including our own Data Science Rosetta Stone tutorials.

The first is Heaton Research's "Data Science Rosetta Stone: Classification in Python, R, MATLAB, SAS & Julia." In this resource on his website, Jeff Heaton walks users through the Titanic Classification Problem, a commonly-used data science springboard project available from Kaggle, through these 5 languages using a logistic regression model. Some strengths of this resource are that it covers 5 programming languages, and it demonstrates a data science project from start to finish (that is, with a cleaned dataset to start with). Moreover, it gives a lot of information about the code and models used and compares the different languages as it goes along. For instance, Heaton explains that R "win[s] the award for the shortest program in this article" (Heaton, 2017). Limitations of this resource include that it only covers one data science problem and modeling technique, and the examples are not easily searchable. This will be discussed more in the goals of our own Data Science Rosetta Stone.

On another webpage with the title of "Data Visualization – The Rosetta Stone of Data Science," Mithun Sridharan argues that due to the fact that analyses and models are becoming more and more complex, and therefore more challenging for the end-users to understand, "visualization is evolving into the 'Rosetta Stone' of Data Science" (Sridharan, 2015). This is a very interesting take on the use of visualization, and we do agree that data visualization can help communicate research more clearly to end-users. While our own Data Science Rosetta Stone will not provide examples of the best and most clear visualizations, it does provide some examples (in fact, a whole chapter), on data visualizations, with links to more resources on data visualizations in each language.

In addition, there are several GitHub pages including the terms “data science” and “rosetta stone” in their name, including *data-raccoon*’s “rosettastone-datascience” with common data science operations in Python and R, and *Columbia-applied-data-science*’s “rosetta” with tools for data science in Python, with a focus on text processing.

While all of these other resources demonstrate how in certain circumstances it is possible to translate between two or more data science programming languages, none do it as completely as Data Science Rosetta Stone, as will be shown in the remainder of this paper.

6.3 Goals

When we started this project, we had a couple of main goals in mind, many of which others surely considered when creating their own Rosetta Stone resources, along with several supplementary goals:

A) Main Goals

(1) *Highlight the commonalities between a few different programming languages in completing data science tasks.*

(2) *Make a resource accessible to students, educators, researchers, new & seasoned data scientists, etc., who are interested in either*

   a. *learning a programming language for data science without significant prior knowledge of programming, or*

   b. *learning a programming language for data science, already knowing another language.*

B) Supplementary Goals

(1) *Reproducible & accessible*: We consider this to be extremely important in the education process of a programming language, and also is necessary for the reliability and therefore usability of the tutorials themselves.

(2) *Quick access to short examples of commonly used tools*: It is our goal to provide self-contained examples to demonstrate a single task without requiring understanding of other examples.

(3) *Basic enough for a beginner*: We’ve all been a beginner at some point, possibly trying to figure out how to get started on a code script. It is our intention that these tutorials are basic enough that they are beginner-friendly and explain some of the basic lingo of a programming language and demonstrate how to get started.

(4) *Advanced enough to include some machine learning*: Machine learning is a “hot topic” of data science and statistics currently. It is our goal that these tutorials are advanced enough that they can provide some direction into using machine learning models.

(5) *Links to more resources*.

(6) *Able to be modified*. We see data science as a collaborative field that is constantly evolving with the further development of existing programming languages. We hope that our tutorials improve alongside this field.

6.4 Outline

Data Science Rosetta Stone: *A Tutorial of and Translation between Data Science Programming Languages* (DSRS) begins with a landing page introducing the resource. The format of the resource is essentially four tutorials, one for each programming language, MATLAB, Python, R, and SAS, with examples of common data science tasks organized in the same way across the languages. To the extent possible, all four tutorials follow the same progression and can be compared by chapter and example number. The tutorials also include extensive helpful links to the documentation for the programming languages. All data used in the examples are available at the data.world project workspace.

The following is the structure of each tutorial, and a brief explanation of the contents of each chapter:

(1) *Reading in Data and Basic Statistical Functions*
This chapter includes examples of importing data into each respective programming environment and determining dimensions and basic information about a dataset. The basic idea behind this chapter is that these are the tools needed when first starting a new data science project to learn and manipulate data in small ways.

(2) *Basic Graphing and Plotting Functions*
Continuing with the idea of learning data, Chapter 2 includes examples of data visualizations, including basic histograms, boxplots, scatterplots, and even a plotted line of best fit.

(3) *Basic Data Wrangling and Manipulation*
This chapter includes similar types of tasks as in Chapter 1, but now at a slightly more advanced level. For instance, operations on variables in the dataset to produce a new variable, dropping variables, adding observations, sorting, and creating user-defined functions. This chapter is particularly helpful for data cleaning and feature engineering because it demonstrates common operations on variables in a dataset.

(4) *More Advanced Data Wrangling and Manipulation*
In this even more advanced chapter of data manipulation, examples of handling data with missing information, merging datasets, creating pivot tables, and returning all unique values from a variable are shown. This chapter is the first to incorporate a new dataset in order to have examples for missing data, and it also demonstrates tasks that are useful for data cleaning. This is the last chapter of data manipulation.

(5) *Preparation & Basic Regression*
Chapter 5 is more specific to data preparation for modeling, and even gives some examples of the most basic modeling techniques. This chapter begins with pre-processing data using principal component analysis and then splitting data into training and test data. The chapter ends with two basic models: logistic regression and linear regression.

