

**ΟΙΚΟΝΟΜΙΚΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΑΘΗΝΩΝ**



**ATHENS UNIVERSITY
OF ECONOMICS
AND BUSINESS**

SCHOOL OF INFORMATION SCIENCES & TECHNOLOGY

***DEPARTMENT OF STATISTICS
POSTGRADUATE PROGRAM***

Popularity and Usage of Statistical/Data Science Software

Written By
Konstantinos Kirillov

M.Sc. Thesis
Submitted to the Department of Statistics
of the Athens University of Economics and Business
in partial fulfillment of the requirements for
the degree of Master of Science in Statistics

Athens, Greece
January 2023



ACKNOWLEDGEMENTS

I would like to personally thank Professor Panagiotis Papastamoulis for his invaluable help during this thesis. Without his ideas and insight, this thesis would not be on this level of quality.



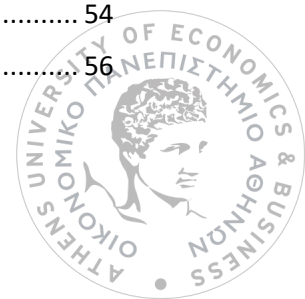
ABSTRACT

The aim of this thesis is to collect, present and analyze the popularity of the statistical tools chosen, all the while trying to determine whether there are latent relationships between them. At first the theory to be used in the subsequent chapters is presented. After that, comes a brief discussion about how the data was collected. After the aggregation of data is complete, then follows an extensive analysis of the dataset and its variables. Afterwards, there is an attempt at grouping the collected observations, by using techniques aimed at clustering for binary data. And finally, the data set is reconstructed through a web scraping procedure and the new data are compared to the old data that were collected in the beginning of the thesis.



Table of Contents:

Chapter 1: Introduction	1
1.1 About Popularity.....	1
1.2 How this Thesis is Structured	2
Chapter 2: Theory	3
2.1 Data Analysis	3
2.1.1 Chi-Squared Test.....	3
2.1.2 Correlation with Cramer’s V	4
2.1.3 Network Analysis	5
2.1.3.1 Network Clustering through Greedy Optimization of Modularity	6
2.1.3.2 Network Clustering through Latent Positioning.....	8
2.1.4 Testing for Statistical Significance.....	9
2.1.5 Benjamini-Hochberg Multiple Hypothesis Testing Procedure	11
2.1.6 Multiple Correspondence Analysis (MCA).....	12
2.2 Clustering for Binary Variables	15
2.2.1 Hierarchical Clustering	15
2.2.2 Finite Mixture Models for Model-Based Clustering (FlexMix).....	17
2.2.3 Bayesian Clustering through Finite Mixture Models (BayesBinMix).....	19
2.2.4 Indices for Comparison of Partitions.....	22
2.2.5 Indices for Detecting Clusters through Hierarchical Clustering Methods	24
2.3 Web Scraping	26
Chapter 3: Collecting the Data	31
3.1 Description of the Dataset and its Variables.....	31
3.1.1 Statistical Tools	31
3.1.2 Other (Grouping) Variables	39
3.2 Observations Upon the Collected Data	42
Chapter 4: Analyzing the Data	44
4.1 An Initial Picture (Word Cloud) of Our Data.....	44
4.2 Frequencies of the Statistical Tools	45
4.3 Weighted Frequencies of the Statistical Tools	48
4.4 Proprietary VS Open-Source Statistical Tools	51
4.5 Correlations.....	54
4.6 The “Country” Variable	56



4.7 The “Seniority Level” Variable.....	59
4.8 The “Industry” Variable.....	61
4.9 The “Term Searched” Variable	64
4.10 Network Analysis	67
4.11 Network Clustering for Statistical Tools.....	71
4.11.1 Clustering through Greedy Optimization of Modularity	71
4.11.2 Clustering through Latent Positioning.....	73
4.12 Testing for Statistical Significance	75
4.13 Multiple Correspondence Analysis (MCA)	77
Chapter 5: Clustering for Binary Data	85
5.1 Hierarchical Clustering.....	85
5.2 Finite Mixture Models for Model-Based Clustering (FlexMix)	88
5.3 Bayesian Clustering through Finite Mixture Models (BayesBinMix)	96
5.4 Comparing the Results of the Clustering Methods	104
Chapter 6: Trends	108
Chapter 7: Conclusion	111
Bibliography.....	112



CHAPTER 1: INTRODUCTION

1.1 About Popularity

Popularity can be considered a subjective concept. That which is common knowledge to one person, may be entirely foreign to another. In general, obscure subjects would never be considered popular, especially if they refer to something new or known to a very small number of individuals. If however, a majority of people recognize the subject being discussed, asked or mentioned, then this same subject may be assumed to be popular. Therefore, it would appear that, measuring popularity can be a tasking procedure.

How does one measure the popularity of statistical software? Here one may find many answers to this same question (R. A. Muenchen, 2013). Some may say that the number of downloads and sales of statistical software suffices to say whether it is popular or not. Let us keep in mind however, that a lot of people have two or more computers that they might use (perhaps at home and the office) and so would download the same statistical software on both in order to do their job. That extra download would not increase the popularity of the software, as the person would have the same general knowledge of it. Thus, as a measure, it appears unreliable. Others might say that the number of printed books or manuals could form a solid basis to answer our question. It makes sense as the newer the software, the less printed material it should have and furthermore, if one such software becomes obsolete, or its usage wanes with time, then it will show by a gradual decline in learning material. This is however a difficult way to measure popularity when we want to emphasize the “statistical” part. Python for example is a widely used programming language by people of all fields imaginable. Should we have wanted to include this language, we would have to isolate the books that are mainly aimed at statistics, which is a very arduous and slow task in itself. And the list of suggestions continues with arguably decent measures of popularity, which however have at least one downside, like the number of skilled engineers worldwide, courses provided by the various boot camp websites and third-party vendors, etc., for instance, much like it is done for the calculations of the TIOBE Index.

Out of all the ways we could measure popularity, this thesis shall focus on job advertisements as its basis in order to determine the most famous statistical and data science software. Naturally, it comes with at least one downside, as it does not include the academic part, however it is a reasonable way to measure popularity. After all, enterprises want their employees to use the fastest, most reliable and cost-efficient software so as to have an upper hand on their competitors. There are of course some outliers, like companies that do not compete, or are non-profit, however they too depict the popularity accurately. If one or two firms use COBOL and need professionals who know how to code in it, then this clearly shows how popular it is.



1.2 How this Thesis is Structured

In Chapter 2, the whole theory that is used in this thesis is presented in the order that it is going to be used. Chapter 3 discusses how the data was collected and mentions some peculiarities that were observed during the collection, something which might help with the interpretation of the analysis later on. In Chapter 4 comes the description and analysis of the data, where connections and latent relationships between the data are explored. In Chapter 5, there is an attempt at clustering our data both by using hierarchical and model-based algorithms. And in Chapter 6, the data are collected again through a web scraping method, five months after the initial collection and compared to the ones that were collected at the beginning.



CHAPTER 2: THEORY

2.1 Data Analysis

2.1.1 Chi-Squared Test

A Chi-Square Test is a hypothesis testing method for categorical variables, which determines whether there are differences between the observed and the expected data (A. Agresti, 2007). There are two main types of it:

1. The *Chi-Square Test* for the independence between two variables, which examines if those said variables are somehow related to each other.
2. The *Chi-Square Test* for the goodness of fit of a variable, which determines how likely it is that this variable comes from a certain distribution.

It is known to have the following formula:

$$X^2 = \sum_{i=0}^n \frac{(\text{Observed}_i - \text{Expected}_i)^2}{\text{Expected}_i}$$

where:

- Observed_i : Is the observed value in the cell i .
- Expected_i : Is the expected value in the cell i (assuming that the null hypothesis is true).
- n : Is the total number of cells of our contingency table.

The test statistic X^2 asymptotically follows the *Chi-square Distribution* X_{df}^2 , meaning that:

$$X^2 \sim X_{df}^2$$

With df being the degrees of freedom, that are calculated in the following manner:

$$\text{degrees of freedom} = (\text{number of rows} - 1)(\text{number of columns} - 1)$$

Thus, the calculated test statistic is compared to the value that X_{df}^2 takes at a certain significance level α , which is predetermined; and then the null hypothesis is either rejected or not.



2.1.2 Correlation with Cramer's V

An interesting thing to see is how strong the correlations are, between the variables present in a set of data. Looking throughout the dataset, however, one would notice that our variables are nominal, categorical variables. Thus, in order to calculate the degree of dependency between the variables, *Cramer's V* shall be applied.

Here is how it came to be:

Having knowledge of the *Pearson's Chi-Squared* statistic, Harald Cramér proposed his own version of a contingency coefficient (H. Cramér, 1946), whose purpose was to take two nominal variables and measure how strongly or weakly associated they were amongst themselves. This proposed coefficient became known as *Cramer's V* (A. M. Liebetrau, 1983), and it had the following formula:

$$Cramer's V = \sqrt{\frac{\varphi^2}{degrees\ of\ freedom}} = \sqrt{\frac{X^2/n}{q-1}} = \sqrt{\frac{X^2/n}{\min(r,c)-1}}$$

where:

- n is the total number of observations.
- c is the total number of columns.
- r is the total number of rows.
- X^2 is the *Pearson's Chi-Square* statistic.

Cramer's V takes values between 0 and 1. Talking about the two extremes, should two variables take the value 0, it shows that they are independent, while the value 1 implies perfect dependency amongst them.



2.1.3 Network Analysis

In a few words, Network Analysis (M. J. Newman, 2010; L. A. Douglas, 2015) is a structure that shows the way in which subjects are connected to each other.

What is a Network?

To put it simply: A *network* is a collection of points that are connected with lines. The points are referred to as *nodes* or *vertices*, while the lines as *edges*. Thus, should two points form a relationship between them, then this relationship is represented by the *edge* that joins those two points together.

The aggregation of all the relationships possible between all the nodes within a dataset creates a network, which can then be visually represented through a graph.

What is an Adjacency Matrix?

It is a mathematical representation of the network. The *adjacency matrix* is a square matrix, whose lines and columns are the uniquely labelled *vertices*. The reason it is formed, is to connect these said *vertices* through *edges* and show how and if the *nodes* interact with each other. It depicts the structure of the network by making a list of *edges* based on the *vertices* provided.

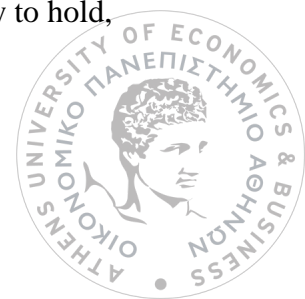
Thus, if an *adjacency matrix* A has $i = 1, \dots, n$ number of rows and $j = 1, \dots, n$ number of columns, then its elements are formed in the following manner:

$$A_{ij} = \begin{cases} 1, & \text{if there is a connection between nodes } i \text{ and } j, \\ 0, & \text{otherwise.} \end{cases}$$

About the Measurements of Connectivity

These are a few of the ways to measure connectivity inside a network:

- *Number of Vertices*: The number of points (or *nodes*) in a network.
- *Number of Edges*: The number of lines (or connections) in a network.
- *Density*: The percentage of actual connections in a network out of all the possible connections that can be formed between the points.
- *Reciprocity*: The percentage of connections that are reciprocated back. Thus, if A is connected to B, then B also must be connected to A. In undirected networks this is always 100%.
- *Transitivity*: The percentage of all the triangular connections of nodes within a network. Meaning that if A is connected to B, and B is connected to C, then for transitivity to hold, A must be also connected to C.



2.1.3.1 Network Clustering through Greedy Optimization of Modularity

After forming a network and its connections, there are a few ways that can potentially detect groups between variables. One of these ways is the following, which seeks structure in communities formed through a greedy optimization of modularity (A. Clauset, M. E. J. Newman, and C. Moore, 2005).

The main idea is that in the real world, lots of networks are represented in a sparse and hierarchical manner and hence, may form communities based on that structure. Therefore, if there are n vertices and m edges, then the relationship that must be formed between them is $m \sim n$.

This algorithm uses *modularity* (Q) to separate communities, meaning that a community must have many edges between its own nodes and less edges with other communities and their nodes. Which essentially means that nodes which have more in common with each other, shall form lots of connections between them, while if they belong to other communities, then these nodes will have lesser lines connecting them.

The greedy optimization refers to the fact that at each point, the optimization is performed locally and all these locals are aggregated at the end to adequately approximate a global maximum and that is why the authors of this procedure mention that it is a faster method than the one proposed by Girvan and Newman in their respective works (M. Girvan and M. E. J. Newman, 2002; M. Girvan and M. E. J. Newman, 2004).

Here is how the algorithm works:

Initially each node is considered as part of a community of only one member and therefore:

$$\Delta Q_{ij} = \begin{cases} \frac{1}{2}m - \frac{k_i k_j}{(2m)^2}, & \text{if } i, j \text{ are connected} \\ 0, & \text{otherwise} \end{cases}$$

and

$$a_i = \frac{k_i}{2m}$$

where:

- ΔQ_{ij} is a sparse matrix that has all the pairs between objects i and j , that have at least one edge between them.
- m is the number of edges within our graph, with: $m = \frac{1}{2} \sum_{ij} A_{ij}$, where A_{ij} is the adjacency matrix for objects i and j .
- k_i is the degree of a vertex i , with $k_i = \sum_j A_{ij}$, with A_{ij} being once again the adjacency matrix for objects i and j .
- a_i being the elements of an ordinary vector with the formula that is described above.



It then calculates a max heap H , which takes each row of ΔQ_{ij} and stores the largest element from it.

Having all these maxima of rows of ΔQ_{ij} , the algorithm picks the largest of them from the max heap H , it joins the two objects i and j and then it updates all the values that this join affects, meaning: ΔQ_{ij} , and then H and a_i . This procedure repeats until only one community remains.

Thus, if two communities i and j form a strong connection, then during the update, they merge into the new line and column j together, while the row and column i is removed completely.

Let a third community of an object n exist in our network, then the update of ΔQ_{jn} happens in the following manner:

$$\Delta Q'_{jn} = \begin{cases} \Delta Q_{in} + \Delta Q_{jn}, & \text{if community } n \text{ is connected to both } i \text{ and } j. \\ \Delta Q_{in} - 2a_j a_k, & \text{if community } n \text{ is connected to } i \text{ but not to } j. \\ \Delta Q_{jn} - 2a_i a_k, & \text{if community } n \text{ is connected to } j \text{ but not to } i. \end{cases}$$



2.1.3.2 Network Clustering through Latent Positioning

This procedure (P. N. Krivitsky and M. S. Handcock, 2008) assumes that there is an unobserved “*social space*” between objects (or actors, as the paper names them) and that within that space, the objects may independently form bonds between themselves. The way that it is determined if two objects shall form a link is through measuring the distance between them and afterwards, calculating the probabilities that these two are close, or far apart within that space.

In this package (*latentnet*), social distances are represented in a Euclidean space. It utilizes a Markov Chain Monte Carlo (MCMC) algorithm for Bayesian inference, while also providing a way to accommodate non-binary edge weights of the data.

A general form of the algorithm, is as follows:

$$Pr(Y = y|\beta, \mathbf{x}, \mathbf{Z}) = \prod_{(i,j) \in Y} Pr(Y_{i,j} = y_{i,j}|\beta, \mathbf{x}, \mathbf{Z}), \quad (a)$$

$$Pr(Y_{i,j} = y_{i,j}|\beta, \mathbf{x}_{\cdot, i,j}, |Z_i - Z_j|) = f[y_{i,j}|E(Y_{i,j}|\beta, \mathbf{x}_{\cdot, i,j}, |Z_i - Z_j|)], \quad (b)$$

$$E(Y_{i,j}|\beta, \mathbf{x}_{\cdot, i,j}, |Z_i - Z_j|) = g^{-1}[\eta_{i,j}(\beta, \mathbf{x}_{\cdot, i,j}, |Z_i - Z_j|)], \quad (c)$$

$$\eta_{i,j}(\beta, \mathbf{x}_{\cdot, i,j}, |Z_i - Z_j|) = \sum_{k=1}^p x_{\cdot, i,j} \beta_k - |Z_i - Z_j|. \quad (d)$$

where:

- $\mathbf{Z} = \{Z_i\}_{i=1}^n$ are the positions of the objects in the “*social space*”.
- $|Z_i - Z_j|$ is the Euclidean distance between two objects i and j .
- p is the total number of parameters within our data.
- n is the total number of observations within our data.

So, what does this hierarchical structure of equations above tells us? In essence, it says that the first equation (a) informs us that the links made between objects are independent. In the second equation (b) takes it a step further, implying that the conditional probabilities of the first equation are dependent on the covariates and latent positions only through their conditional mean. The third equation (c) moves ahead and relates that the conditional mean above is expressed through the inverse of a known link function g , as well as a predictor function $\eta_{i,j}$. And the final equation (d) says that this predictor function, is a linear function.



2.1.4 Testing for Statistical Significance

Since our variables are nominal, categorical variables, should one need to test the statistical significance of independence between them, then the standard thing one could do would be a *Chi-Square Test*.

It would be ideal to do this test on our data and get the *p-values*, however there is a slight complication. When one might try to construct contingency tables between statistical tools and some other variables (especially the “*Industry*” variable), then they would see that these tables are populated very sparsely by data. These unbalanced data tables make the results of these asymptotic tests questionable. Luckily, there is the *Markov Chain – Monte Carlo (MCMC)* option (C. R. Mehta and N. R. Patel, 1989; A. Agresti, D. Wackerly, and J. M. Boyett, 1979) that estimates those same *p-values* quite accurately, while also being unaffected by the aforementioned sparsity in our data.