(6) *Supervised Machine Learning*
This chapter is similar to the final two sections of the previous chapter (logistic and linear regression models), with the addition of training and testing the models. This chapter also includes decision trees, random forest models, gradient boosting models, extreme gradient boosting models, support vector models, and neural networks.

(7) *Unsupervised Machine Learning*
Instead of users specifying the target variable and the input variables to predict the target variable, unsupervised modeling does not specify the target variable, but instead seeks to "cluster" the features. This chapter includes some of the most well-known clustering techniques, including k-means clustering, spectral clustering, and Ward hierarchical clustering.

(8) *Forecasting*
This chapter includes examples of forecasting the trend of time-series data for about 24 months in the future. Two of the models included are ARIMA and Simple Exponential Smoothing. These models take seasonality of trend into account which helps to produce the best prediction and are very useful in finance and marketing.

(9) *Model Evaluation & Selection*
Tools of model evaluation, comparison, and selection are discussed in this chapter, including accuracy metrics.

6.5 Discussion

1. Reading in Data and Basic Statistical Functions

From our research we found that the techniques displayed in this chapter are standard and simple across all programming languages. In fact, most of these tasks only required 1-5 lines of code in MATLAB, Python, and R. In the case of SAS, 1-2 data steps and/or procedures which each include 1-5 lines of code were needed, so relative to typical SAS programs, this is about the same amount of effort as in the case of 1-5 lines of code in MATLAB, Python, and R. The only major differences were with respect to syntax. The results of each task were the same across each language, as the tasks were simply to calculate summary statistics, produce tables, and generally understand the dataset.

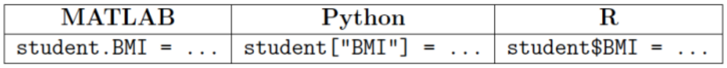
2. Basic Graphing and Plotting Functions

Like in chapter 1, the results of this chapter were standard across the different languages. It is worth noting that Section 2.4, which was the task of visualizing a relationship between two continuous variables by producing a scatterplot and a plotted line of best fit, required the most code out of all of the tasks in this chapter for each programming language. This is due to the fact that the parameters for the line of best fit first must be determined, and then the line of best fit must be plotted with a textbox on the scatterplot to display the equation of the line.

While the graphics of the plots varied, each language includes the tools to produce standard plots and graphs used in data science. In Section 2.6, which was the task of visualizing a continuous variable, grouped by a categorical variable, using side-by-side boxplots, two different methods were demonstrated for Python and R. This was in order to show different packages, and also demonstrate how to add color and a legend to the plot. In Python, the seaborn package (Waskom, 2017) was used since it is very useful for more advanced graphics, compared to the standard matplotlib pyplot package (which provides a MATLAB-like plotting framework) (Hunter et al., 2017). In R, the standard R package's capability was shown, as well as ggplot2 package (Wickham et al., 2016) which added the color to the plot. In SAS, the standard method of producing the side-by-side boxplots automatically colored the plots by groups. The main procedure used in SAS was the SGPLOT procedure (SAS Institute, 2017).

3. Basic Data Wrangling and Manipulation

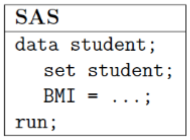
Like the first two chapters, the results of each task were the same across the different languages. Similarities between the languages of MATLAB, Python, and R became evident in this chapter. For example, in all three languages a new variable, or column, in a dataset can be created just by calling it:



MATLAB	Python	R
`student.BMI = ...`	`student["BMI"] = ...`	`student$BMI = ...`

Table 2: Creating a new variable in a dataset in MATLAB, Python, and R.

On the other hand, in SAS, in order to create a new variable in a dataset, a data step must be used to set the current dataset, and create a new dataset with the new variable:



SAS

`data student;`
`   set student;`
`   BMI = ...;`
`run;`

Table 3: Creating a new variable in a dataset in SAS.

For sorting in Section 3.5, it was found that MATLAB, R, and SAS use a stable sorting algorithm by default. A stable sorting algorithm is one in which the relative ordering of two equal values in a vector is maintained through the sorting process. In the case of this task in DSRS, the **student** dataset is being sorted by the **age** variable. Multiple students are age 11. In a stable sorting algorithm, the prior ordering of those students with the same age is maintained through the sorting process. In Python, however, the default algorithm is not a stable sorting algorithm for the `sort_values()` function. Therefore, `kind = 'mergesort'` was specified in Python, since merge sort is a stable sorting algorithm.

In this chapter there were more examples of where the code of the different languages diverged. For example, it was rather simple to create a user-defined function in Python and R, but this was more challenging in SAS (longer code) and MATLAB.

4. More Advanced Data Wrangling

In this fairly short chapter, the results for the first three tasks of dropping missing variables and merging datasets by a common variable and then by observation number were the same across all languages. The pivot table task turned out to be the most challenging of these advanced tasks, though it is commonly requested in programs such as Excel. In Python and R, the number of lines of code to produce the pivot table was very similar. In SAS, however, the SQL procedure, which is a procedure in SAS to write SQL code, was used to produce the pivot table.

In the last task of returning all unique values from a text (character) variable, the code for MATLAB, Python, and R was very similar as it was simply one line of code in each language. In SAS, however, the IML procedure was used for the first time in DSRS for this task. The IML procedure treats datasets as a matrix and allows for indexing into datasets by row and column number. Once the IML procedure environment was invoked in SAS, the task of returning all unique values was also one line of code. The rest of the code was simply to set up the environment and print the results.