The MCMC estimation of *p-values* for the Chi-Square Test:

The algorithm uses a combination of works by Hope (A. C. A. Hope, 1968) and Patefield (W. M. Patefield, 1981).

Here is a generic contingency table for summarizing two nominal variables against each other:

	Variable 1				Total
Variable 2	x_{11}	x_{12}	...	x_{1c}	m_1
	x_{21}	x_{22}	...	x_{2c}	m_2
	\vdots	\vdots	\ddots	\vdots	\vdots
	x_{r1}	x_{r2}	...	x_{rc}	m_r
Total	n_1	n_2	...	n_c	N

Table 1: A contingency table for two nominal variables of r and c number of elements respectively.

Here, the margins of this table are the values m_1, m_2, \dots, m_r and n_1, n_2, \dots, n_c . The way that *MCMC* algorithms work, is: They keep these margins fixed and take random samples from the group of tables that have the same margins, while also calculating the *likelihood-ratio statistic*. Afterwards, this *likelihood statistic* of each of the random samples is compared to the one in the reference set and either contributes to the calculation of the *p-value* or not. Meaning that if that statistic of the random sample is as extreme as the one in the reference set (or that if it is greater than or equal to the *likelihood-ratio statistic* initially calculated), then this value is considered to contribute towards the estimation of the *p-value*.

The total favourable tables that can contribute to the calculation of the *p-value* come under the following hypergeometric probability:



$$P(\{x_{ij}\}) = \frac{\prod_{i=1}^r m_i! \prod_{j=1}^c n_j!}{N! \prod_{i=1}^r \prod_{j=1}^c x_{ij}}$$

Therefore, if the sampled tables come in proportion to their hypergeometric probability and in total, M tables are sampled from the reference set, while R of these tables contribute to the calculation of the p -value (meaning that they are as extreme as the original table), then the estimation of the p -value, according to the *MCMC* algorithm will be:

$$p - \widehat{value} = \frac{R}{M}$$

With a variance of:

$$Var(p - \widehat{value}) = \frac{p - \widehat{value} (1 - p - \widehat{value})}{M}$$

Therefore, the more tables are sampled, the lesser the width of the confidence interval of the p -values, which in turn leads to pretty accurate results.



2.1.5 Benjamini-Hochberg Multiple Hypothesis Testing Procedure

There are many ways to test a lot of hypotheses and one of them, is the one proposed by Y. Benjamini and Y. Hochberg, which is based on the False Discovery Rate (Y. Benjamini and Y. Hochberg, 1995).

What is the False Discovery Rate (FDR)?

Simply put, False Discovery Rate (*FDR*) is the proportion of all the Type I Errors out of all discoveries, or in other words, when one falsely rejects the null hypothesis. Right below is a table that depicts all the possible outcomes for testing a number of m hypotheses in total:

	<i>Accept H_0</i>	<i>Reject H_0</i>	<i>Total</i>
<i>H_0 is TRUE</i>	U	V	m_0
<i>H_0 is FALSE</i>	T	S	$m - m_0$
<i>Total</i>	$m - R$	R	m

Table 2: A contingency table that shows the number of errors committed, when testing m null hypotheses.

Looking at *Table 2* above, we know that U , V , T and S are unknown, random variables, while R and m are known. Using that same table, False Discovery Rate (*FDR*) is defined in the following way:

$$FDR = E[Q] = E\left[\frac{V}{V+S}\right] = E\left[\frac{V}{R}\right]$$

with:

$$Q = \begin{cases} V/R, & \text{when } R > 0 \\ 0, & \text{when } R = 0 \end{cases}$$

Benjamini-Hochberg Procedure:

Consider that there are a total of H_1, H_2, \dots, H_m hypotheses to be tested and these yield P_1, P_2, \dots, P_m *p-values*. Then the procedure of controlling for the False Discovery Rate is the following:

- Propose an acceptable False Discovery Rate (*FDR*) level, q^* .
- Order the *p-values* in a way that: $P_{(1)} \leq P_{(2)} \leq \dots \leq P_{(m)}$.

If then, k is the largest i , for which $P_{(i)} \leq \frac{i}{m} q^*$ is in effect, then reject all the previous hypotheses, or in other words, reject $H_{(i)}$, for $i = 1, 2, \dots, k$.



2.1.6 Multiple Correspondence Analysis (MCA)

Multiple Correspondence Analysis (*MCA*) is a technique that helps us analyze, detect and depict hidden structures within our data (F. Husson, S. Le, J. Pagès, 2017; M. Greenacre and J. Blasius, 2006). It works much like the Principal Component Analysis (*PCA*) (H. Abdi, and L. J. Williams, 2010), but for categorical data.

What is Correspondence Analysis:

It is an exploratory method for categorical variables, which analyzes and graphically presents the underlying design within a given data set. The tables it uses (which are called Burt tables) are symmetrical, cross-tabulation tables that involve either frequencies or counts and do not contain negative entries. The term “correspondence analysis” (*CA*) stems from the fact that the tables are constructed and studied by combining two (or more in case of *MCA*) corresponding sets of data. Much like Principal Component Analysis (*PCA*), this method interprets inertia through graphical representation, now however, it is done for two categorical variables, instead of continuous ones.

What is Multiple Correspondence Analysis:

Multiple Correspondence Analysis (*MCA*) is the same method much like the simple Correspondence Analysis (*CA*), only now it is applied between more than just two categorical variables. This technique essentially studies the similarities between individual observations and their variables and aims to provide a topology that depicts just how similar or not these observations really are. The way Multiple Correspondence Analysis (*MCA*) compares them, is firmly based on a presence-absence of the choices that they made. For instance, in our dataset, if firm 1 and firm 2 either both picked R as a prerequisite for their candidate, or none of them did, then these two companies are similar to each other, if only one of them picked this program, then they should be considered dissimilar.

There are two types of distances that it can calculate:

- Distance between the observations or individuals.
- Distance between the variables or categories.

Therefore, if we consider that our data are put in a matrix that is entirely composed of 0 and 1 and which has a total of $i = 1, 2, \dots, M$ observations in its rows and $j = 1, 2, \dots, N$ categorical variables in its columns, then:

1. The Distance Between Observations i and i' is:

$$distance_{i,i'} = \sqrt{C \sum_{j=1}^N \frac{(x_{ij} - x_{i'j})^2}{m_j}}$$

where:



- C is a constant, whose choice stems from the X^2 distance of the row profiles.
- x_{ij} is an element of the observation i and variable j from our data matrix.
- $x_{i'j}$ is an element of the observation i' and variable j from our data matrix.
- m_j is the total number of observations that belong to variable (or category) j .

2. The Distance Between Variables j and j' is:

$$distance_{j,j'} = \sqrt{C' \sum_{i=1}^M \left(\frac{x_{ij}}{m_j} - \frac{x_{ij'}}{m_{j'}} \right)^2}$$

where:

- C' is a constant, whose choice comes from the X^2 distance of the column profiles.
- x_{ij} is an element of the observation i and variable j from our data matrix.
- $x_{ij'}$ is an element of the observation i and variable j' from our data matrix.
- m_j is the total number of observations that belong to variable (or category) j .
- $m_{j'}$ is the total number of observations that belong to variable (or category) j' .

The two constants C and C' mentioned above come from the X^2 distance of column and row profiles, whose distances are:

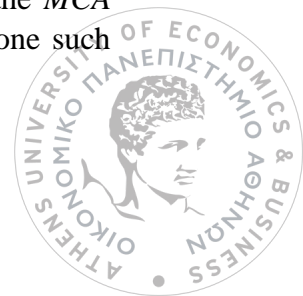
$$distance_{X^2}(\text{between row profiles } i \text{ and } i') = \sqrt{\sum_{j=1}^N \frac{1}{f_{\cdot j}} \left(\frac{f_{ij}}{f_{i\cdot}} - \frac{f_{i'j}}{f_{i'\cdot}} \right)^2}$$

and

$$distance_{X^2}(\text{between column profiles } j \text{ and } j') = \sqrt{\sum_{i=1}^M \frac{1}{f_{i\cdot}} \left(\frac{f_{ij}}{f_{\cdot j}} - \frac{f_{ij'}}{f_{\cdot j'}} \right)^2}$$

And therefore, should constants be chosen to be: $C = M/N$ and $C' = M$, then Multiple Correspondence Analysis (*MCA*) will relate back to simple Correspondence Analysis (*CA*).

What is generally expected is that the percentage of the inertia explained by the dimensions of the Multiple Correspondence Analysis (*MCA*) will be much lower when compared to the ones explained by other methods such as the Principal Component Analysis (*PCA*). The reason for that is because the *PCA* analyzes only the linear relationship between components, while the *MCA* studies general relationships between them. Therefore, should anyone want to depict one such relationship, they would need at least:



$\min(\text{Observations in Category 1}, \text{Observations in Category 2}) - 1$
as the number of dimensions.



2.2 Clustering for Binary Variables

It is common knowledge that Machine Learning is primarily divided into two categories:

- Unsupervised Machine Learning, and
- Supervised Machine Learning.

Clustering is a grouping procedure that belongs in the category of Unsupervised Machine Learning. It is a process that seeks a grouping of data (if it exists) into an (unknown) number of classes from a given dataset. The goal of such algorithms is to identify latent groupings of observations, which allow us to predict the class of observations even without a labelled response vector. There are many clustering algorithms and they have a plethora of approaches to identifying the clusters in data. In this thesis, hierarchical and model-based clustering techniques are applied to the binary data.

2.2.1 Hierarchical Clustering

This procedure (G. James, T. Hastie, R. Tibshirani and D. Witten, 2015; T. Hastie, R. Tibshirani, J. Friedman, 2013) attempts to identify and build a hierarchy of relatively homogeneous groups of cases (or variables) based on selected characteristics of data. It should be noted that we do not know in advance how many clusters we want, but in order to determine just that, we use tree-like visual representation of the observations. All the clusters are combined (or separated) based on distance measures between and within clusters.

Hierarchical Clustering Analysis can be done in two ways:

- **Divisive Hierarchical Clustering:** Here all observations start as part of one big cluster, and splits are performed over and over again, as one moves down the hierarchy. This approach is also known as a "top-down" approach. Should a cluster be separated, these two new pieces can never go back to being one again. That is why this procedure is called hierarchical, because there is a structure to the separation of the clusters.
- **Agglomerative Hierarchical Clustering:** This is the very opposite of the Divisive procedure described above. Each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy. It is also known as a "bottom-up" approach. One particular characteristic of this procedure is that once a cluster is formed, it cannot be separated again.

Up until this point it is similar to hierarchical clustering that is applied to continuous variables. The only difference is how we calculate the distances, since our data contains binary, categorical variables.



How to Calculate Distances for Binary Data:

Should we want to compare two partitions as far as their similarity is concerned, then in the binary case there are only four possible outcomes, which are presented in the table below:

Partition	U			
	Outcome	1	0	Total
V	1	a	b	$a + b$
	0	c	d	$c + d$
	Total	$a + c$	$b + d$	$a + b + c + d$

Table 3: A similarity-dissimilarity matrix between two partitions (objects) U and V .

What we can say about these two partitions is how similar or dissimilar they are. If they share the same number of ones and zeroes (or a and d in the table above), then they can be considered similar to each other, if however they do not, then these two objects are characterized as dissimilar. Here are some distance metrics that use *Table 3* above (B. S. Everitt, S. Landau, M. Leese and D. Stahl, 2011):

- *Simple Matching Distance* = $1 - \left(\frac{a+d}{a+b+c+d}\right)$
- *Jaccard Distance* = $1 - \left(\frac{a}{a+b+c}\right)$
- *Rogers & Tanimoto Distance* = $1 - \left(\frac{a+d}{a+2(b+c)+d}\right)$
- *Sneath & Sokal Distance* = $1 - \left(\frac{a}{a+2(b+c)}\right)$
- *Gower & Legendre Distance* = $1 - \left(\frac{a+d}{a+0.5(b+c)+d}\right)$

How one picks a suitable distance depends on how they want to treat the zero-zero similarity matches (or d in the *Table 3*), as well as the dissimilarities between the partitions (meaning b and c in *Table 3*).



2.2.2 Finite Mixture Models for Model-Based Clustering (FlexMix)

This is the frequentist approach to clustering binary data through the use of finite mixture models. Here is how it roughly works (F.Leisch, 2004):

Suppose finite mixture models that contain a total of $k = 1, 2, \dots, K$ components, which are of the following form:

$$h(y|x, \psi) = \sum_{k=1}^K \pi_k f(y|x, \theta_k) \quad (a)$$

with:

$$\pi_k \geq 0, \quad \sum_{k=1}^K \pi_k = 1$$

since it is a probability.

Here, y is a (multivariate) response variable, x is a vector of explanatory variables, π_k is the prior probability of component k , θ_k is a parameter vector for component k of the density function $f(y|x, \theta_k)$ and finally, $\psi = (\pi_1, \pi_2, \dots, \pi_K, \theta'_1, \theta'_2, \dots, \theta'_K)^T$ is the vector of all parameters.

Now, if $f(y|x, \theta_k)$ is a multivariate normal, while $x \equiv 1$, then this mixture of models is done not for a regression, but for a detection of clusters through the so-called *model-based clustering*.

From probability theory, here is the posterior probability that an observation of the form of (x, y) , belongs to a particular group j :

$$Pr(j|x, y, \psi) = \frac{\pi_j f(y|x, \theta_j)}{\sum_{k=1}^K \pi_k f(y|x, \theta_k)} \quad (b)$$

And this is used to separate the classes as the clusters are chosen based on the maximum posterior probability.

The way this algorithm estimates the optimal parameters is by maximizing a *log-likelihood*. Assuming that our sample has $i = 1, 2, \dots, N$ observations, then the *log-likelihood* that we want to maximize, is:

$$\log(L) = \sum_{i=1}^N \log[h(y_i|x_i, \psi)] \stackrel{(a)}{=} \sum_{i=1}^N \log \left[\sum_{k=1}^K \pi_k f(y_i|x_i, \theta_k) \right]$$

For this package, the maximization is done through the *Expectation-Maximization (EM)* algorithm and thus is comprised of the following two steps:



1. The **E-Step**, or the first step of the *EM* algorithm estimates the posterior class probabilities for each observation:

$$\hat{p}_{ik} = Pr(k|x_i, y_i, \hat{\psi})$$

by using equation (b) from above and then calculating class probabilities like so:

$$\hat{\pi}_k = \frac{1}{N} \sum_{i=1}^N \hat{p}_{ik}$$

2. After that, follows the **M-Step** of the *EM* algorithm, which maximizes each component's *log-likelihood*, using as weights the posterior probabilities calculated above. So, in other words:

$$\max_{\theta_k} \sum_{i=1}^N \hat{p}_{ik} \log [f(y_i|x_i, \theta_k)]$$

Those *EM* steps, that were described above, repeat up until the log-likelihood goes under a pre-specified threshold, or in the case it diverges, if this *log-likelihood* reaches a maximum number of iterations.

This procedure has two drawbacks. Firstly, it can be slow for a large number of data, or a large number of repetitions. And secondly, it can get stuck on a local maximum for quite some time and never reach a global maximum, thus making our solution not as accurate as we would hope it would be.



2.2.3 Bayesian Clustering through Finite Mixture Models (BayesBinMix)

This is the bayesian approach to clustering binary data through the use of finite mixture models. Here is how it works (P. Papastamoulis and M. Rattray, 2017):

Let us assume that our data are contained in a matrix X_{ij} , with $i = 1, 2, \dots, N$ being the total number of observations, while $j = 1, 2, \dots, d$ the total number of variables. Also consider that all these observed values come from a mixture of independent Bernoulli distributions. Then if we take a random sample of these data, like x_{ij} , for $i = 1, 2, \dots, n$ and $j = 1, 2, \dots, d$, then:

$$x_i \sim \sum_{k=1}^K p_k \prod_{j=1}^d f(x_{ij}; \theta_{kj}) = \sum_{k=1}^K p_k \prod_{j=1}^d \theta_{kj}^{(x_{ij})} (1 - \theta_{kj})^{(1-x_{ij})} \mathbb{I}_{(0,1)}(x_{ij})$$

where:

- $\theta_{kj} \in \Theta = (0, 1)$ represents the probability of success for the k -th cluster and the j -th variable, with $k = 1, 2, \dots, K$, where the total number of optimal clusters is unknown, but finite. Thus, we could set a K_{max} and estimate the number of optimal K .
- $\mathbf{p} = (p_1, p_2, \dots, p_K) \in \mathcal{P}_{K-1}$ is the vector of weights attributed to the k -th cluster, where $\mathcal{P}_{K-1} = \{p_k; k = 1, 2, \dots, K - 1: 0 \leq p_k \leq 1; 0 \leq p_K = 1 - \sum_{k=1}^{K-1} p_k\}$ from probability theory.
- $\mathbb{I}_A(\cdot)$ is an indicator function of a subset A .

Should we further consider that observation i comes from mixture component z_i and that $i = 1, 2, \dots, n$ of those allocation variables go unobserved, so that:

$$Pr(z_i = k | \mathbf{p}, K) = p_k, \quad \text{with: } z_i \in \mathcal{Z}_K = \{1, 2, \dots, K\}$$

Then the complete likelihood of this model can be written as:

$$L_K^C = \prod_{k=1}^K p_k^{(n_k)} \prod_{j=1}^d \theta_{kj}^{(s_{kj})} (1 - \theta_{kj})^{(n_k - s_{kj})}, \quad (\mathbf{p}, \boldsymbol{\theta}) \in \mathcal{P}_{K-1} \times \Theta^{Kd}$$

with:

$$\left. \begin{aligned} \bullet n_k &= \sum_{i=1}^n \mathbb{I}(z_i = k) \\ \bullet s_{kj} &= \sum_{i=1}^n \mathbb{I}(z_i = k) x_{ij} \end{aligned} \right\} \text{for a given } (\mathbf{x}, \mathbf{z}) \in \mathcal{X}^n \times \mathcal{Z}_K^n$$



The great thing about the Bayesian approach is that prior distributions can be applied on the unknown parameters. In this case, these are the priors assigned:

$$\begin{aligned} K &\sim \text{Discrete}\{1, \dots, K_{max}\} \\ p|K &\sim \text{Dirichlet}(\gamma_1, \dots, \gamma_K) \\ \theta_{kj}|K &\sim \text{Beta}(a, \beta) \end{aligned}$$

Which are considered independent for $k = 1, 2, \dots, K$ and $j = 1, 2, \dots, d$. The *Discrete* distribution posed in the prior above can either be a *Uniform*, or a *Poisson* distribution with $\lambda = 1$. As a default, the algorithm assumes that $\gamma_1 = \dots = \gamma_K = \gamma > 0$, so as to not to divide the components based on any particular prior information.

Let us define two subsets:

$$\begin{aligned} A_1^{[i]} &= \{j = 1, \dots, d: x_{ij} = 1\} \\ A_0^{[i]} &= \{j = 1, \dots, d: x_{ij} = 0\} \end{aligned}$$

Here is the way the allocation procedure works:

After proposing an initial state for K and z , let it be $\{K^{(0)}, z^{(0)}\}$ repeat the following steps for a number of $t = 1, 2, \dots$ iterations:

1. For $i = 1, 2, \dots, n$:
 - a. Calculate $n_k^{[i]} = \sum_{h \neq i} \mathbb{I}(z_h = k)$ and $s_{kj}^{[i]} = \sum_{h \neq i} \mathbb{I}(z_h = k)x_{hj}$ for $k = 1, 2, \dots, K^{(t)}$ and $j = 1, 2, \dots, d$. With:

$$z_h = \begin{cases} z_h^{(t)}, & \text{for } h < i \\ z_h^{(t-1)}, & \text{for } h > i \end{cases}$$

- b. Update $z_i^{(t)}|z_{[-i]}, \dots$, according to the equation depicted right below:

$$\text{Pr}(z_i = k|z_{[-i]}, K, \mathbf{x}) \propto \frac{n_k^{[i]} + \gamma_k}{(\alpha + \beta + n_k^{[i]})^d} \prod_{j \in A_1^{[i]}} (a + s_{kj}^{[i]}) \prod_{j \in A_0^{[i]}} (\beta + n_k - s_{kj}^{[i]})$$

with $z_{[-i]}$ being the allocation variable vector that is missing the i -th variable, meaning that: $z_{[-i]} = \{z_1, \dots, z_{i-1}, z_{i+1}, \dots, z_n\}$.

2. Suggest updating *Metropolis-Hastings* moves to $z^{(t)}$ in the following manner:
 - **Move 1:** After selecting two mixture components, suggest a random redistribution of the assigned observations.



- **Move 2:** After selecting two mixture components, suggest a random move of the selected subset of observations from the first to the second one.
 - **Move 3:** After selecting two mixture components, suggest to move the assigned observations based on the full conditional probabilities of those that have already been processed.
3. With the purpose of updating $\{K^{(t)}, z^{(t)}\}$, suggest an Absorption/Ejection move in the following manner (supposing that the current state is $\{K, \mathbf{z}\}$):
- a. An ejection is attempted with probability p_K^e , where:

$$p_K^e = \begin{cases} 1, & \text{for } K = 1 \\ \frac{1}{2}, & \text{for } K = 2, \dots, K_{max} - 1 \\ 0, & \text{for } K = K_{max} \end{cases}$$

If not, then an absorption is attempted.

- b. Assuming that an ejection could occur, let the candidate state be: $\{K', \mathbf{z}'\}$, with $K' = K + 1$.
- Suggest a redistribution of observations that were assigned to the ejecting component between itself, and the ejected component based on the $Beta(\tilde{\alpha}, \tilde{\alpha})$ distribution.
 - Accept the candidate state with a probability of $\min\{1, R\}$, with:

$$R = R(\tilde{\alpha}) = \frac{f(K', \mathbf{z}' | \mathbf{x}) Pr(\{K', \mathbf{z}'\} \rightarrow \{K, \mathbf{z}\})}{f(K, \mathbf{z} | \mathbf{x}) Pr(\{K, \mathbf{z}\} \rightarrow \{K', \mathbf{z}'\})}$$

The parameter $\tilde{\alpha}$ is picked to be sufficiently large in probability so as to eject empty components.

- c. Should on the other hand, an absorption be attempted, then:
- All the assigned observations of the absorbed component shall be reallocated to the absorbing component.
 - The candidate state shall be accepted with a probability of $\min\{1, 1/R(\tilde{\alpha})\}$.

Finally, it should be mentioned that this package uses Metropolis-coupled MCMC sampler in order to improve mixing, while also running parallel, “heated” chains.



2.2.4 Indices for Comparison of Partitions

There are indices for cluster evaluation, which can measure how similar two partitions are, when compared to each other. Presented below is the way they work (J. M. Santos and M. J. Embrechts, 2009; M. Meila, 2007; L. Hubert and P. Arabie, 1985):

Assuming that there is a set of objects $S = \{O_1, O_2, \dots, O_n\}$ and this set is parted in two different ways: $U = \{u_1, u_2, \dots, u_R\}$ and $V = \{v_1, v_2, \dots, v_C\}$, such that:

$$\left. \begin{array}{l} \bigcup_{i=1}^R u_i = \bigcup_{j=1}^C v_j = S \\ u_i \cap u_{i'} = v_j \cap v_{j'} = \emptyset \end{array} \right\} \text{for } 1 \leq i \neq i' \leq R \text{ and } 1 \leq j \neq j' \leq C$$

Then the contingency table between these two partitions will be:

Partition		V				
	Group	v_1	v_2	...	v_C	Total
U	u_1	t_{11}	t_{12}	...	t_{1C}	$t_{1\cdot}$
	u_2	t_{21}	t_{22}	...	t_{2C}	$t_{2\cdot}$
	\vdots	\vdots	\vdots	\ddots	\vdots	\vdots
	u_R	t_{R1}	t_{R2}	...	t_{RC}	$t_{R\cdot}$
Total		$t_{\cdot 1}$	$t_{\cdot 2}$...	$t_{\cdot C}$	$t_{\cdot\cdot} = n$

Table 4: $R \times C$ Contingency table between two partitions (objects) U and V .

Here t_{ij} is the number of objects in the i -th subset of R , and j -th subset of C . Let us now define the four following quantities:

- a : Are the objects in a pair, which were placed in the same group, in both U and V . And therefore:

$$a = \sum_{i=1}^R \sum_{j=1}^C \binom{t_{ij}}{2} = \frac{\sum_{i=1}^R \sum_{j=1}^C (t_{ij})^2 - n}{2}$$

- b : Are the objects which were in a pair, that got placed in the same group in U , but in different groups in V , with:

$$b = \sum_{i=1}^R \binom{t_{i\cdot}}{2} - a = \frac{\sum_{i=1}^R (t_{i\cdot})^2 - \sum_{i=1}^R \sum_{j=1}^C (t_{ij})^2}{2}$$

- c : Are the paired objects that got into different groupings in U , but the same groupings in V , which would be:



$$c = \sum_{j=1}^c \binom{t_j}{2} - a = \frac{\sum_{j=1}^c (t_j)^2 - \sum_{i=1}^R \sum_{j=1}^c (t_{ij})^2}{2}$$

- d : Are the objects in a pair that got into different groups in both U and V . And thus:

$$d = \binom{n}{2} - a - b - c = \frac{\sum_{i=1}^R \sum_{j=1}^c (t_{ij})^2 + n^2 - \sum_{i=1}^R (t_i)^2 - \sum_{j=1}^c (t_j)^2}{2}$$

Having these four quantities, *Table 4* can degenerate into a 2×2 contingency table like the one right below:

Partition	V		Total
U	Pair in same group	Pair in different groups	
Pair in same group	a	b	$a + b$
Pair in different groups	c	d	$c + d$
Total	$a + c$	$b + d$	$n = a + b + c + d$

Table 5: 2×2 Contingency table between two partitions (objects) U and V .

Having this table (*Table 5*), the indices that perform clustering evaluation upon how similar they are, are calculated in the following manner (J. M. Santos and M. J. Embrechts, 2009; M. Meila, 2007; B. S. Everitt, S. Landau, M. Leese and D. Stahl, 2011):

- $Rand\ Index = \frac{a+d}{a+b+c+d}$
- $Jaccard\ Index = \frac{a}{a+b+c}$
- $Fowlkes - Mallow\ Index = \frac{a}{\sqrt{(a+b)(a+c)}}$
- $Adjusted\ Rand\ Index = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c) + (c+d)(b+d)]}{\binom{n}{2}^2 - [(a+b)(a+c) + (c+d)(b+d)]}$



2.2.5 Indices for Detecting Clusters through Hierarchical Clustering Methods

One of the toughest decisions is to choose the optimal number of clusters for binary data, especially if it is through Hierarchical Clustering, which mostly provides a general view of how the data are grouped. Plus, every method used may depict a different structure altogether and hence it is quite tricky to detect the best number of clusters.

In a study done by E. Dimitriadou, S. Dolnicar and A. Weingessel (E. Dimitriadou, S. Dolnicar and A. Weingessel, 2002), it was determined that some indices were better than others and therefore could detect the correct number of clusters within the data more reliably. Here the following indices shall be used (D. Ratkowsky, G. Lance, 1978; H. P. Friedman and J. Rubin, 1967; D. L. Davies and D. W. Bouldin, 1979):

- *Ratkowsky – Lance Index* = $mean(\sqrt{varSSB/varSST})$

where:

- *varSSB* is the sum of squares between clusters for each variable.
- *varSST* is the total sum of squares for each variable.

This method estimates the optimal number of clusters by taking the maximum difference to the cluster at its right side, where the curve's decrease is maximized or in other words: $max_k(i_k - i_{k+1})$, with k being the number of clusters and i_k its index value.

- *Friedman – Rubin Index* = $TraceW^{(-1)}B$

where:

- W is the sum of scatter matrices in each cluster.
- B is the scatter matrix of the cluster centers.

This method estimates the optimal number of clusters by taking the maximum difference to the cluster at its left side, where the curve's increase is maximized or in other words: $max_k(i_k - i_{k-1})$, with k being the number of clusters and i_k its index value.

- *Davies – Bouldin Index* = $R = (\sum_{i=1}^n R_i)/n$

where:

- R_i is the maximum value of R_{ij} , where: $R_{ij} = (SSW_i + SSW_j)/DC_{ij}$, with SSW_i and SSW_j being the sums of squares within clusters i and j respectively, while DC_{ij} symbolizing the distance between clusters i and j .



This method estimates the optimal number of clusters by taking the minimum value of second differences, or in other words: $\min_k [(i_{k+1} - i_k) - (i_k - i_{k-1})]$, with k being the number of clusters and i_k its index value.

The *Ratkowsky-Lance Index* was characterized as the most reliable index overall, except for the case of profile identification, where the other two indices (*Friedman-Rubin Index* and *Davies - Bouldin Index*) were considered to be the most adequate.



2.3 Web Scraping

Many websites contain a large amount of invaluable data, like pictures, stock prices, sports statistics, or just plain text. In order to gather all those data, a user could copy and paste the object of interest, but what would happen if it was not just one single piece of text, but multiple entries that are contained in many different sections of a webpage? It would be very tedious and time consuming to gather all the relevant pieces of information by hand. What a user could do instead, is use a web scraping algorithm to achieve his goals.

Web scraping is a technique, which automatically gathers data from the internet, by using the hierarchical structure of a webpage and then exports the results into a format that is more useful to the user. However, since the webpages come in all shapes and sizes, so do the algorithms that perform web scraping. In general, a webpage is comprised of different languages, such as HTML, CSS, JavaScript, PHP, etc. that are interlinked within its source code and have it respond dynamically to whatever a user does (like click, or point, or hover over an object with a mouse, or perhaps type a string of characters using a keyboard, etc.). Therefore, the goal of web scraping, is to utilize this unique structure and get the data that a user desires. There are of course many web scrapers available online, however for the needs of this thesis, what is needed is a custom-made web scraper that will produce the desirable results.

This whole procedure could be done in any of the popular languages, but as R was used since the beginning, so it was used for this part as well. It really helps that it had some of the more user-friendly libraries, which will be discussed below, along with other tools that were used to make this project a reality.

Docker:

This is a tool that is used for application deployment either through the cloud, or locally on a personal computer. What the web scraping program needs is a host (or a server) to run the library called RSelenium on a local machine, by deploying a Docker image of the web browser that is going to be used and then connecting that image with our code and its dependencies. If RSelenium was not chosen as a library, Docker would not even be mentioned, but the alternative was a very slow method.

The rvest library:

This library is commonly used in unison with others (e.g., *polite* library) to perform web scraping. It is used to locate whatever part of the source code a user might need. It can access HTML code hierarchically and use CSS selectors that speed up the whole parsing process.

The pipe operator (`%>%`) is used to access the inner workings of each HTML element. Therefore if anyone wanted to access and read the first paragraph within the `<body>` part of the HTML code, then they would need to write the following code:

```
html %>% html_element("body") %>% html_element("p") %>% html_text()
```



Therefore from the whole html output, this piece of code first accesses the `<body>` part of the HTML code and then directs its attention to the first element of a paragraph (which is symbolized as `<p>` in HTML).

Apart from that, as it was mentioned above, this library also uses CSS selectors. Since most sites use inline CSS code along with HTML code, it becomes easier to skip a few levels in order to reach the piece of information that a user needs, as CSS code is more unique and thus can be instantly recognized by a web scraper. Therefore, if we have many paragraphs within a source code (and thus lots of `<p>`), if these paragraphs have inline CSS code, for instance like:

```
<p id="fourth">
```

Then it is easy to jump to that piece of code by using CSS selectors, as no other paragraph will bear the same id. The four most important ones are the following:

- *p* to select all paragraphs inside HTML. For example: `html_elements("p").`
- *.keyword* to select all *class* elements that start with a keyword. Thus if we have a paragraph with an inline CSS code, like: ``, then we could select it by typing: `html_elements(".sign").`
- *p.special* to select all `<p>` with *class* elements that start with special. Same thing as above, but for paragraphs only.
- *#keyword* to select all *id* elements that start with a keyword. So if we have the following piece of source code `<p id="fourth">` surrounding the information we need, then we can select it in the following manner: `html_element("#fourth").`

The RSelenium library:

This library is using the Selenium Webdriver Application Programming Interface (API), which can help a user automate and perform browser-based actions (like clicking, scrolling, selecting and much more) and is mainly used for testing applications. The only downside in this instance, is that the program must link to the Selenium Server to perform the actions required and that is the reason why Docker is used. But other than that, this Java-based program can perform any action a browser does and thus allowing R users to automate any procedure that they might need to do, which is extremely helpful especially for web scraping applications.



The XML library:

This is another versatile library that can parse and create both XML and HTML source code. It is similar to rvest in its functionality however it goes a bit slower when selecting specific topics from multiple webpages. It uses `htmlParse()`, which generates an R structure representing the XML/ HTML tree, as well as `xpathSApply()`, which searches within a text for the information needed according to the path provided by the user.

Specifically for LinkedIn:

In the case of LinkedIn, one would see that it is a dynamic webpage, meaning that when a user scrolls or clicks on an object, it responds to the action. In the job section, one could clearly think that it is all just one page, but if one explores the source code, then they will see a two-fold hierarchy that uses anchor tags and internal links to each of the job advertisements. Therefore, the main page stores all the links to the jobs, within a certain country, and for the particular choice of words that were used within the search bar and only after we click on the particular job, can we see its description and the particulars of the job on the right-hand side of the page.

However, that is not all. As it was mentioned above, LinkedIn is a dynamic website and that also applies to whenever a user scrolls down the webpage itself. There are parts of source code that remain invisible as long as nothing happens, but as the user scrolls down the webpage, parts of code get loaded and appear within the source code. This too must be done automatically through R in order to get adequate results, as many jobs remain invisible, as long as the source code remains in its original form when a user only opens a webpage without scrolling down and loading most of the available links. In this part of code, this can go up to 150 jobs in total, per country, per term searched, as after this, there is a button prompt to load even more job positions, however it was deemed adequate to stop there so as to reduce the bias in the data.

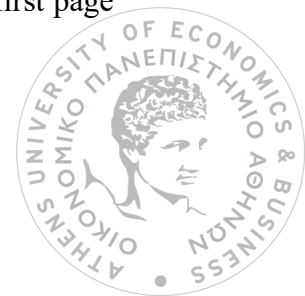
How it works:

Before performing any data scraping, we first install Docker locally on our computer, because it is required for the library RSelenium to work. After we map the container port, to the host port, we will need to pull and load the image of the browser that we will be using (for example Firefox) through the command prompt of our operating system. Once that is done, we can start the web scraping procedure.