5. Preparation & Basic Regression

This chapter represents what can be thought of as the second part of DSRS: shift from data exploration, visualization, and preparation to data modeling. In the first task of this chapter, which is to produce the first four principal components of the popular Iris dataset, the absolute value of the results were similar across all languages. The main differences were that the direction of the principal components differed in some cases, and the results from SAS were the transpose of the results from MATLAB, Python, and R, simply because of the way SAS presents the results compared to the other languages. Also, SAS automatically scales the data before producing principal components, so in MATLAB, Python, and R scaling of the data was needed to produce the same results.

In the next task of producing train and test datasets, the results are not the same as each language has its own pseudorandom number generator, so even if the seed used to start the generation of random numbers is the same, there is no guarantee that the random numbers generated will be the same. The purpose of this task was to demonstrate that train and test datasets can be generated similarly in all languages, as this is useful for training and testing models, as will be demonstrated in subsequent tasks and chapters.

For the rest of the tasks, the results were almost identical across all languages, demonstrating that logistic and linear regression models in all languages are the same, though the code to produce the models is different.

6. Supervised Machine Learning

Continuing on in the second part of DSRS, this chapter demonstrates supervised machine learning models which are those models where both the input and target variables are given to the model in a training phase, and then an optional testing phase is used to predict unknown targets given new input variables.

This chapter includes decision trees, random forest models, gradient boosting models, extreme gradient boosting models, support vector models, and neural networks. Not all of these models are available in all languages, and this was indicated where necessary. It is important to note that the training and test data used in these examples were the same across all the languages (not simply the training and test data output from Section 5.2, which was to split data into training and testing data and export as a .csv file) so that the results could be compared more easily, and these data are available at the data.world repository for this project.

As was seen in the evaluation of Chapter 5, the results of the first two tasks of Chapter 6 (logistic and linear regression), including both model parameters and evaluation statistics, were the same across all languages.

While each of the languages support decision trees and random forest models, the results of these modeling tasks differed across the languages. This is most likely due to the fact that these models have a random start and, as has been explained earlier, the random number generators in each language differ. Moreover, the kinds of output supported by these models in the languages are different. For example, R offers a plot of the decision tree model. In order to have some information to compare across the languages, we output the variable importance in each language (the top 5 important variables for the model), which are similar, though not exactly the same across the languages. This is, again, due to the random starts of the models.

Python, SAS, and R each support gradient boosting, extreme gradient boosting, and neural network models. In SAS, however, the gradient boosting models are available through SAS Enterprise Miner, which is a SAS Graphical User Interface that allows users to build a model network by connecting nodes on a diagram, rather than specifying the model through code. Furthermore, the extreme gradient boosting models in SAS are available through running R code through the SAS IML procedure. In addition to variable importance, we also output some model evaluation, such as confusion matrices or mean square error, where appropriate, and found them to be similar across the different languages.

Support vector models are available in all languages, though support vector regression models are not currently available in SAS. For the support vector models, we did not find any significant differences across the languages.

Overall, we found the Python scikit-learn package to be very useful in producing supervised machine learning models, particularly because the syntax is similar across the different languages. In R, supervised models were slightly more challenging because we needed to use several different packages for the models. In SAS we used both the basic SAS coding environment, R code called from inside the IML procedure, and SAS Enterprise Miner to achieve the different models desired. Nevertheless, for the most part the models were similar and available in all languages, which was our first main research goal for this project.

7. Unsupervised Machine Learning

Finishing up the second part of DSRS, this chapter demonstrates unsupervised machine learning models. These models differ from supervised machine learning models in that the target variable is not specified, but rather only input variables are specified and the machine attempts to separate the data into clusters.

K-means and Ward clustering techniques are available in all the languages considered, while spectral clustering, DBSCAN, and self-organizing map techniques are only available in Python, R, and SAS. In SAS, Spectral clustering and DBSCAN are implemented by calling R code from inside the IML procedure, and self-organizing map is available through SAS Enterprise Miner. The output from these different models is standard since it is simply a matrix of the predicted clusters. Like in the previous chapter, we found the Python scikit-learn package to be very useful for clustering techniques, except for self-organizing map.

These clustering techniques often take a random start which is why the results are not identical across the languages. In our examples, we show a confusion matrix of the actual target variables vs. the predicted clusters. This would not likely be done in practice, since typically clustering is done when there isn't a particular target variable in mind, but it is helpful to see how accurate the clustering techniques are in identifying different groups of the data.

8. Forecasting

In this last chapter of modeling, forecasting models were demonstrated which focus on time-series data which include seasonality, with a goal of predicting months or years in the future, based on previous data. We found the ARIMA model, which is a popular forecasting model, to be available in all languages. In R and SAS, Simple Exponential Smoothing and Holt-Winters models were available. In Python, we demonstrated these two models by defining the function to specify the models. In Python and R, we demonstrated the Facebook Prophet model as a bonus model that these two languages support.

Overall, we found the results to be very similar across the different languages. The plots provided by each language are useful for evaluating the forecast of the series.

9. Model Evaluation & Selection

In this final chapter of DSRS, we demonstrate how to evaluate both regression and classification models. The first two tasks of evaluating using the coefficient of determination and accuracy score were based on the scikit-learn definitions of these terms, which Python provides by default in the scikit-learn package. In the other languages, these methods of scoring were produced through code.

This chapter could be improved in future iterations of this project, possibly through adding examples of cross-validation.