- First of all, we construct our url, based on the country and the term to be searched. For instance:

[https://www.linkedin.com/jobs/search?keywords=%22Data Scientist% 22&location=England&position=1&pageNum=0](https://www.linkedin.com/jobs/search?keywords=%22Data%20Scientist%22&location=England&position=1&pageNum=0)

This refers to the position of a “Data Scientist” in England and it directs on the first page of the LinkedIn search engine.



- After that is done, we request a connection with the HTTP server by using the link that was already constructed.
- When we get directed to the first page of LinkedIn, we scroll down 5 times, in order to load the maximum number of job advertisements that we can for the country and the search term that we selected above.
- Afterwards, we open the source code and parse the HTML code in order to get it all in a script back on our computer, by using the XML library.
- From that script, we select only the internal href links that contain the job advertisements.
- After constructing this list of internal, anchor links within the site, we access each and every one of them, in order to parse the job descriptions and characteristics of interest and store them in a huge data frame, which is done by using the library rvest. This library allows us to access the HTML, CSS and JavaScript code hierarchically and get to the script that interests us (in this case the job description, as well as Industry and Seniority Level variables).
- Within that loop a pattern-searching vector is applied to the description of the job which tries to detect words of interest (such as R, Python, SAS, etc.), considering lowercase, as well as uppercase terms that could be included.
- After that procedure is done, the algorithm terminates the server connection to the site and returns the data frame that the user asked.

In conclusion, this algorithm is time-efficient, meaning that it can do a job which manually took 10 days to complete, within a mere 40 minutes, however it is not as smart as a human being, which means that it will not discard many of the repeating jobs that appear in many countries, or ambiguous jobs from recruiting agencies, or search the internet and infer the true seniority level, or the industry to which a job belongs to. But other than that, it does the job adequately.

Two HTTP-based errors:

Since this application is web-based, it means that it can get faulty at times, and especially when a connection cannot be established due to errors in the HyperText Transfer Protocol (HTTP). Two of the most common errors are the following:

- HTTP Error 500: Internal Server Error. It means that the request of accessing a page has been unsuccessful and in order to correct it, it is required to reload the image of the browser used in Docker.



- **HTTP Error 429: Too Many Requests.** This is a way for each site to protect itself from a Distributed Denial-of-Service (DDoS) attack, which, in essence exhausts a website's resources by constantly bombarding it with connection requests. Usually there is a time out, which can last from a couple of minutes to a whole day. Thus, in order to bypass it, one solution would be to wait the period of time, but as time can be pressing, one can change their IP address, by either restarting their router, thus allowing the DHCP to reallocate a new public IP, or by running the following lines of code in the Command Prompt (*cmd*) in order to refresh and switch an old IP address to a new one:

```
netsh winsock reset
netsh int ip reset
ipconfig /release
ipconfig /renew
ipconfig /flushdns
```

[You can find the whole code at: <https://github.com/KonstantinosK88/Web-Scraping>]



CHAPTER 3: COLLECTING THE DATA

3.1 Description of the Dataset and its Variables

3.1.1 Statistical Tools

Considering some of the most popular software today, here is a list of software that were monitored and compared to each other in terms of popularity:

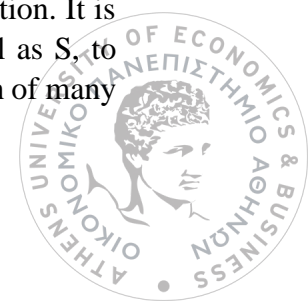
Statistical Programs		
R	Python	SAS
MATLAB	Stata	EViews
Minitab	SPSS	Winbugs/Openbugs
JAGS	Stan	SQL
C/C++	Rust	Tableau
PowerBI	Alteryx	Hadoop
Spark	Scala	SAP
Qlik	Hive	Julia
Spotfire	JMP	VBA
Java	MS Access	MS Excel
MS Azure	AWS	Git
Elasticsearch		KNIME

Table 6: A list of (35) statistical tools (programs, frameworks, etc.) to be used in the analysis.

In total, what we have are 35 programs, platforms, frameworks and software used by statisticians, which are going to be used and compared in order to see which ones are more popular than others. To achieve just that, 1000 observations have been collected from job postings on a website named: *LinkedIn*. It could have easily been one of the other popular sites for job searches, like GlassDoor or Indeed for instance, however LinkedIn was chosen for its ease of access, the way it is built (as HTML code is crucial for web scraping) as well as the fact that it is used by millions of people worldwide. Thus, it constitutes a good platform to conduct this search.

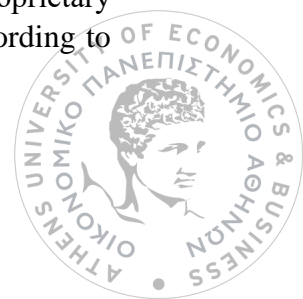
Let us now relay some information about each of the statistical tools mentioned in *Table 6*:

- **R:** Created by John Chambers and his colleagues in Bell Laboratories in 1993, R is a powerful programming language for statistical computing and graphical representation. It is open-source and can incorporate snippets of code from C/C++, Fortran as well as S, to which R is considered to be closely related. Throughout the years the contribution of many



users as well as the easy usage of statistical packages has made it a very reliable tool for any statistician. Today it is available on Linux, Mac and Windows Operating Systems.

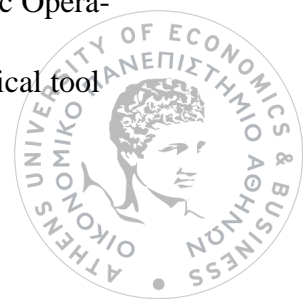
- **Python:** Created by the Guido van Rossum it first released to the public in 1991 under the version of Python 0.9.0. It was formed to become the next big thing after the ABC programming language, which it did, if one considers its success in our time. Not only is this an open-source programming language, but it is also a very convenient tool for both beginners and professionals. Much like R, it can do difficult statistical calculations as well as modelling and the graphical representation of the results or the data, although it was not created with the statistician in mind and therefore sometimes might require a few extra steps to get to the same result as R. Arguably, due to its simplicity it is a strong contender against R, as it can do the same things that R does, although, of course, not everyone agrees. Nevertheless, Python is a powerful programming language, which is used widely in many different fields nowadays (like Programming, Statistics, Gaming, etc.). In 2022 it is available on Linux, Mac, Windows as well as Android Operating Systems.
- **SAS:** Dubbed as Statistical Analysis System, SAS was conceived in SAS Institute in 1972. Its main goal is the management and analysis of data. It provides an intuitive point-and-click graphical user interface (GUI), which does not require any technical knowledge from the user, apart from perhaps knowing what information is to be extracted from the data. It has also implemented an AI algorithm that helps in predictive analysis, as well as machine learning. It is a proprietary software, that has a free trial, but afterwards is billed on an hourly basis (\$0.55/hour) and how much one uses it. It also incorporates the usage of Microsoft Azure as its service is also cloud based. One other thing that should be mentioned is that this language was the starting point for JMP, which is also used in statistics. It is available for Windows, IBM mainframe and Unix/Linux Operating Systems.
- **MATLAB:** Conceived by Cleve Moler and developed by MathWorks MATrix LABoratory, also known as MATLAB got its first commercial release in 1984, but as a programming language its development was done in the late 1970s. MATLAB is a very powerful tool, which can be used for programming, data analysis and of course graphical representation of the data. This is a proprietary programming language that has lots of pricing options (\$2100 for a perpetual license, \$840 for annual leasing) and a 30-day free trial. Mostly it is used by mathematicians, engineers, scientists and economists. MATLAB is available for Windows, Mac and Linux Operating Systems.
- **Stata:** Developed first by William Gould and afterwards by Sean Beckett, this statistical software package saw its first release in 1985. Stata is a statistical software built for all data science needs. After the version 8.0, this tool has incorporated an easy-to-use graphical user interface (GUI) with menus and dialog boxes and can be very versatile when it comes to presenting the user's results, while also providing automated reporting. It is a proprietary statistical software, and its pricing starts from \$840/year and goes upwards according to



the needs of the customer. In 2022, Stata is available for Windows, Mac and Linux Operating Systems.

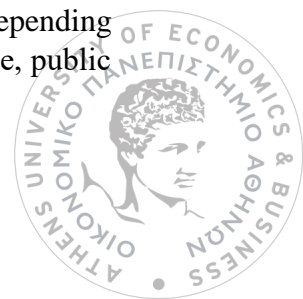
- **EViews:** It was developed by Quantitative Micro Software (QMS) and released its 1.0 version in 1994. This statistical package is mostly aimed at time-series modelling, forecasting and econometric analysis. It is available only for the Windows Operating System. Its object-oriented interface works as a series of menus from which the user selects the analysis that he needs to do. It is a proprietary statistical package and its pricing is on a per version basis (with the current one, version 13, being priced at \$1775 for a single user license).
- **Minitab:** Created in 1972 by Barbara Ryan, Thomas Ryan, Jr., and Brian Joiner at the at the Pennsylvania State University, this statistical package aims at analyzing, visualizing and forecasting based on the data provided by the user. As of 2020 it has migrated its services to the cloud and it is available on the Windows Operating System only, in 2022. It has a menu-based user interface that allows the user to select what he wants to do with the data. It is a proprietary statistical package that has a starting price of \$1610 and then other pricing options according to a user's needs.
- **SPSS:** Originally developed by Norman H. Nie, Dale H. Bent, C. Hadlai Hull at IBM, this statistical software suite saw its first release in 1968. SPSS stands for Statistical Package for the Social Sciences, which was its audience in that era, but then it changed to Statistical Product and Service Solutions as its audience grew. It supports versions for Windows, Mac and Linux Operating Systems. It has extensions that incorporate R, Python and even SQL. The users can either access its many features through a series of pull-down menus, or program their own structures into it through a proprietary command line. It is a proprietary software suite that is owned by IBM. It is priced at a subscription based model and IBM Statistics starts at \$99/month for each user.
- **WinBUGS/OpenBUGS:** Both of these are statistical software that are used for Bayesian analysis. WinBUGS was developed by the BUGS Project at Cambridge and it saw its first release in 1997. The thing is, that it has stopped updating its versions since 2007, with 1.4.3 being its last stable version. OpenBUGS is as efficient and reliable as Win-BUGS, with a few differences. It works well with R, as it was designed to run on S-Plus. This is an open-source statistical application and it can run on Windows, Mac and Linux Operating Systems.
- **JAGS:** Just Another Gibbs Sampler, or JAGS is once again a statistical application that is used for Bayesian analysis, much like WinBUGS/OpenBUGS, however, it is arguably a bit more intuitive than resolution when encountering bugs in the program. It was developed in 2007 by DIA Bayesian Scientific Working Group (BSWG) and was written in C++. This is an open-source application, which has versions on both Windows and Mac Operating Systems.

It should be mentioned that although JAGS was monitored as a possible statistical tool



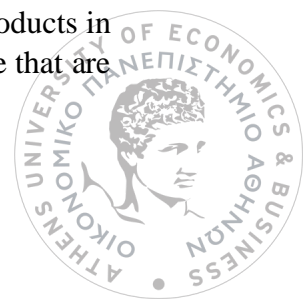
for analysis, in the end it was excluded as it had not been observed even once in the data that was gathered.

- **Stan:** This is a programming language used for statistical inference, that was named after Stanislaw Ulam. Its initial release was in 2012. This language can be accessed primarily through R, but also Python, MATLAB, Julia and Stata. It is an open-source language and it can be used on Linux, Mac and Windows Operating Systems.
- **SQL:** It was developed by Donald Chamberlin and Raymond Boyce and first appeared in 1974. At first it was named SEQUEL (for Structured English Query Language), but then got shortened into SQL (for Structured Query Language). This is a query language that was built to handle data structures within a relational database system. The language itself is open to the public and although most of its clients that use it are open-source (like PostgreSQL, SQLite, or MySQL), some of them are proprietary (Infobright or Scimore-DB). It has all kinds of versions that can run on Linux, Mac and Windows Operating Systems.
- **C/C++:** C was designed by Dennis Ritchie and released in 1972, while C++ by Bjarne Stroustrup as an extension of C, which came to be in 1985. Today everybody knows these two programming languages and although it is not necessarily a requirement to know them in order to do statistical analysis, it is very advantageous, as many languages were written in one of them (like R for instance was written in C). Therefore, it can make calculations a lot faster if one applies code straight from C/C++ into their routine, which is really helpful, especially when time is of the essence. Apart from building some languages and software, C/C++ are significant in the programming world and have influenced many other software, such as Java, Python, Rust, etc. These languages are open to the public for implementation and they have been known to have cross-platform applications, such as Linux, Mac and Windows, etc.
- **Rust:** It is a high-performance programming language which was built by Graydon Hoare in 2010 as a general-purpose language and it is mostly used for programming systems. It constitutes an alternative to C/C++ and Java in that aspect. It can be used for statistics and data science as well, although does not yet have the versatility of its competitors (such as the Scikit library of Python). Rust is open for implementation and can run on all platforms, like Linux, Mac and Windows, etc.
- **Tableau:** It was founded by Pat Hanrahan, Christian Chabot, and Chris Stolte in 2003. Tableau is a platform that allows for visualizations of data through dragging and dropping items in its environment, which does not require programming knowledge from the user. This tool is mostly used in data analysis and can create complex graphs in an easy and rapid fashion. It is a proprietary platform that is billed on a monthly subscription basis depending on the users needs its “Creator” license costs \$70/month, while also having a free, public



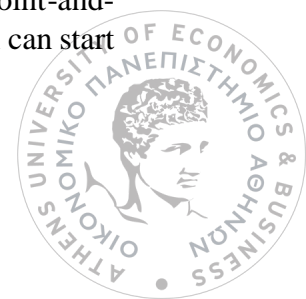
version with limited capabilities. Tableau is available for Mac and Windows Operating Systems.

- **PowerBI:** Conceived by Thierry D'Hers and Amir Netz, this platform first released in 2011. It is a data visualization tool, much like Tableau, which allows the user through a series of menus, as well as dragging and dropping, to visualize the data and create dashboards. It is mostly used in data presentation and analysis, and what is best is that it does not require any programming skills from the user. This is a proprietary platform and its “Pro” edition is on a monthly based subscription, costing \$9.99/month, while it also has a free and limited version of its software. It is available on Windows as well as mobile devices.
- **Alteryx:** Created in 2006, this is an environment for building analytical processes. This tool automates the procedure of processing and visualizing the data, which makes it fast and easy to use. It can run predictive modelling without having to know any code, but just by dragging and dropping the specific functions that a user shall utilize. It is a proprietary platform and it costs an individual user \$5195/year. It runs on Windows and can also run on Mac via a VM.
- **Hadoop:** This is an open-source framework mainly used for processing and analyzing big data. This collection of software has been written in Java. It uses cloud-based technologies, such as computer clustering to expedite and augment its processing power. It is mainly used for analytics that draw information from big data, and even supporting machine learning applications. Hadoop’s ecosystem includes Hive and Spark apart from other software as well. It runs on most operating systems, including Linux, Mac and Windows.
- **Spark:** This is a unified analytics engine that processes large scale data sets. It was developed in the University of California and later donated into the Apache Software foundation. Its first version came out in 2014. What Spark does is it solves a hardware problem that might make an SQL query run slow, or not having enough space on the machine used to process the large set of data. It interfaces with other frameworks and programs to achieve its goal, such as Hadoop. It is open-source and can run on multiple platforms, including Linux, Mac and Windows Operating Systems.
- **Scala:** Is an object-oriented, high-level programming language designed by Martin Odersky, with its first public release being in 2004. It offers interoperability with Java both with the code and its libraries. Statisticians use it mostly for big data processing and machine learning procedures. This is an open-source project and it can easily run on Linux, Mac and Windows Operating Systems.
- **SAP:** This acronym is in German and it stands for Systems, Applications, and Products in Data Processing. It is an umbrella of different frameworks, products and software that are

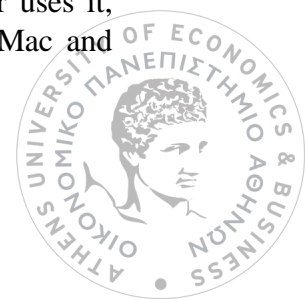


used in running a whole business. The company itself was founded in 1972. Today it can even be used in data science through its SAP Analytics Cloud SaaS, where the user is guided through a series of menus to manipulate the data as he pleases. All of its products are proprietary products and the subscription is based on what the user needs (for instance SAP Analytics Cloud is priced at \$36/month). It can work on Linux, Mac and Windows Operating Systems.

- **Qlik:** This is a business and big data analytics platform. The company was founded in 1993. It allows its users to do big data analysis and create interactive dashboards fast and easily through point-and-click menus. Therefore, no coding skills are necessary. One of its most used tool is Qlik Sense, which a cloud-based analytics solution. This is a proprietary platform and its pricing starts at \$30/month. Originally it runs on Windows OS but can also do so on Mac through the use of a Virtual Machine.
- **Hive:** This piece of software was built on top of Hadoop to aid it in data query and analysis, but today it is a standalone project. Its first version came out in 2010. It works in an SQL-type of manner, meaning that it can provide queries in a similar fashion. Hive was written in Java. It is open-source and can run on most operating systems, including Linux, Mac and Windows.
- **Julia:** Designed by Jeff Bezanson, Alan Edelman, Stefan Karpinski and Viral B. Shah, it initially released in 2012. It is a dynamic, general-purpose programming language. Today it is used in data analysis and machine learning, while also providing data visualization. It is extremely fast, because its compiler (JIT compiler) compiles high level-code into machine language before running it. It is a very versatile programming language and can work with multiple other languages and software such as Python, R, Rust, C++, SQL, etc. It is an open-source project and can run on Linux, Mac and Windows Operating Systems.
- **Spotfire:** Initially it was built by Christopher Ahlberg and Ben Schneiderman in the early 1990s and later was purchased by TIBCO in 2007. Spotfire is an artificial intelligence platform, that can analyze and visualize the data that the user provides through a point-and-click menu. While much like Alteryx, it can create functions and connections between them through dragging and dropping the items that a user selects. It can also work in unison with R, Python, MATLAB, KNIME, SAS, etc. as engines that are behind these data functions. It is a proprietary platform that is based on a monthly subscription, which can start as low as \$25/month. It is developed for Windows OS, but can also run on Mac and Linux through a Virtual machine software.
- **JMP:** This is a suite first developed by John Sall in the late 1980s for statistical analysis. It can perform a plethora of functions such as hypothesis testing, data exploration and visualization, machine learning, predictive analytics, etc., through a series of point-and-click menus, much like SPSS. JMP is a proprietary suite of statistical tools, which can start at \$1200/year per user. It can run on Mac and Windows Operating Systems.



- **VBA:** Visual Basic for Applications is an event-driven programming language built into most of Microsoft Office applications. It first appeared in 1993. Mostly, it is very hard and unintuitive to perform statistical analysis in this language, but it combines with Excel and can create macros that aid analysis and visualizations, which make Excel much more powerful in that aspect. It is a proprietary programming language and is included in the Microsoft Office Suite, which costs \$439.99 for the 2021 version. It runs on the Windows Operating System.
- **Java:** Designed by James Gosling, this object-oriented programming language first appeared in 1995 its revolutionary idea was to run on all possible platforms without the need to recompile. Today many of the big data frameworks use it or are written in it, such as Spark, Hadoop, Hive, ElasticSearch, etc. But even on its own it is perfectly capable of performing data analysis, machine learning, deep learning and data visualization. It is an open-source language and can run on almost any system imaginable.
- **MS Access:** This is a database management system, which was created by Microsoft in 1992. It was written in C++ and is now a part of Microsoft Suite. It is used to create and manage databases, much like SQL, while it also supports VBA programming. It is a proprietary system and its standalone cost is \$159.99. MS Access operates on the Windows Operating System.
- **MS Excel:** Was first developed by Microsoft in 1987. It is a spreadsheet made to contain data. After many years of development it has become a very versatile program and now can be also used for data analysis through the use of macros and visualizations. That being said, it is not suitable for big data, as every sheet has a maximum number of rows. MS Excel is a proprietary tool and its standalone cost is \$159.99. It can run on Windows, Mac, Android and iOS Operating Systems.
- **MS Azure:** This is a cloud-based platform first developed by Microsoft in 2008. It supports many software, programs and tools both from Microsoft and other vendors. Apart from providing management and cloud-based storage, it has the capability of implementing machine learning models right through it, but also be used for analysis. This is a proprietary platform and its subscription varies based on what the client needs (Advanced analytics on Big Data is currently \$273.36/month). It can be used by Linux, Windows, iOS and Android.
- **AWS:** Much like MS Azure, this too is a cloud-based platform. It was developed by Amazon in early 2000s. It provides services like its competitor, meaning machine learning and AI based services, as well as the rudimentary data storage and management through a cloud-based system. It is a proprietary tool, which is billed as much as a user uses it, depending on the programs that they choose to utilize. It can run on Linux, Mac and Windows.



- **Git:** This is a version control system, meaning that it is a way to monitor any changes that might have happened to a project, or a file. This is very useful especially if there are constant changes to the work that is done and can refer to a specific version of a program that is being used or updated. As algorithms and frameworks are being developed by a lot of employees that work in parallel, this software becomes rudimentary to help these people work in unison. Git is open-source and can run on Linux, Mac and Windows Operating Systems.
- **ElasticSearch:** This is NoSQL JSON-based data storage, search and analytics engine that was created by Shay Banon in 2010. It supports clients in Java and Python, among other languages. It can perform machine learning processes (both supervised and unsupervised) through Elastic Stack but can also operate as an analytics platform from user-written queries. It is not as powerful as PowerBI for example, but can provide an insight into the data to be used. It is a proprietary engine and its pricing starts from \$95/month. It can run on Linux, Mac and Windows Operating Systems.
- **KNIME:** Konstanz Information Miner is a platform for data analysis that can perform modelling, and visualization of the data without a lot of programming knowledge. It was created in the University of Konstanz in 2004. It is written in Java and it is a free and open-source platform, which can be run on Linux, Mac and Windows Operating Systems.



3.1.2 Other (Grouping) Variables:

Apart from the statistical software aforementioned, some variables of possible interest have also been noted for each of those job advertisements that were positioned into columns after the initial 34 (without JAGS).

One such column, was the “*Business Name*” column. Here registered, as the name suggests, are the companies that were observed in our dataset. Upon collection, the aim was of course to have as many uniquely named enterprises as possible. That of course was harder to achieve as many of those firms were international. Therefore, the strategy was that if a company had the same position in another country or had multiple job positions under the same terms that were being searched, then this job was ignored. Hence, in this dataset, companies of the same name have different job positions.

After the previous column, followed the column named “*Industry*”. This is a column that classifies the businesses that were previously observed in the “*Business Name*” column into their respective sectors. The goal here was to make broad categories that would each include as many observations as possible. At the end of the collection of our data, these were the groups formed within this column:

Industries		
Accounting	Advertising	Automotive
Aviation	Banking	Construction
Consulting	Education	Energy
Engineering	Environmental	Farming
Finance	Food & Beverages	Gambling
Gaming	Government Administration	Health
Hospitality	Insurance	IT
Logistics	Manufacturing	Music
Networking	Non-Profit	Publishing
Real Estate	Recruiting	Research
Retail & Wholesale	Software	Telecommunications
Tourism	Transportation	Utilities
	Entertainment	

Table 7: A list of (37) industries that were observed in our data set.

After the collection and careful research, we ended up with 37 categories of industry in total, as far as this dataset is concerned.

Then, there was the column which was named “*Seniority Level*”. It had five levels in total, that were based on the experience needed to successfully apply for the job. Specifically, these levels were categorized in the following manner:



Seniority Level	Years of Experience
Entry - Level	0 years of experience
Associate	1-2 years of experience
Junior	2-3 years of experience
Mid - Level	3-5 years of experience
Senior	5+ years of experience

Table 8: A list of (5) levels inside the “*Seniority Level*” variable; and how it is categorized.

At times the job description would mention the minimum amount of experience needed to apply for the position. When it specified the level itself (e.g., Senior Data Analyst) without providing the years of experience in the description, then this was the category that the position was attributed to. However, should the position level which was assigned by the company differ from the years of experience needed in the description, then the levels discussed above superseded the name in the title (e.g., a Junior Data Scientist that needed at least one year of experience was assigned to the Associate level). And finally, if the experience level was vague, but necessary, then the lowest level of experience (meaning that of the Associate) was assumed to be needed in order to apply.

Next was the “*Country*” column, which as the name suggests, contains the countries where the observed jobs may be found. This is the list of the total number of countries used:

Countries		
Denmark	Estonia	Finland
France	Germany	Iceland
Ireland	Latvia	Lithuania
Norway	Poland	Sweden
	United Kingdom	

Table 9: A list of (13) countries used in our dataset.

After the completion of our dataset, the list included 13 countries from the northern Europe, with countries ranging from France to Estonia and going as up as Iceland itself.

“*Date Collected*” was the column after that and it contained the actual date upon which the data was collected. The whole dataset was collected in a span of 10 days, but as there were observations that did not include any of the observed statistical software, an extra 9 observations were added at a later date and replaced these uninformative data points.

The last column is named “*Term Searched*” and it contains all the popular terms that were used to search for a job that a statistician could do. It contained the following three groups:



Term Searched
Data Analyst
Data Scientist
Statistician/Biostatistician

Table 10: A list of (3) terms that were used during the search for our data.

Another term that was considered was the role of an Actuary, but as the research found only one such mention in a total of the five initial countries, it was eventually dropped.

The dataset described above would therefore take each job description and catalogue all the data that were pertinent to it, meaning the country, the industry, the terms searched and so on. Should therefore one of the 35 (JAGS is still included during the monitoring process) monitored pieces of software were to be observed in the job description of the advertisement, then this software would receive the value 1 in our dataset, otherwise it would remain at 0. That being said, the tracked software must have been mentioned explicitly in the job description. For example, should the job advertisement ask for candidates who know the Microsoft Suite, then this would not be an automatic point to Access and Excel (that are definitely a part of it), as the suite itself contains 5 other, additional programs, and therefore it should not be assumed that they mean one of those 35 that are being monitored by us (although knowing what the other programs in MS Suite do, it is highly probable that they do ask for either Access or Excel).

Thus, in summary, what this dataset has, are programs that were asked specifically by name in the job description from the potential applicant for the vacant position.



3.2 Observations Upon the Collected Data

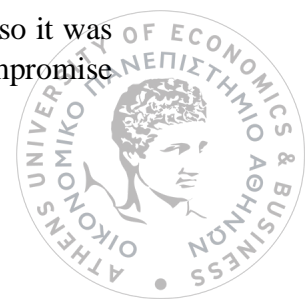
The data has been collected in the fashion that was described above. However, before the discussion is moved towards the analysis part, there are some things that must be addressed that concern this dataset and its idiosyncrasies.

This set has no jobs that do not have at least one of the 35 programs, platforms, frameworks or software used by statisticians that we wanted to monitor. Technically speaking, there are no rows among the 1000 that were observed for which the first 35 columns sum to zero. So one might ask, where there really no job advertisements for statisticians or data scientist that did not have any of those specific software that we were trying to look out for? Of course there were. There were positions that seemed very vague when they described what kind of skills were needed. Some just stated something along the lines of: “Programming knowledge necessary in at least one of the statistical software”, without specifying clearly whether it was, for example R or Python that the candidate needed to know. Other jobs emphasized rules and regulations of which the applicant should have some knowledge of, or perhaps other programming languages, or management software that were unfortunately not included in our dataset, as they had little to do with statistics. Therefore, as these jobs were not just one category, but spanned in multiple sectors, they were removed from the set as they would only spread confusion.

Furthermore, while gathering the data, the column named “Industry” had to be filled with the proper sector, to which each job belonged to. The goal was to get the general purpose of each company. This task was not as easy as it sounds as many firms had very vague descriptions about what they actually provide to the public and instead used power words, such as: “innovative” or “cutting-edge” and so on. Other companies belonged to more than one sector. One such example is Caverion, a Finnish firm that specializes in both construction and maintenance of buildings. Here, as this would be confusing to our dataset and so as to not have too many individual and unique categories, it was decided to get companies such as these under one common, broader term. Caverion for instance was put under the group named Construction, because that is what it partially does. Having this in mind, the groups are a bit more general than they let on.

LinkedIn, for those who do not know, has a field where a company may specify the seniority level that they want from a candidate. So if a job is addressed to people who are just starting their career, then that field should indicate that it is mentioning a job of Entry-Level experience, or if they need an experienced person, it could be Mid or Senior Level. This is what logic dictates, however, and it is not clear as to why it happens, there were a few jobs that, albeit describing an Entry-Level person wanted five plus years of experience. It may be a tactic to drive the salary down, or just a mistake. And on the other end of the spectrum there were jobs that clearly needed an experienced person and demanded no experience whatsoever. That was why the job classification system that *LinkedIn* uses was mostly ignored, and the column named “*Seniority Level*” was based on its own way of measurement so as to avoid any weird or inaccurate situations.

The dataset started at the northern part of Europe, as far as the countries were concerned. It began in Scandinavia and went downwards. A strange thing that was observed, was that Iceland had only one such job for statisticians and after debate, it was considered prudent to keep this one observation. After all, Iceland is a smaller place when compared to other countries and so it was only natural to lack in particular work positions for statisticians. Plus, it should not compromise



our results too much, as it is expected that a statistician's geographical location should not affect his skillset. After all, companies will strive to have candidate who know the best and latest software so as to be competitive with other firms.

After concluding with the gathering of our data, it was apparent that Bayesian software were getting almost no mentions, or in other words they had the lowest number of jobs that demanded that particular program. But does that mean that they are not used at all? Far from it, however it appears that companies that utilize them, do not care which one of them will a worker choose to use. Mostly they apply a generic: "Must know Bayesian Analysis" in the description and do not care or consider it to be trivial how the statisticians will do it, or which program they will eventually use as long as it gets results. That is also why statistical tools like JAGS that had zero observations were discarded in the end.

One could argue that some of the jobs have a lot of demands when it comes to knowing some of these statistical software. As it seemed in the description some of the firms were actually looking for such an all-knowing individual, however most of them are just suggestions as to what the company might be using and maybe some alternatives that they can work with. Hence, if a row of statistical programs used, has a lot of statistical programs, it does not necessarily mean that this company demands the knowledge of them all, but as far as popularity goes, they are all considered known as that firm chose to mention these particular statistical tools and assumed that the potential applicant will know of them.

Furthermore, at times, it appeared as if the job poster did not know what kind of person they were looking for. One such position mentioned that: "The candidate should at the very least know either SQL, or Python", however any person who has worked with these two computer programs, would attest that they are not interchangeable and can do very different things. At other times advertisements referring to Data Scientists actually appeared to be looking for Data Analysts, as they needed a person to process raw data, instead of a person who would model, or draw inference from these said data. This is perhaps due to the fact that companies are not sure how to divide these two roles, or do not know whether they differ at all.

Lastly, as it was mentioned before, the focus of this thesis was only on the following three job titles:

- Data Analyst
- Data Scientist
- Statistician/Biostatistician

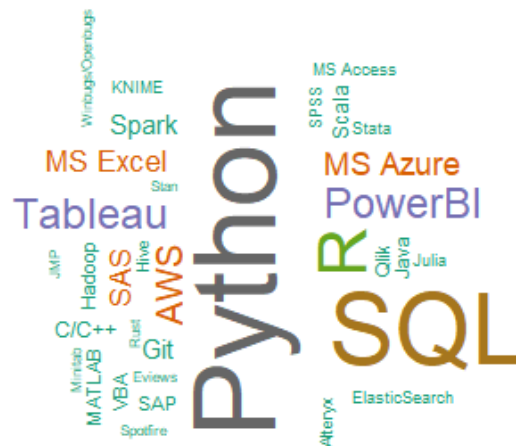
There were, of course, other roles that a statistician could do, such as a Quantitative Analyst, but they were so few and sparse that only the three above were finally selected. The sparsity could be due to the fact that many of these titles were written in their mother tongue and although this thesis did translate a lot of these titles, some of them were of ambiguous meaning, when put through the *Google Translate* software. Thus, much like the title of the Actuary, these other positions were left out of our dataset.



CHAPTER 4: ANALYZING THE DATA

4.1 An Initial Picture (Word Cloud) of Our Data

Now that we have a rough idea what our data shall contain, let us try and visualize what we have collected so far and see if we can surmise anything about the popularity of the statistical software that was monitored. Here is a word cloud of our data:



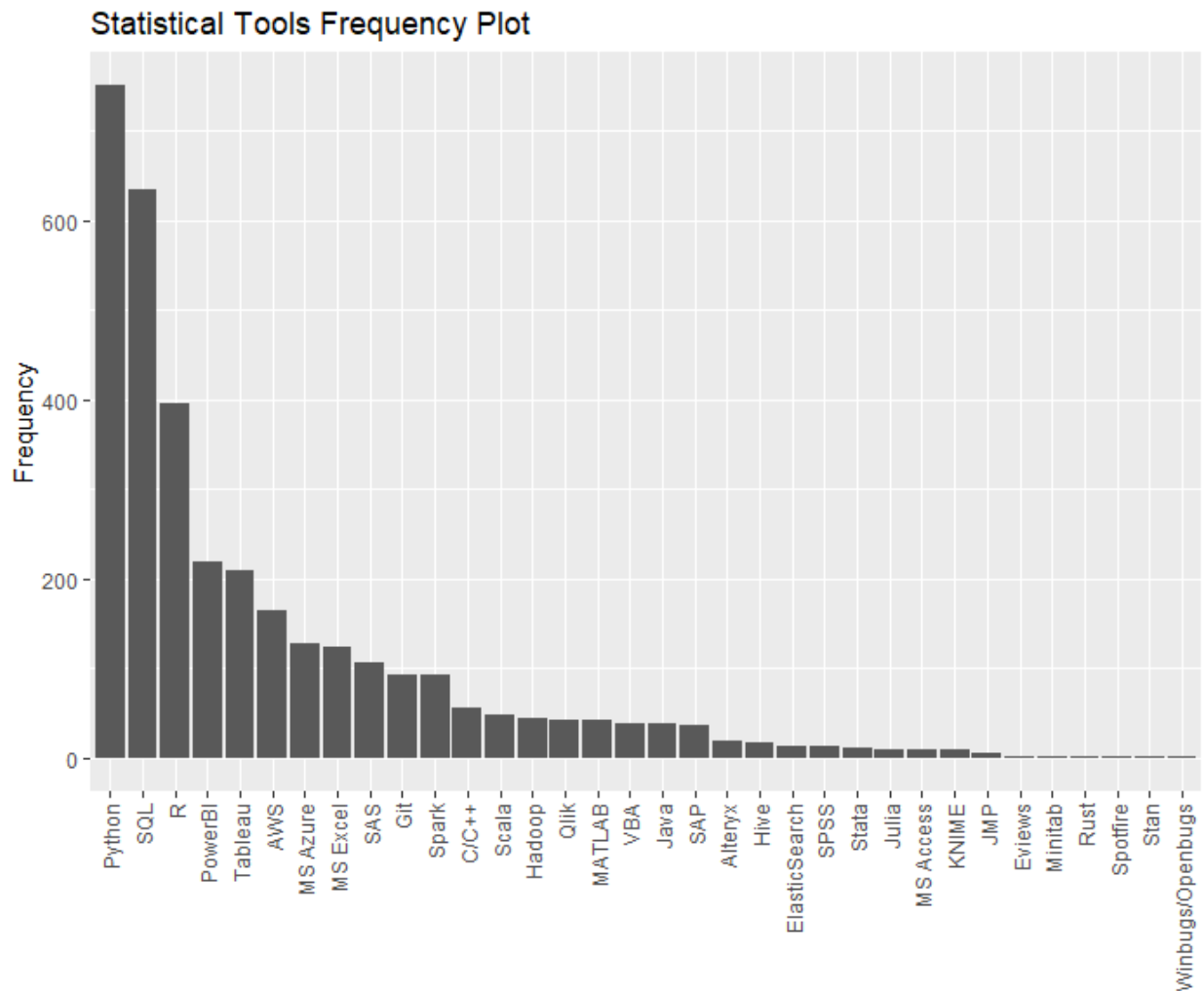
Graph 1: A word cloud of our data. This is an aggregation of the most frequent words used in our dataset. The more times a word appears in our dataset, the bolder and bigger it appears on the *word cloud*.