6.6 Applications

We anticipate that the applications of this research and online resource will include greater collaboration on data science projects between professionals with experience in a variety of programming languages, as was the hope when this project was started at ANZ in the summer of 2017. Moreover, we hope that aspiring and practicing data scientists will use Data Science Rosetta Stone as a tool in their data science toolkit to help them learn how to complete some basic data science tasks, translate a project into a new language, or simply to see examples of the correct syntax of code in a language.

7 Conclusion & Evaluation

MATLAB, Python, R, and SAS are the most popular data science programming languages today. While it is generally believed that the capabilities and results of these four languages for common data science tasks are rather similar, the use of these languages varies throughout fields that employ data science. In order to address challenges of researchers switching between these popular languages and colleagues from different programming language backgrounds working together on data science projects, we produced an online set of tutorials, which we call Data Science Rosetta Stone (DSRS).

DSRS walks through common data science tasks simultaneously in four different languages. These tutorials can be used in a variety of ways. We anticipate that DSRS will be used by aspiring and seasoned data scientists learning a new programming language or a new data science technique, seeking quick access to short examples of data science tasks, and searching for links to further resources and documentation for a procedure, function, or short piece of code. With respect to evaluating our success in achieving our outset goals, let us first examine our main goals:

> (1) *Highlight the commonalties between a few different programming languages in completing data science tasks*.
>
> Throughout the research and completion of DSRS, it was found that the functionality of these different languages is often similar, if not exactly the same. It was also found that some languages are *better* than others in certain cases, but one in particular was not found to be superior overall.
>
> (2) *Make a resource accessible to students, educators, researchers, new & seasoned data scientists, etc., who are interested in either*
>
> c. *learning a programming language for data science without significant prior knowledge of programming, or*
>
> d. *learning a programming language for data science, already knowing another language.*
>
> DSRS is currently available publicly at the DataScienceRosettaStone.com webpage, though it was previously only published internally at ANZ. Since it's deployment in October of 2017, DSRS has been accessed by 105 users in 10 different countries, with the

majority (63.8%) in the United States (Google Analytics, 2018). The most popular page accessed by the users, excluding the home page, is the Python tutorial page (Google Analytics, 2018). Organic search and direct referral are the leading channels of access (Google Analytics, 2018).

Therefore, our two main goals have been achieved, though more can be done to increase awareness of DSRS, and its variety of uses. Let us now turn to our supplementary goals:

(a) *Reproducible and accessible*. The data and code used in the examples and tutorials to produce the results given are publicly available.
(b) *Quick access to short examples of commonly used tools*. The tutorials include a "Table of Contents" for quick access to self-contained tutorials. There is more that could be done to address this goal, possibly through more explanation about syntax and techniques used. Moreover, some of the tutorials flow one into the next so they are not all precisely self-contained. For the most part, though, we feel that these tutorials are more self-contained and shorter than other similar tutorials.
(c) *Basic enough for a beginner*. The tutorials begin with reading data into the programming environment, gathering basic statistics about the data, and producing simple plots to understand the data. We assume a beginner has some basic experience coding, so picking up the new syntax is the major hurdle to overcome. Through offering many different examples using different packages and procedures when possible, we feel that we have demonstrated the basic syntax of the different programming languages in a manner that would be sufficient for a beginner.
(d) *Advanced enough to include some machine learning*. Our tutorials cover both supervised and unsupervised machine learning models. However, there is a caveat: *These tutorials do not explain the motivations, assumptions, and use-cases of each particular model used, but rather only demonstrate their use. In accordance with ethical data science, we expect that users will not blindly implement models, but will do more to understand the techniques used before employing the models.*
(e) *Links to more resources.* Below each example in these tutorials, there are links to the documentation of the procedure, function, and/or coding technique used to produce the example. Furthermore, there are Appendices and Indexes of more links to resources.
(f) *Able to be modified.* Currently, the only way for a user to modify the code and results in the tutorials is to contact the authors through the website. To the extent possible, we will be updating, modifying, and adding to the code in these tutorials. In the future we hope it is possible for users to in some way edit the tutorials, or propose improvements, and these changes could be implemented similar to the capabilities of GitHub.

Another goal that we have for the future of DSRS, that we did not outline previously is the ability for a user to have two or more tabs of the tutorials open at a time within one session. Therefore, a user could see Python and R code, for example, side-by-side to compare without having to switch between two different tabs on the browser. Also, we would like to make DSRS a community of data science that fosters collaboration on projects and provides helpful and open discussion concerning data science pedagogy.

References

Breiman, L. (2001). Statistical Modeling: The Two Cultures. *Statistical Science*, 16(3), 199-231.

Cleveland, W.S. (2014). Data Science: An Action Plan for Expanding the Technical Areas of the Field of Statistics. *Statistical Analysis and Data Mining*, 7, 414-417.

Cleverism (n.d.). *MATLAB*. Retrieved from https://www.cleverism.com/skills-and-tools/matlab/.

Data Science Association (n.d.). *About Data Science*. Retrieved from http://www.datascienceassn.org/about-data-science.

DataKind. (n.d.) *About Us*. Retrieved from http://www.datakind.org/about.

DataKind. (n.d.) *Projects*. Retrieved from http://www.datakind.org/projects.

Donoho, D. (2017). 50 years of Data Science. *Journal of Computational and Graphical Statistics*, 26(4), 745-766.