For our data, as we can gather from the *Graph 1* above, the most popular data science and statistics software was Python, followed closely by SQL and R. It should come as no surprise as in 2022 these are the tools most frequently mentioned both in businesses and in academia due to their simplicity in use, their ability to use powerful packages and versatility in presenting and visualizing the data. Plus, they require no subscription or fee, which is yet another incentive for any skeptical company or school. Tableau and PowerBI come right next to them, and these are proprietary software that were built with a main goal to help workers visualize and summarize the data given in a fast and easy way. These arguably have a more intuitive interface than the previous three mentioned and they do not need any programming knowledge to use them. They both have a free or public version of their respective software, but they are of course more limited to what they can do when compared to the full versions. Perhaps the companies do not endorse them as much as others due to their monthly subscription.

On the other end of the word cloud, we can barely see the Bayesian Analysis packages. Unfortunately, they were not observed practically at all in the job advertisements, but as it was already stated, it appeared as if the companies that were interested in such a skill, did not care which software was used to conduct it.

4.2 Frequencies of the Statistical Tools

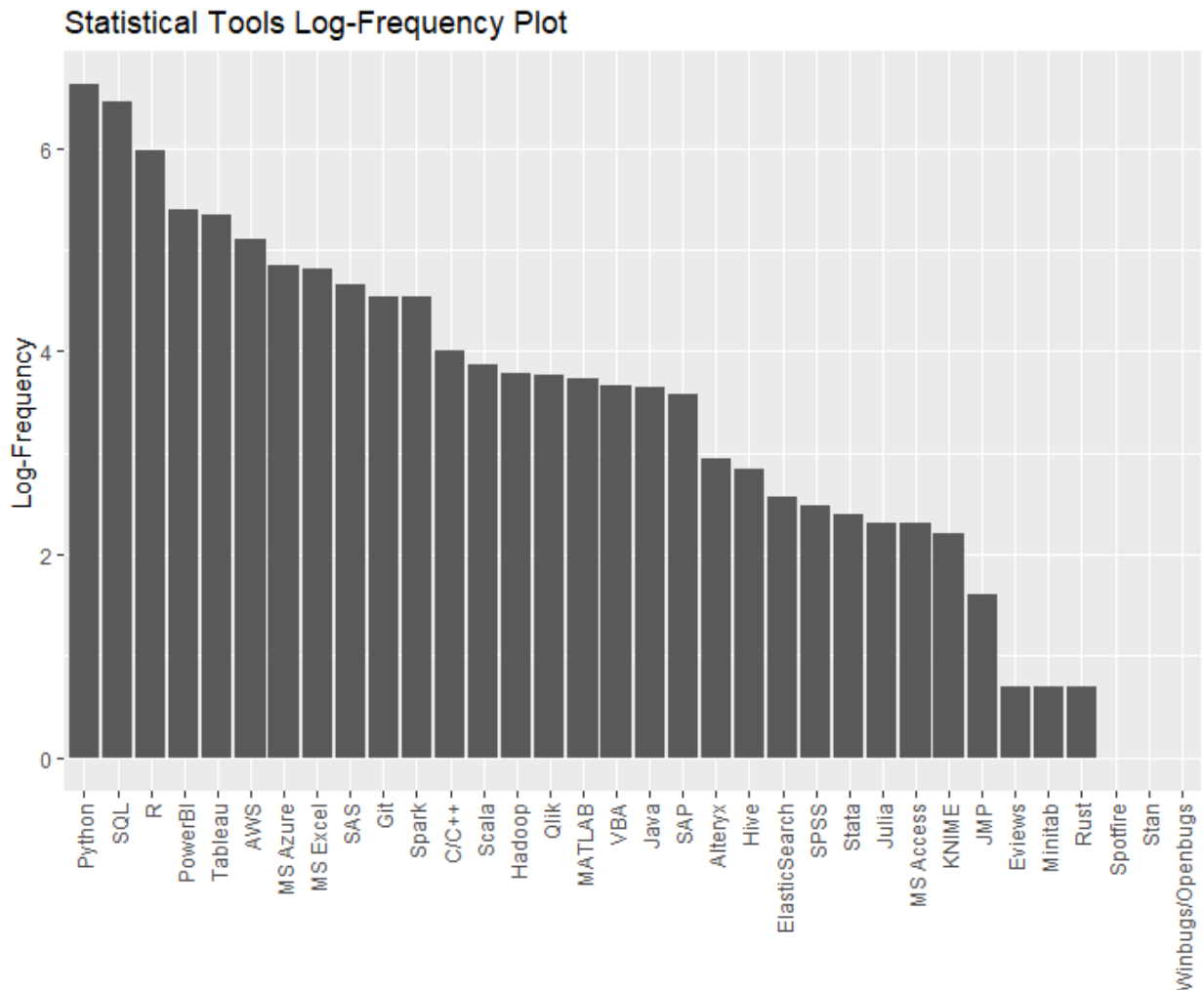
Let us now see that same data in a frequency bar-plot, so as to get a clearer picture about what is really going on:



Graph 2: A plot that shows the frequency for every statistical tool that was monitored.

Here we can distinctly see how these statistical programs, platforms, frameworks and software are ranked in popularity according to the number of times they were observed in our dataset, starting from the greatest and most popular on the left side and ending up with the least famous of them all on the right-hand side. Just as we rightly concluded from the word cloud above, the top three software are indeed Python, SQL and R, but now we see that SQL and Python are much closer to each other than to R, as they were observed more frequently in the dataset. An advantage of this graph is that not only does it show us which software are more popular than others, but that it also depicts the degree, or how far apart they actually are from one another.

The plot above does seem a bit skewed to the right, as the most popular statistical tools had much larger frequencies, when compared to the least popular ones. That way, the last five statistical tools that were being observed seem to have frequency equal to zero, which is definitely not the case. Therefore, in order to make things a bit clearer, here is a graph of the Log-Frequencies:



Graph 3: A plot that shows the log-frequency for every statistical tool that was monitored.

This graph makes it easier to see what the plot above it has already shown. The log-frequencies have added a new level of separation, especially to the programs that were observed less frequently in our data set.

The top three programs appeared a lot of times in our dataset and were rightfully put in the upper level. Here one can also notice a difference that was not apparent on the Statistical Programs Frequency plot. Apparently, Stan, Openbugs/Winbugs and Spotfire were observed only once within our data. Previously it looked as if they were on the same level as other statistical tools such as EViews, Minitab and Rust, which was far from the truth. As it was previously stated, the languages that contributed to Bayesian Analysis were the least popular of the bunch, as the firms

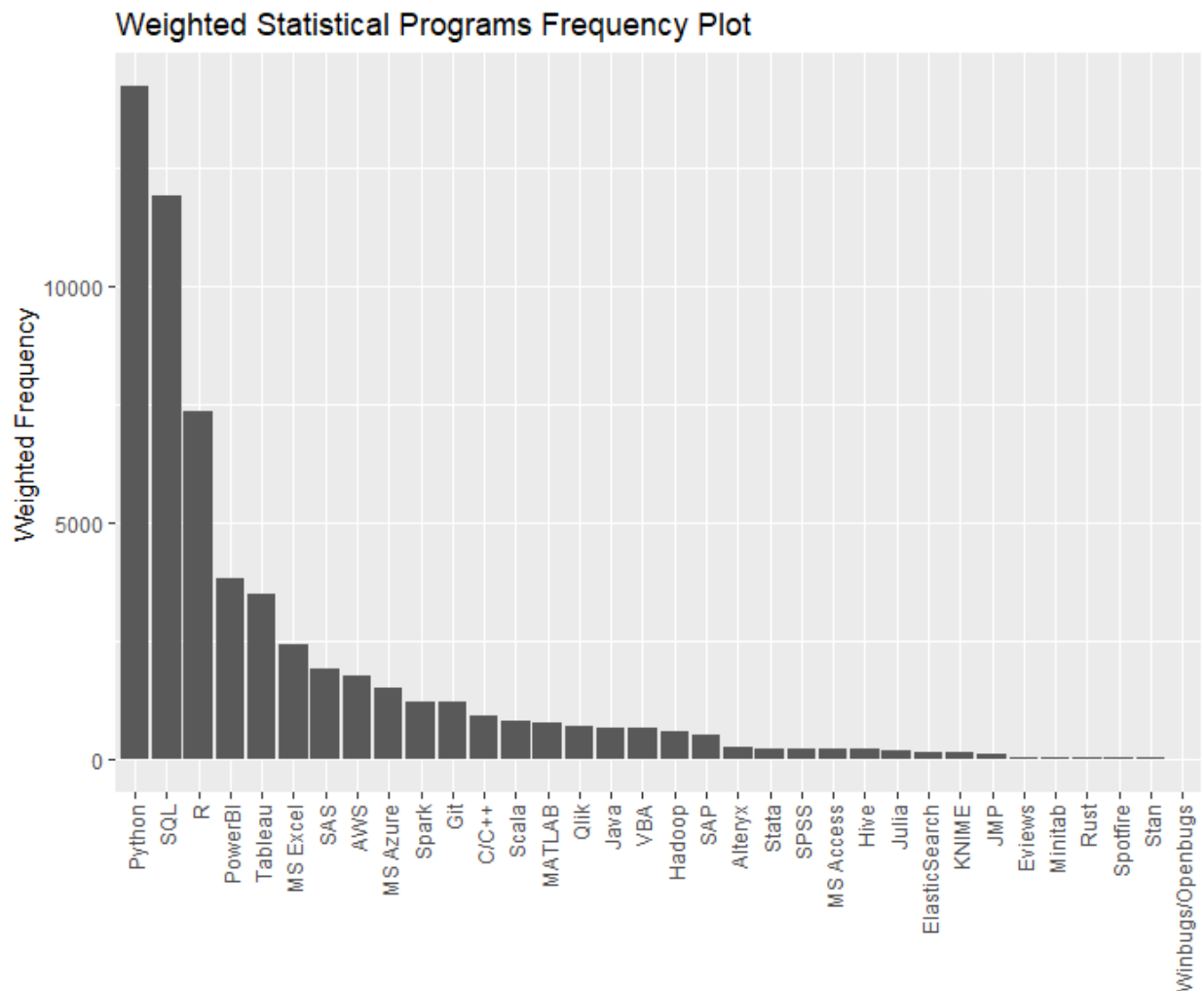
either did not know or did not see it as a major issue with what sort of programs were used to conduct such an analysis on their datasets and hence were very vague when describing their ideal candidate for such a position. In any case, they were not mentioned explicitly in the description and thus received the lowest number of points.

Spotfire was also one of the least frequent statistical tools inside the dataset. Perhaps one reason is its capabilities, or the monthly subscription, or even the fact that it only runs on Windows that made companies try and avoid mentioning it in the required skills of a candidate.



4.3 Weighted Frequencies of the Statistical Tools

While reading the descriptions of job advertisements, there appeared to be an internal categorization between the statistical programs. Some programs seemed more vital to the position than others and it would be interesting to see how they are ranked as far as their usefulness to the company is concerned. To that end, another weighted dataset was created, and this, was the weighted frequency plot that came out of it:



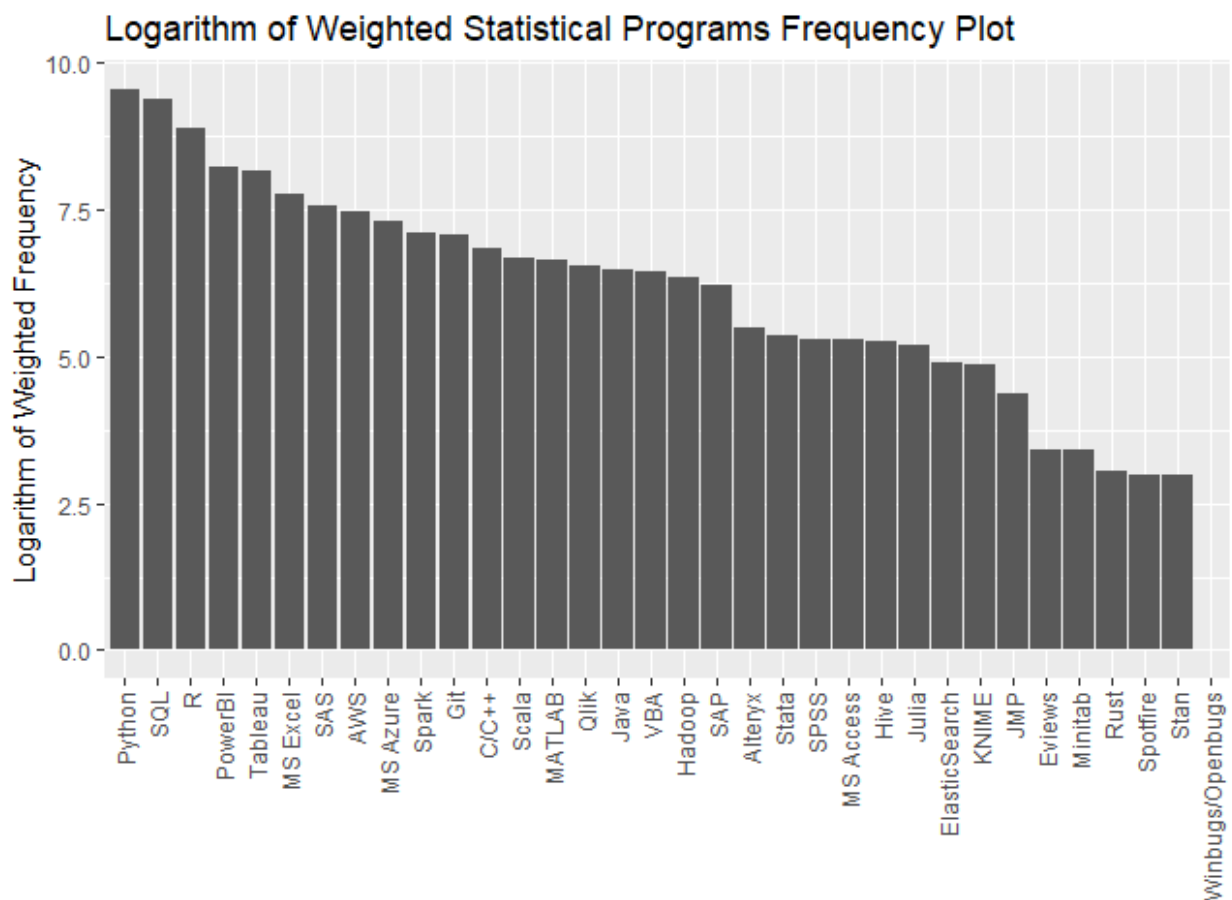
Graph 4: A plot that shows the weighted frequency for every statistical tool that was monitored.

The groups that were used to create *Graph 4* above, were separated in the following manner:

- **Main** category: The programs included in this category are vital to the role of the candidate and he or she must absolutely know them. These are the programs that will primarily be used in the role of the position that is advertised. The weight attributed to this category is 20.

- **Complementary** category: The knowledge of these programs was not as important as in the group right above, however it was still necessary to at least have some experience if possible. The statistical tools in this category play a more supportive role to the other, prime programs used in the category above. The weight given to each piece of software belonging to this group was 10.
- **Extra/Plus** category: As far as these programs are concerned, they were considered an added bonus if the job seeker knew how to use some or all of them, but certainly did not appear to play an enormous role in the selection process. Their weight would thus be that of 1.

Let us also give a logarithm of *Graph 4* above so as to get a better picture:



Graph 5: A plot that shows the logarithm of the weighted frequency for every statistical tool that was monitored.