Duhigg, C. (2012). *How Companies Learn Your Secrets*. Retrieved from http://www.nytimes.com/2012/02/19/magazine/shopping-habits.html?pagewanted=1&_r=1&hp).

Ellenberg, J. (2014). *How Not to Be Wrong: The Power of Mathematical Thinking*. New York, NY: The Penguin Press.

Gillam, M., Feied, C., Handler, J., Shneiderman, B., Plaisant, C., Smith, M., Dickason, J. (2009). The Healthcare Singularity and the Age of Semantic Medicine. In T. Hey, S. Tansley, & K. Tolle (Eds.), *The Fourth Paradigm: Data-Intensive Scientific Discovery* (pp. 55-108). Redmond, Washington: Microsoft Research.

Google Analytics. *Data Science Rosetta Stone*. Retrieved on March 13, 2018.

Granville, V. (2017). *Difference between Machine Learning, Data Science, AI, Deep Learning, and Statistics*. Retrieved from https://www.datasciencecentral.com/profiles/blogs/difference-between-machine-learning-data-science-ai-deep-learning.

Heaton, J. (2017). *Data Science Rosetta Stone: Classification in Python, R, MATLAB, SAS, & Julia*. Retrieved from http://www.heatonresearch.com/2017/08/17/ds_rosetta_stone.html.

Hill, K. (2012). *How Target Figured Out A Teen Girl Was Pregnant Before Her Father Did.* Retrieved from https://www.forbes.com/sites/kashmirhill/2012/02/16/how-target-figured-out-a-teen-girl-was-pregnant-before-her-father-did/#729840376668.

Hornik. K. (2017). *R FAQ: Frequently Asked Questions on R.* Retrieved from https://cran.r-project.org/doc/FAQ/R-FAQ.html#Why-is-R-named-R_003f.

Hunter, J., Dale, D., Firing, E., Droettboom, M. (2017). *matplotlib: pyplot*. Retrieved from https://matplotlib.org/api/pyplot_api.html.

Ihaka, R. (1998). *R: Past and Future History: A Draft of a Paper for Interface '98*. Retrieved from https://www.stat.*acklaretrnd*.ac.nz/~ihaka/downloads/Interface98.pdf.

Klein, B. (n.d.). *History of Python*. Retrieved from https://www.python-course.eu/python3_history_and_philosophy.php.

Livingstone, R. (2017). *The future of online advertising is big data and algorithms.* Retrieved from http://theconversation.com/the-future-of-online-advertising-is-big-data-and-algorithms-69297.

Marr, B. (2016). *What Is The Difference Between Artificial Intelligence And Machine Learning?* Retrieved from https://www.forbes.com/sites/bernardmarr/2016/12/06/what-is-the-difference-between-artificial-intelligence-and-machine-learning/#211f24ff2742.

Mitchum, R. (2016). *Using data science to confront policing challenges*. Retrieved from https://news.uchicago.edu/article/2016/08/25/using-data-science-confront-policing-challenges.

Moler, C. (2004). *The Origins of MATLAB*. Retrieved from https://www.mathworks.com/company/newsletters/articles/the-origins-of-matlab.html.

Nix, E. (2015). *What is the Rosetta Stone?* Retrieved from http://www.history.com/news/ask-history/what-is-the-rosetta-stone.

Piatetsky, G. (2014). *Four main languages for Analytics, Data Mining, Data Science*. Retrieved from https://www.kdnuggets.com/2014/08/four-main-languages-analytics-data-mining-data-science.html.

Press, Gil. (2013). *A Very Short History of Data Science.* Retrieved from https://www.forbes.com/sites/gilpress/2013/05/28/a-very-short-history-of-data-science/#3a0507b55cfc.

Rosetta Stone. (n.d.). Retrieved from https://www.rosettastone.com/.

RStudio Support. (2017). *Using Sweave and knitr*. Retrieved from https://support.rstudio.com/hc/en-us/articles/200552056-Using-Sweave-and-knitr.

SAS Institute. (2018). *About SAS: Giving you The Power to Know since 1976.* Retrieved from https://www.sas.com/en_us/company-information.html#.

SAS Institute. (2018). *Machine Learning: What it is and why it matters*. Retrieved from https://www.sas.com/en_us/insights/analytics/machine-learning.html.

SAS Institute. (2017). *SGPLOT Procedure*. Retrieved from http://documentation.sas.com/?docsetId=grstatproc&docsetTarget=p1t32i8511t1gfn17sw07yxtazad.htm&docsetVersion=9.4&locale=en.

Scikit-learn. (2017). *scikit-learn: Machine Learning in Python*. Retrieved from http://scikit-learn.org/stable/.

Sridharan, M. (2015). *Data Visualization – The Rosetta Stone of Data Science*. Retrieved from https://icrunchdata.com/blog/535/data-visualization-the-rosetta-stone-of-data-science/.

Tukey, J.W. (1962). The Future of Data Analysis. *The Annals of Mathematical Statistics*, 33(1), 1-67.

U.S. Food & Drug Administration (2017). *Electronic Regulatory Submissions and Review Helpful Links.* Retrieved from https://www.fda.gov/Drugs/DevelopmentApprovalProcess/FormsSubmissionRequirements/ElectronicSubmissions/ucm085361.htm.

Waskom, M. (2017). *Seaborn: statistical data visualization.* Retrieved from https://seaborn.pydata.org/.

Wickham, H., Chang, W. (2016). *Package 'ggplot2'.* Retrieved from https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf.

Yadav, H. (2017). *Difference between AI, Machine Learning and Deep Learning*. Retrieved from https://harshityadav95.blogspot.com/2017/06/difference-between-ai-machine-learning.html.

Appendices

A: MATLAB References and Resources

Create Functions in File. (n.d.). *MathWorks*. Retrieved from https://www.mathworks.com/help/matlab/matlab_prog/create-functions-in-files.html.

Data Types. (n.d.). *MathWorks*. Retrieved from https://www.mathworks.com/help/matlab/data-types_data-types.html.

Descriptive Statistics. (n.d.). *MathWorks*. Retrieved from https://www.mathworks.com/help/matlab/descriptive-statistics.html.

Functions – By Category. (n.d.). *MathWorks*. Retrieved from https://www.mathworks.com/help/matlab/functionlist.html.

MATLAB. (n.d.). *MathWorks*. Retrieved from https://www.mathworks.com/products/matlab.html.

Matrices and Arrays. (n.d.). *MathWorks*. Retrieved from https://www.mathworks.com/help/matlab/learn_matlab/matrices-and-arrays.html.

Statistics and Machine Learning Toolbox. (n.d.). *MathWorks*. Retrieved from https://www.mathworks.com/help/stats/index.html.

B: Python References and Resources

Built-in Types. (n.d.). *Python Software Foundation*. Retrieved from https://docs.python.org/2/library/stdtypes.html.

Built-in Functions. (n.d.). *Python Software Foundation*. Retrieved from https://docs.python.org/3.1/library/functions.html.

Data Structures. (n.d.). *Python Software Foundation*. Retrieved from https://docs.python.org/2/tutorial/datastructures.html.

decimal – Decimal fixed point and floating point arithmetic. (n.d.). *Python Software* Foundation. Retrieved from https://docs.python.org/2/library/decimal.html.

Intro to Data Structures. (n.d.). *Pandas*. Retrieved from https://pandas.pydata.org/pandas-docs/stable/dsintro.html.

Introduction: What is PyFlux. (n.d.). *PyFlux*. Retrieved from http://pyflux.readthedocs.io/en/latest/.

NumPy. (n.d.). *NumPy*. Retrieved from http://www.numpy.org/.

numpy.array. (n.d.). *SciPy.org*. Retrieved from https://docs.scipy.org/doc/numpy/reference/generated/numpy.array.html.

pandas.DataFrame. (n.d.). *Pandas*. Retrieved from http://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html.

PyClustering. (n.d.). *Python Software Foundation*. Retrieved from https://pypi.python.org/pypi/pyclustering.

Prophet: Forecasting at scale. (n.d.). *Facebook*. Retrieved from https://facebook.github.io/prophet/.

Python. (n.d.). *Python Software Foundation.* Retrieved from https://www.python.org/.

Python – Lists. (n.d.). *Tutorials Point*. Retrieved from https://www.tutorialspoint.com/python/python_lists.htm.

Python – Strings. (n.d.). *Tutorials Point.* Retrieved from https://www.tutorialspoint.com/python/python_strings.htm.

Python – Tuples. (n.d.). *Tutorials Point*. Retrieved from https://www.tutorialspoint.com/python/python_tuples.htm.

Python Data Analysis Library. (n.d.). *Pandas*. Retrieved from http://pandas.pydata.org/.

Python Package Introduction. (n.d.). *XGBoost*. Retrieved from http://xgboost.readthedocs.io/en/latest/python/python_intro.html.

PyPlot. (n.d.). *Matplotlib*. Retrieved from https://matplotlib.org/api/pyplot_api.html.

re – Regular expression operations. (n.d.). *Python Software Foundation*. Retrieved from https://docs.python.org/2/library/re.html#module-re.

scikit-learn: Machine Learning in Python. (n.d.). *scikit-learn*. Retrieved from http://scikit-learn.org/stable/.

seaborn: statistical data visualization. (n.d.). *seaborn*. Retrieved from https://seaborn.pydata.org/.

Sets and Frozensets. (n.d.). *Python Course*. Retrieved from https://www.python-course.eu/sets_frozensets.php.

Welcome to Bokeh. (n.d.). *Bokeh*. Retrieved from http://bokeh.pydata.org/en/latest/.

Welcome to Statsmodels's Documentation. (n.d.). *StatsModels*. Retrieved from http://www.statsmodels.org/stable/index.html.

C: R References and Resources

Bergmeir, C., Benitez, J. M., et al. (2017). *Package 'RSNNS'*. Retrieved from https://cran.r-project.org/web/packages/RSNNS/RSNNS.pdf.

Breiman, L., Cutler, A., Liaw, A., Wiener, M. (2018). *Package 'randomForest'*. Retrieved from https://cran.r-project.org/web/packages/randomForest/randomForest.pdf.

Character Vectors. (n.d.). *R Documentation*. Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/base/html/character.html.

Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y. (2018). *Package 'xgboost'*. Retrieved from https://cran.r-project.org/web/packages/xgboost/xgboost.pdf.

Complex Numbers and Basic Functionality. (n.d.). *R Documentation*. Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/base/html/complex.html.

Couture-Beil, A. (2015). *Package 'rjson'*. Retrieved from https://cran.r-project.org/web/packages/rjson/rjson.pdf.

Data Frames. (n.d.). *R Documentation*. Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/base/html/data.frame.html.

Factors. (n.d.). *R Documentation*. Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/base/html/factor.html.