Therefore, the statistical tools of both *Graph 4* and *Graph 5* depicted above were ranked based on the importance that they have each of the job positions that were observed in our data. Looking at the bar-plot, we can discern some changes when compared to the Statistical Programs Frequency

bar-plot in the previous section (*Graph 2* and *Graph 3*). Here we notice some languages moving upward in necessity for a job position. What does it mean, for instance, that SAS went from position number 9 in *Graph 2* to number 7 in *Graph 4*? It means that despite it not being as popular as other statistical tools, it still is much more useful than it appears. As the data was collected, SAS was the leader in Healthcare and Pharmaceutical sectors (dubbed “*Health*” for the sake of simplicity) and thus it comes as no surprise to see it among the top statistical programs used. Perhaps, if there were more jobs for pure statisticians observed, it could easily be in the top 5.

Inspecting *Graph 4* and *Graph 5*, we can also notice that, for the most part, the least popular statistical tools were also the least necessary to the firms, when they chose the most suitable candidate. That stands to reason, as these programs, frameworks and whatnot are not as widely known as other, better alternatives.

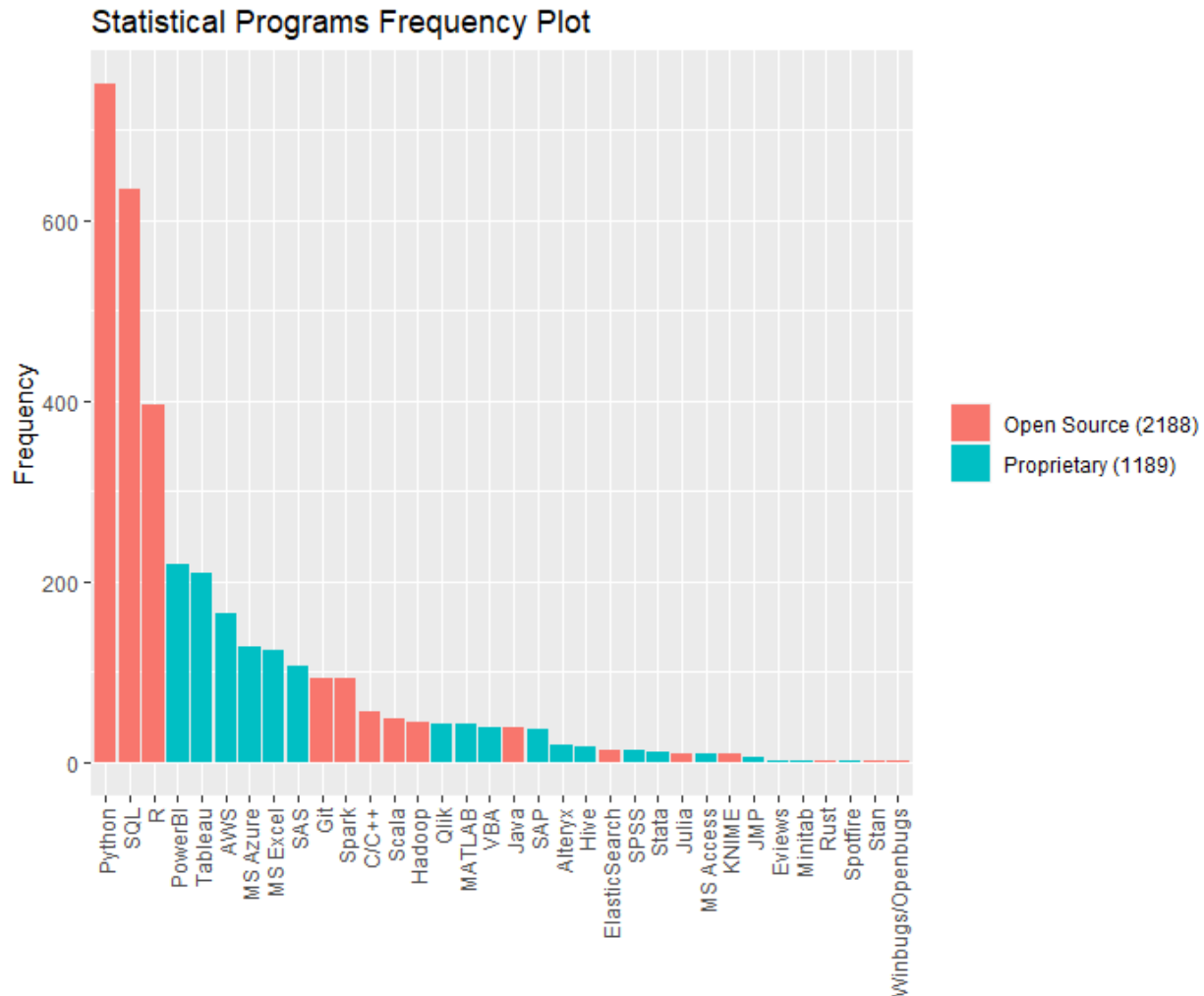
Spotfire and Stan were a bit more useful than Winbugs/Openbugs to firms, something that is especially apparent in *Graph 5*.

There are also statistical tools that are considered up and coming and could in a span of a few years be among the most popular statistical software. Julia is considered to be one of these, however in 2022 it is still quite low on the list. One of the reasons, is that it is mainly used in academic research and is still considered to be a pretty young programming language, when compared to other such tools, as it only made its debut in 2012. Thus, for the moment, this dataset accurately depicts just how famous it is in our era. And this holds for all the other statistical programs that are slowly gaining prominence.



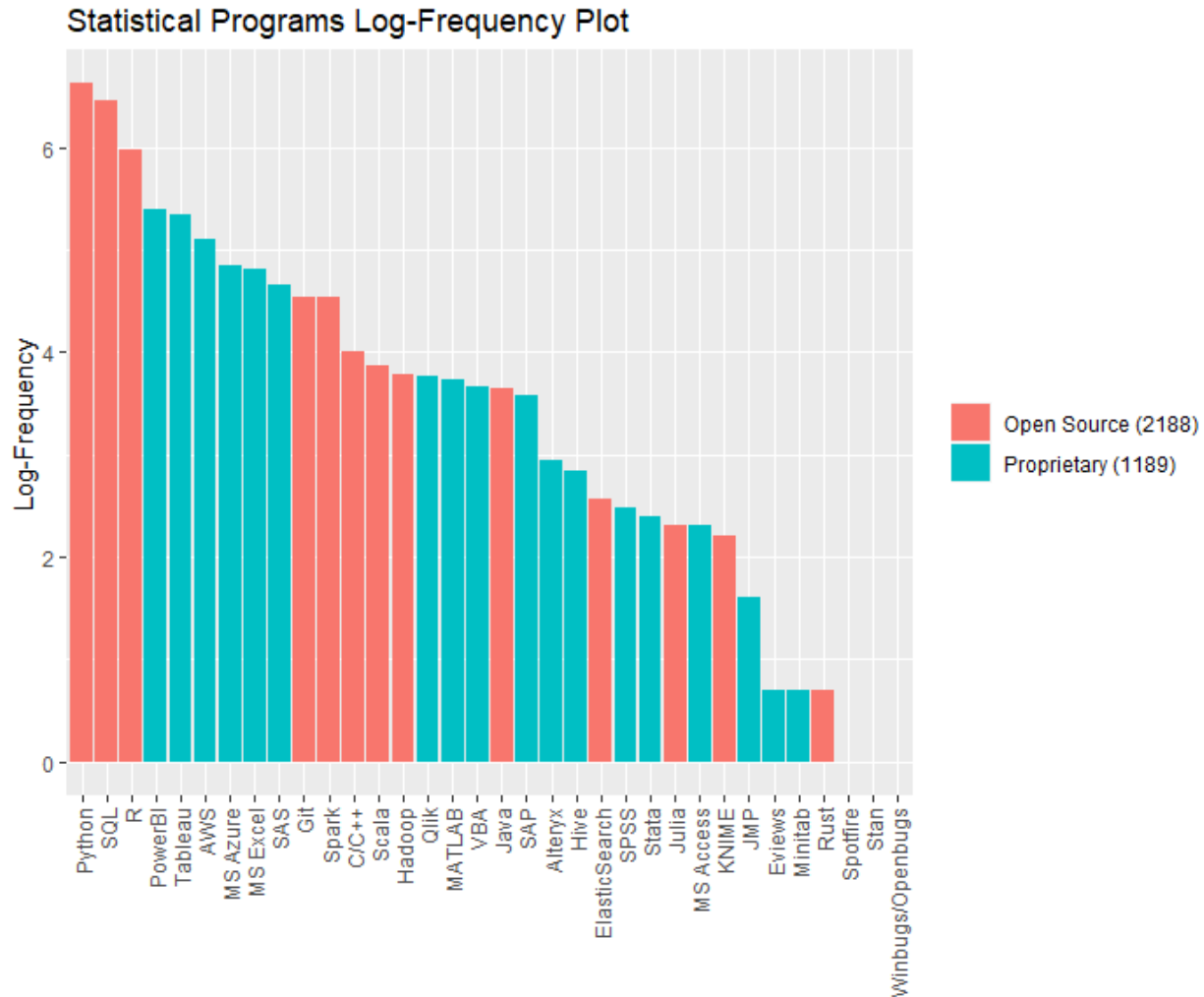
4.4 Proprietary VS Open-Source Statistical Tools

Looking back at chapter 3.1.1 *Statistical Tools*, we can see that some of those programs, frameworks, platforms, etc. were free and open-source to the public, while others were proprietary and users could only access them by buying these tools, or through a subscription plan. Let us now present the frequency plots above, having these two categories in mind:



Graph 6: A plot which shows the frequency for each statistical tool that was monitored, coloured by the category it belongs to (Open-Source, or Proprietary). The number inside the parentheses shows how many times each category appears in our data set, or in other words is the total sum of all frequencies per category.

And of course, for more clarity, here is the same graph as *Graph 6*, with log-frequencies:



Graph 7: A plot which shows the log-frequency for each statistical tool that was monitored, coloured by the category it belongs to (Open-Source, or Proprietary). The number inside the parentheses shows how many times each category appears in our data set, or in other words is the total sum of all frequencies per category.

Looking at the two graphs above (*Graph 6* and *Graph 7*), it can be seen that a total of 15 of these statistical tools were open-source or open for public use, while a total of 19 programs were proprietary.

The top three programs that were in high demand in the job advertisements were free and open-source and they were seen very frequently in our data set, something that especially shows in *Graph 6*.

Of the last three statistical tools, as can faintly be seen in *Graph 6*, Winbugs/Openbugs and Stan were open-source, while Spotfire was proprietary. It has already been mentioned that a possible reason for these Bayesian analysis tools to be in this last place is that they were merely not

mentioned by name, while Spotfire is perhaps there due to the subscription fees and other better alternatives.

In the dataset itself, it would appear that more often than not, the companies would ask for knowledge of free and open-source statistical tools. As it is summarized in the parentheses of the two graphs above (*Graph 6* and *Graph 7*), open-source programs were observed 2188 times, while proprietary were detected 1189 times. Almost two times as much.

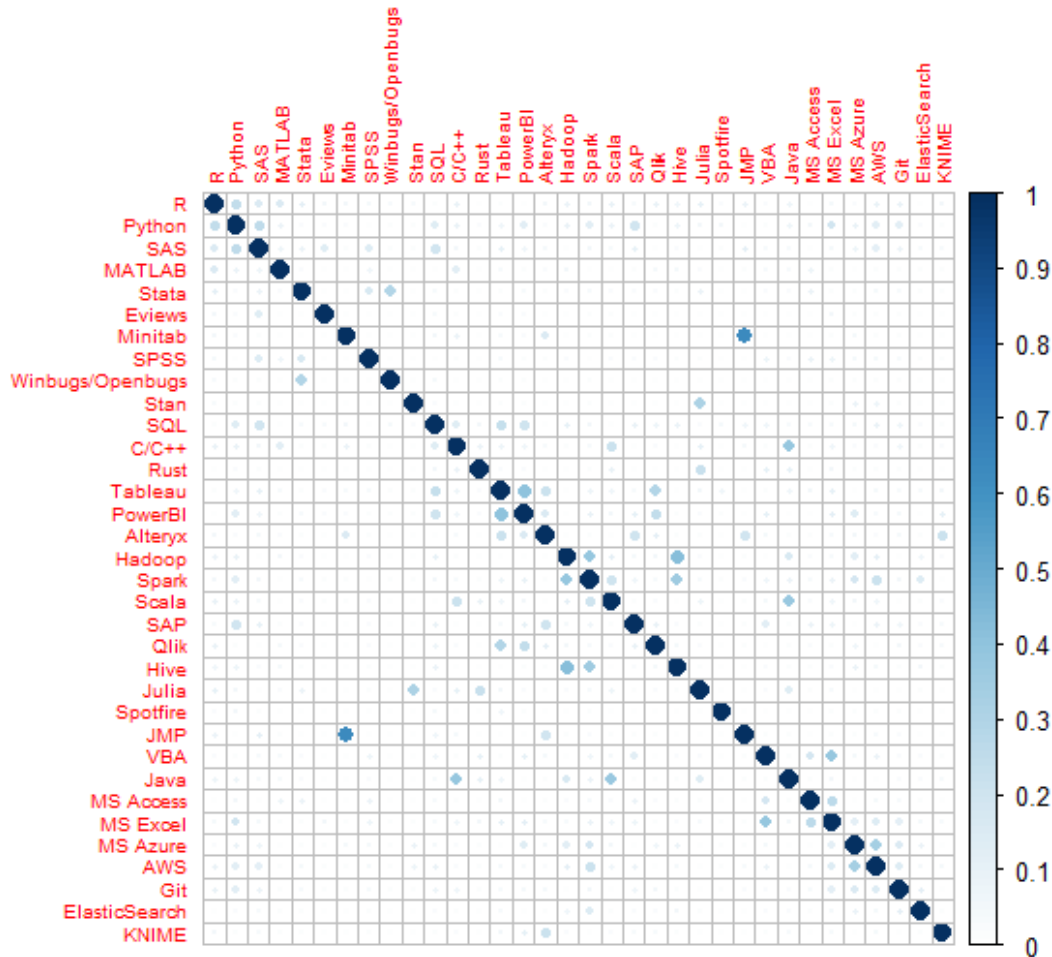
Just because they are free, does not mean that they are not capable of doing great and many things. For instance, R can produce a plethora of complex diagrams, much like Tableau, although admittedly in a much slower pace and plus, the candidate has to know programming in order to use R correctly.

It is astounding to see the contrast between the first three open-source tools and the next six that are proprietary. Perhaps it is due to the subscription fees that many companies choose to avoid them, even though some of these tools make the job a lot easier and are faster in producing their results. Of course not all companies are big companies that can afford these subscriptions. Or maybe it is that these proprietary tools are unnecessary to some firms and their job can be done completely in the other statistical tools that are free and open to the public.



4.5 Correlations

In order to see how strong the correlations are, between the variables present in our set of data, *Cramer's V* shall be applied, due to the fact that these are nominal, categorical variables.



Graph 8: A correlation plot between each variable in the data set, based on *Cramer's V*. It takes two values 0 (the variables are independent) and 1 (the variables are strongly dependent to each other).

So, what can be said about the correlations, when looking at the *Graph 8* above? The first thing to notice is that the majority of statistical tools are either independent, or weakly dependent to each other, something that is apparent from the white, or small, pale, blue dots that appear throughout the correlogram. Perhaps the only two statistical tools which are strongly correlated to each other are JMP and Minitab. They seem to have a correlation coefficient of approximately 60%, which means that in the majority of cases observed, whenever jobs needed the candidate to have either one of these two tools, it was highly probable that the other would also be a prerequisite. R and Python share a weak correlation between them; hence it appears that these two pieces of software were seen together in some, but definitely not the majority of all cases. And the same thing can be said about Tableau and PowerBI, for instance, as these two programs were often mentioned together.