Hahsler, M., Piekenbrock, M., Arya, S., Mount, D. (2017). *Package 'dbscan'*. Retrieved from https://cran.r-project.org/web/packages/dbscan/dbscan.pdf.

Hyndman, R., O'Hara-Wild, M., Bergmeir, C., Razbash, S., Wang, E. (2017). *Package 'forecast'*. Retrieved from https://cran.r-project.org/web/packages/forecast/forecast.pdf.

Integer Vectors. (n.d.). *R Documentation*. Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/base/html/integer.html.

Karatzoglou, A., Smola, A., Hornik, K. (2016). *Package 'kernlab'*. Retrieved from https://cran.r-project.org/web/packages/kernlab/kernlab.pdf.

Kuhn, M., et al. (2018). *Package 'caret'*. Retrieved from https://cran.r-project.org/web/packages/caret/caret.pdf.

Logical Vectors. (n.d.). *R Documentation*. Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/base/html/logical.html.

List. (n.d.). *R Tutorial*. Retrieved from http://www.r-tutor.com/r-introduction/list.

Matrices. (n.d.). *R Documentation*. Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/base/html/matrix.html.

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A., Leisch, F., Chang, C., Lin., C. (2017). *Package 'e1071'*. Retrieved from https://cran.r-project.org/web/packages/e1071/e1071.pdf.

Named List Members. (n.d.). *R Tutorial*. Retrieved from http://www.r-tutor.com/r-introduction/list/named-list-members.

Numeric Vectors. (n.d.). *R Documentation*. Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/base/html/numeric.html.

Prophet: Forecasting at scale. (n.d.). *Facebook*. Retrieved from https://facebook.github.io/prophet/.

R – Arrays. (n.d.). *Tutorials Point*. Retrieved from https://www.tutorialspoint.com/r/r_arrays.htm.

R – Data Types. (n.d.). *Tutorials Point*. Retrieved from https://www.tutorialspoint.com/r/r_data_types.htm.

Raw Vectors. (n.d.). *R Documentation*. Retrieved from https://stat.ethz.ch/R-manual/R-devel/library/base/html/raw.html.

RDocumentation. (n.d.). *RDocumentation*. Retrieved from https://www.rdocumentation.org/.

Ridgeway, G., et al. (2017). *Package 'gbm'*. Retrieved from https://cran.r-project.org/web/packages/gbm/gbm.pdf.

Ripley, B. D. (2018). *Package 'tree'*. Retrieved from https://cran.r-project.org/web/packages/tree/tree.pdf.

Warnes, G.R., Bolker, B., Gorjanc, G., Grothendieck, G., Korosec, A., Lumley, T., MacQueen, D., Magnusson, A., Rogers, J., et al. (2017). *Package 'gdata'*. Retrieved from https://cran.r-project.org/web/packages/gdata/gdata.pdf.

Wehrens, R., Kruisselbrink, J. (2017). *Package 'kohonen'*. Retrieved from https://cran.r-project.org/web/packages/kohonen/kohonen.pdf.

Wickham, H., Chang, W. (2016). *Package 'ggplot2'*. Retrieved from https://cran.r-project.org/web/packages/ggplot2/ggplot2.pdf.

Wickham, H., Francois, R., Henry, L., Muller, K. (2017). *Package 'dplyr'*. Retrieved from https://cran.r-project.org/web/packages/dplyr/dplyr.pdf.

D: SAS References and Resources

%INCLUDE Statement. (n.d.). *SAS Institute Inc.* Retrieved from http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000214504.htm.

ARRAY Statement. (n.d.). *SAS Institute Inc.* Retrieved from http://documentation.sas.com/?docsetId=lestmtsref&docsetTarget=p08do6szetrxe2n136ush727sbuo.htm&docsetVersion=9.4&locale=en.

CONTENTS Procedure. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm#n1hqa4dk5tay0an15nrys1iwr5o2.htm.

Data Types for SAS Data Sets. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/fedsqlref/67364/HTML/default/viewer.htm#n19bf2z7e9p646n0z224cokuj567.htm.

Definition of a SAS Data Set. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/lrcon/62955/HTML/default/viewer.htm#a001005709.htm.

The DMDB Procedure. (2000). *SAS Institute Inc*. Retrieved from https://support.sas.com/documentation/onlinedoc/miner/em43/dmdb.pdf.

How the DATA Step Works: A Basic Introduction. (n.d.). *SAS Institute Inc*. Retrieved from http://documentation.sas.com/?docsetId=basess&docsetTarget=n053a58fwk57v7n14h8x7y7u34y4.htm&docsetVersion=9.4&locale=en.

Eberhardt, P. (2012). *The SAS® Hash Object: It's Time to .find() Your Way Around.* Retrieved from http://support.sas.com/resources/papers/proceedings12/147-2012.pdf.

IF-THEN/ELSE Statement. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000202239.htm.

INFILE Statement. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000146932.htm.

INPUT Statement. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000146292.htm.

MERGE Statement. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000202970.htm.

The NEURAL Procedure. (2000). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/onlinedoc/miner/em43/neural.pdf.

OUTPUT Statement. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000194540.htm.

Overview: CLUSTER Procedure. (n.d.). *SAS Institute Inc*. Retrieved from https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_cluster_sect001.htm.

Overview: COMPARE Procedure. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm#p1l5iwaf47ma83n1euxxp10g8xh5.htm.

Overview: CORR Procedure. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/procstat/66703/HTML/default/viewer.htm#procstat_corr_overview.htm.

Overview: FCMP Procedure. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm#n0pio2crltpr35n1ny010zrfbvc9.htm.

Overview: Export Procedure. (n.d.). *SAS Institute Inc.* Retrieved from http://documentation.sas.com/?docsetId=proc&docsetTarget=p09k160vk93xxhn171zz1z6551w2.htm&docsetVersion=9.4&locale=en.

Overview: FASTCLUS Procedure. (n.d.). *SAS Institute Inc.* Retrieved from https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_fastclus_sect001.htm.

Overview: FREQ Procedure. (n.d.). *SAS Institute Inc.* Retrieved from https://support.sas.com/documentation/cdl/en/statug/63962/HTML/default/viewer.htm#statug_freq_sect001.htm.

Overview: HPSPLIT Procedure. (n.d.). *SAS Institute Inc.* Retrieved from http://support.sas.com/documentation/cdl/en/stathpug/66410/HTML/default/viewer.htm#stathpug_hpsplit_overview.htm.

Overview: MEANS Procedure. (n.d.). *SAS Institute Inc.* Retrieved from http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm#n0k7qr5c2ah3stn10g1lr5oytz57.htm.

Overview: PLM Procedure. (n.d.). *SAS Institute Inc.* Retrieved from https://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm#statug_plm_a0000000115.htm.

Overview: PRINCOMP Procedure. (n.d.). *SAS Institute Inc.* Retrieved from https://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm#statug_princomp_sect001.htm.

Overview: PRINT Procedure. (n.d.). *SAS Institute Inc.* Retrieved from http://support.sas.com/documentation/cdl/en/proc/65145/HTML/default/viewer.htm#p1dlh3svb4rrasn14h8jm6nyrj5o.htm.

Overview: PROC IMPORT. (n.d.). *SAS Institute Inc.* Retrieved from http://documentation.sas.com/?docsetId=acpcref&docsetTarget=p0mqdq52ktqgjjn1vseqoiuyt3v1.htm&docsetVersion=9.4&locale=en.

Overview: REG Procedure. (n.d.). *SAS Institute Inc.* Retrieved from https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_reg_sect001.htm.

Overview: SCORE Procedure. (n.d.). *SAS Institute Inc.* Retrieved from https://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm#statug_score_sect001.htm.

Overview: SGPLOT Procedure. (n.d.). *SAS Institute Inc.* Retrieved from http://documentation.sas.com/?docsetId=grstatproc&docsetTarget=p1t32i8511t1gfn17sw07yxtazad.htm&docsetVersion=9.4&locale=en.

Overview: SORT Procedure. (n.d.). *SAS Institute Inc.* Retrieved from http://documentation.sas.com/?docsetId=proc&docsetTarget=p0ha9ymifyaqldn14m2xlqw252wa.htm&docsetVersion=9.4&locale=en.

Overview: SURVEYSELECT Procedure. (n.d.). *SAS Institute Inc.* Retrieved from https://support.sas.com/documentation/cdl/en/statug/63347/HTML/default/viewer.htm#statug_surveyselect_sect001.htm.

Overview: TREE Procedure. (n.d.). *SAS Institute Inc.* Retrieved from https://support.sas.com/documentation/cdl/en/statug/63033/HTML/default/viewer.htm#statug_tree_sect001.htm.

Run R code inside SAS easily. (2017). *SAS Communities*. Retrieved from https://communities.sas.com/t5/General-SAS-Programming/Run-R-code-inside-SAS-easily/td-p/210116.

SAS Analytical Products 14.2 Documentation. (n.d.). *SAS Institute Inc*. Retrieved from https://support.sas.com/documentation/142/index.html.

SAS® Enterprise Miner ™ 14.1: High-Performance Procedures. (2015). *SAS Institute Inc.* Retrieved from https://support.sas.com/documentation/onlinedoc/miner/em14/emhpprcref.pdf.

SAS/ETS® 13.2 User's Guide: The ARIMA Procedure. (2014). *SAS Institute Inc*. Retrieved from https://support.sas.com/documentation/onlinedoc/ets/132/arima.pdf.

SAS/ETS® 13.2 User's Guide: The ESM Procedure. (2014). *SAS Institute Inc*. Retrieved from https://support.sas.com/documentation/onlinedoc/ets/132/esm.pdf.

SAS/IML ® 9.22 User's Guide. (2010). *SAS Institute Inc*. Retrieved from https://support.sas.com/documentation/cdl/en/imlug/63541/PDF/default/imlug.pdf.

SAS/STAT® 9.2 User's Guide: The GENMOD Procedure. (2008). *SAS Institute Inc*. Retrieved from https://support.sas.com/documentation/cdl/en/statuggenmod/61787/PDF/default/statuggenmod.pdf.

SET Statement. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000173782.htm.

The SGSCATTER Procedure. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/grstatproc/62603/HTML/default/viewer.htm#sgscatter-chap.htm.

SQL Procedure: Overview. (n.d.). *SAS Institute Inc*. Retrieved from http://documentation.sas.com/?docsetId=sqlproc&docsetVersion=9.4&docsetTarget=n0w2pkrm208upln11i9r4ogwyvow.htm&locale=en.

The TIMESERIES Procedure. (n.d.). *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/etsug/66840/HTML/default/viewer.htm#etsug_timeseries_overview.htm.

WHERE Statement. (n.d.) *SAS Institute Inc*. Retrieved from http://support.sas.com/documentation/cdl/en/lrdict/64316/HTML/default/viewer.htm#a000202951.htm.