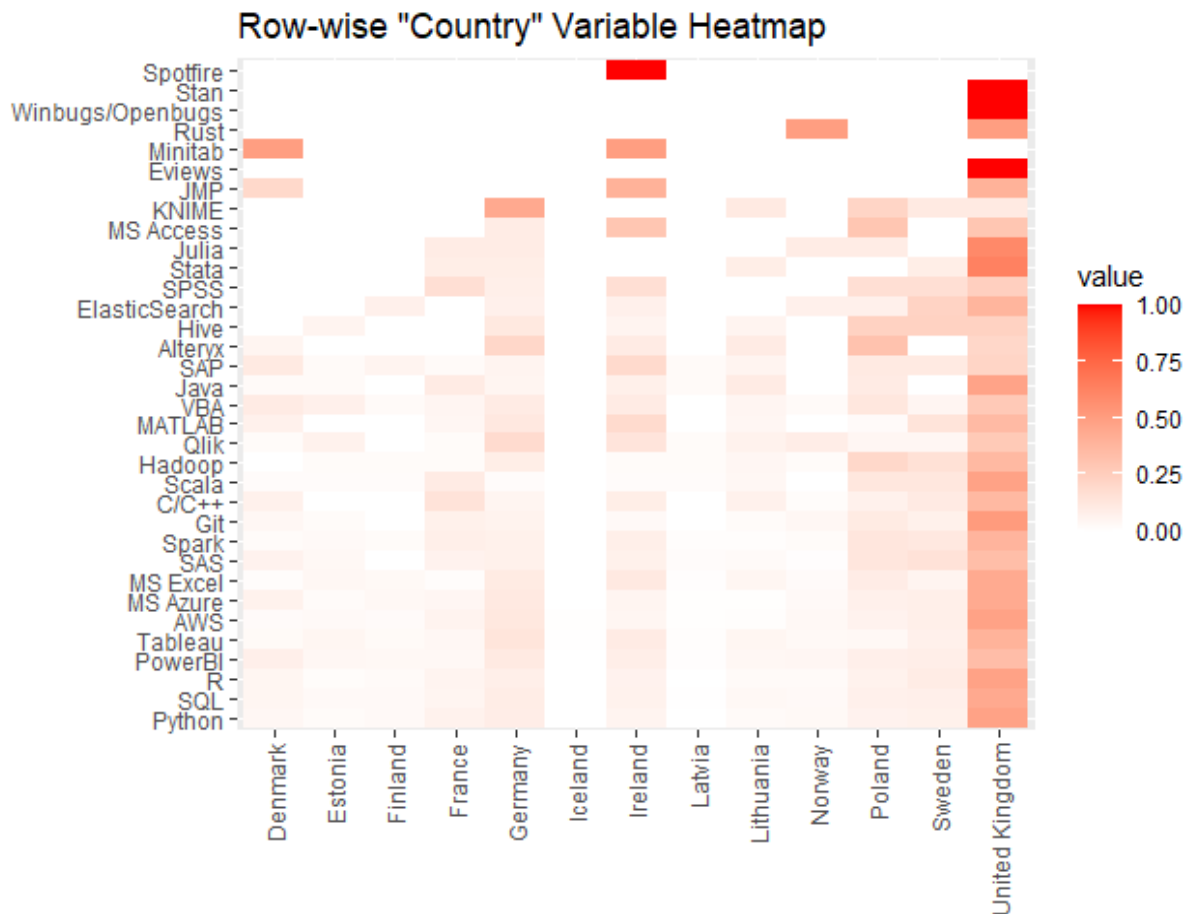
ther. Plus there are some weak correlations among such programs as Hadoop, Spark and Hive, which were expected to be correlated in some sense since they are part of the same suite (Apache). And the same thing can be said about MS Excel, MS Access and VBA, that are a part of the Microsoft Office Suite.



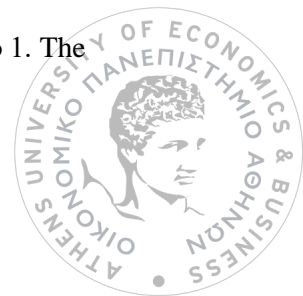
4.6 The “Country” Variable

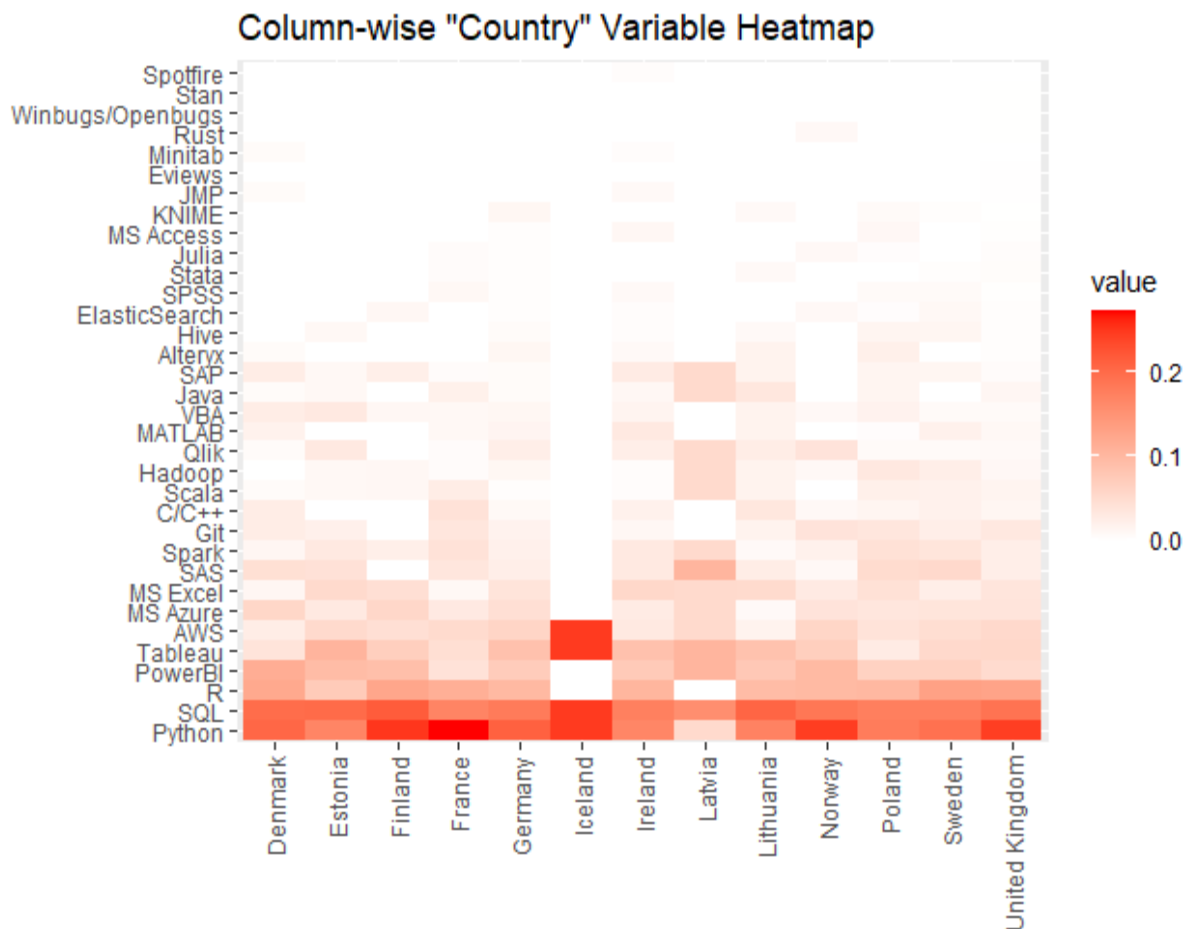
The “Country” variable contains all the countries that were used during the collection of our dataset. In total we collected data from 13 northern European countries. Now that we have this variable, it is not a farfetched question to ask, whether we can make groups of these countries based on the variables that contain the presence or absence of statistical software. In other words, can we really say that one country differs from another as far as job descriptions are concerned and hence, the set of skills that they require? It would stand to reason that they should not. An average Data Analyst in Sweden should be using the same technologies as the one working in say, France. The statistical tools are famous because they are easy to use in any part of the world. The only thing that could be said is perhaps that for some reason (perhaps education) the people in France tend to use R more often than people in Denmark. But apart from that, and since a lot of the firms that are hiring are international companies, meaning that they likely use the same technologies across all of their branches, it is expected that there will not be any perceivable difference in groups and thus, this shall be an unsuitable variable to group our data.

Perhaps it would be better to show how well these groups are formed in the following two heatmaps that concentrate on row-wise and column-wise proportions:



Graph 9: Is a plot of row-wise proportions for the variable “Country”. Each row above adds to 1. The statistical tools (on the y-axis) were ordered by frequency.





Graph 10: Is a plot of column-wise proportions for the variable “Country”. Each column above adds to 1. The statistical tools (on the y-axis) were ordered by frequency.

It is apparent that *Graph 9* compares how much more a statistical tool appears in job advertisements between countries. For example, it could answer a question of the following form: “Out of the 1000 observations that were collected, what percentage of R appeared in Latvia, France, and the other countries that were observed?”. It would appear that most of the observations were in the United Kingdom, which rightly depicts the job advertisements that were found per each country, as that country had the most jobs for statisticians, data analysts and data scientists, followed (but not too closely) by Germany and Ireland.

On the other hand, *Graph 10* shows which statistical software are more commonly asked in the job descriptions within each of the 13 countries that were monitored, in other words it concentrates on each country individually. For instance, it could answer questions of the following form: “What was the proportion of R observed in one of the 13 countries aforementioned, when compared to the other 33 statistical tools?”. Here we can also confirm that the most used software are R, Python and SQL almost in every country, something that was clear in the frequency plots before.

As suspected, these separations are very vague. Should a machine learning algorithm that aims to separate these countries be based only on the statistical tools provided in this dataset, then the results will most likely be disappointing. It is curious to see that Latvia did not have one single

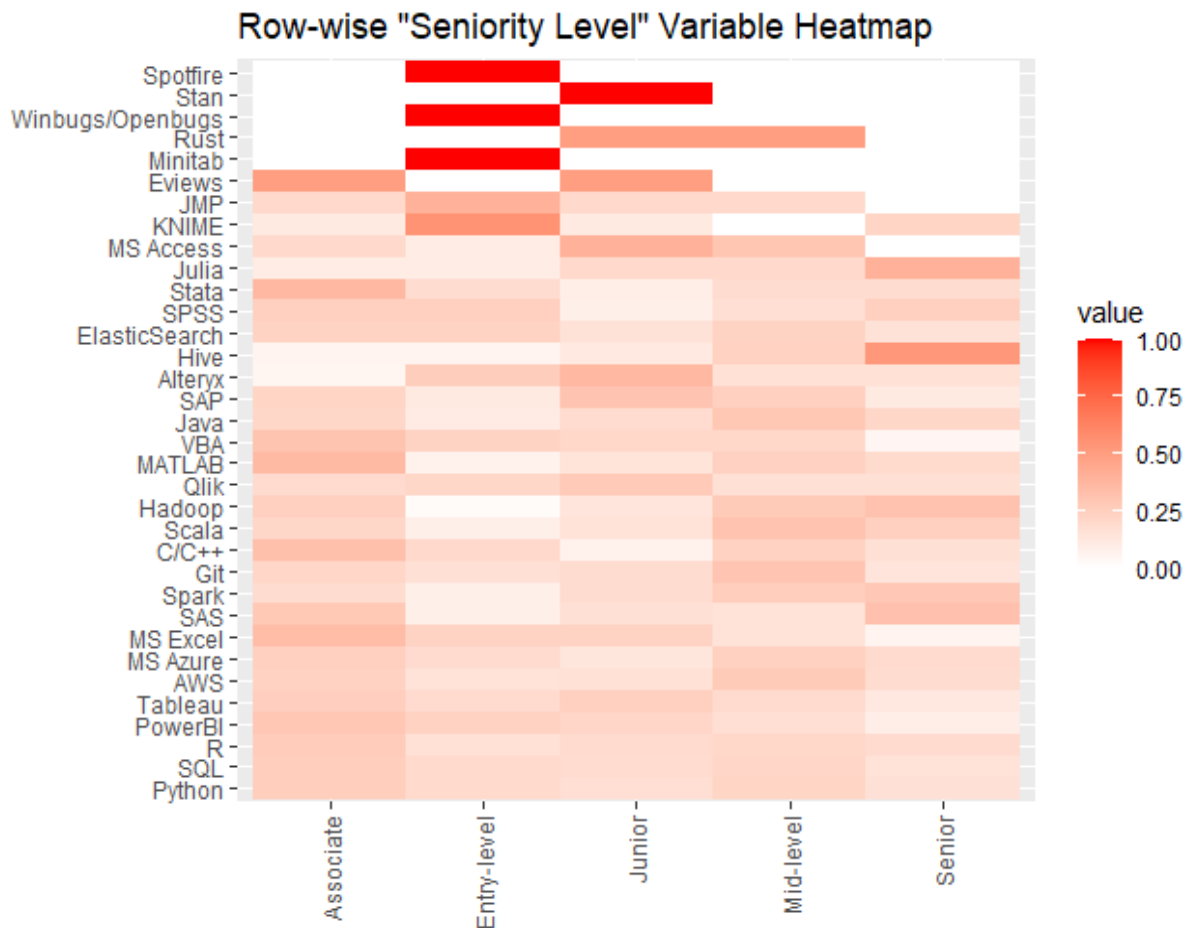
company that needed the knowledge of R from its candidates, but other than that when looking at the two graphs, the separations are almost imperceptible. Comparing, for example, Denmark and Estonia, we see on the column-wise graph that they both have similar demands as far as statistical tools are concerned, with the exception that Denmark also uses MATLAB, Mini-tab and C/C++, while Estonia does not. Looking at the row-wise graph however, it can be seen that the observations that contain these programs are perhaps too few to make a clear-cut separation between these countries and this holds for most of the other countries in our data, hence making this variable potentially inappropriate for grouping the data.



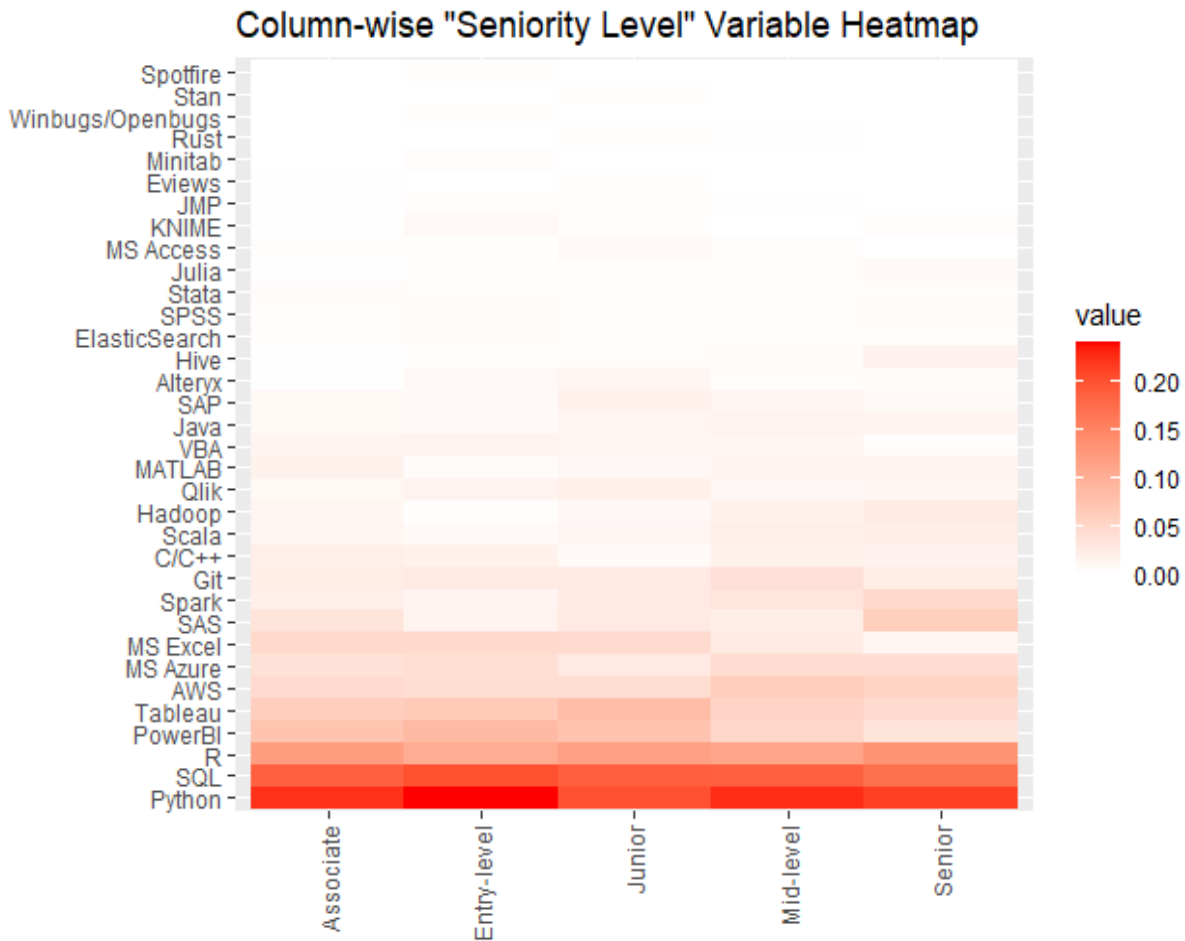
4.7 The “Seniority Level” Variable

This variable was another candidate for the grouping of our data based on the statistical tools provided. It was decided that this variable would have 5 levels in total (which are shown in *Table 8*), based upon the experience needed to successfully apply for the job. One would expect it to answer questions like: “Are Seniors expected to know different sets of statistical tools when compared to Junior level staff?”

Perhaps yes, or maybe no, in order to find an answer, let us plot the heatmaps much like it was done for the “Country” variable, in order to visualize any groupings that there might be.



Graph 11: Is a plot of row-wise proportions for the variable “Seniority Level”. Each row of the graph above adds to 1. The statistical tools (on the y-axis) were ordered by frequency.



Graph 12: Is a plot of column-wise proportions for the variable “Seniority Level”. Each column of the graph above adds to 1. The statistical tools (on the y-axis) were ordered by frequency.

Looking at *Graph 11* and *Graph 12* above, one can see that, although vague, some groups do seem to form and there appears to be a slight separation between them. Perhaps it is not as bad as it seemed, however, let us not forget that most of these job positions used either R, Python or SQL and therein lies a lot of uncertainty. Column-wise, these positions are almost identical, while row-wise, those programs that are coloured in crimson red, were only a few observations in total (perhaps 1 or 3 in total out of a 1000) and therefore would be reliable, but for these few select cases. Thus, groups could probably be formed, but there is a lot of mixing and confusion to be expected. After all, the colouring of these groups seems to be almost identical when we look at each row individually and therefore maybe this is not such a good grouping variable after all.

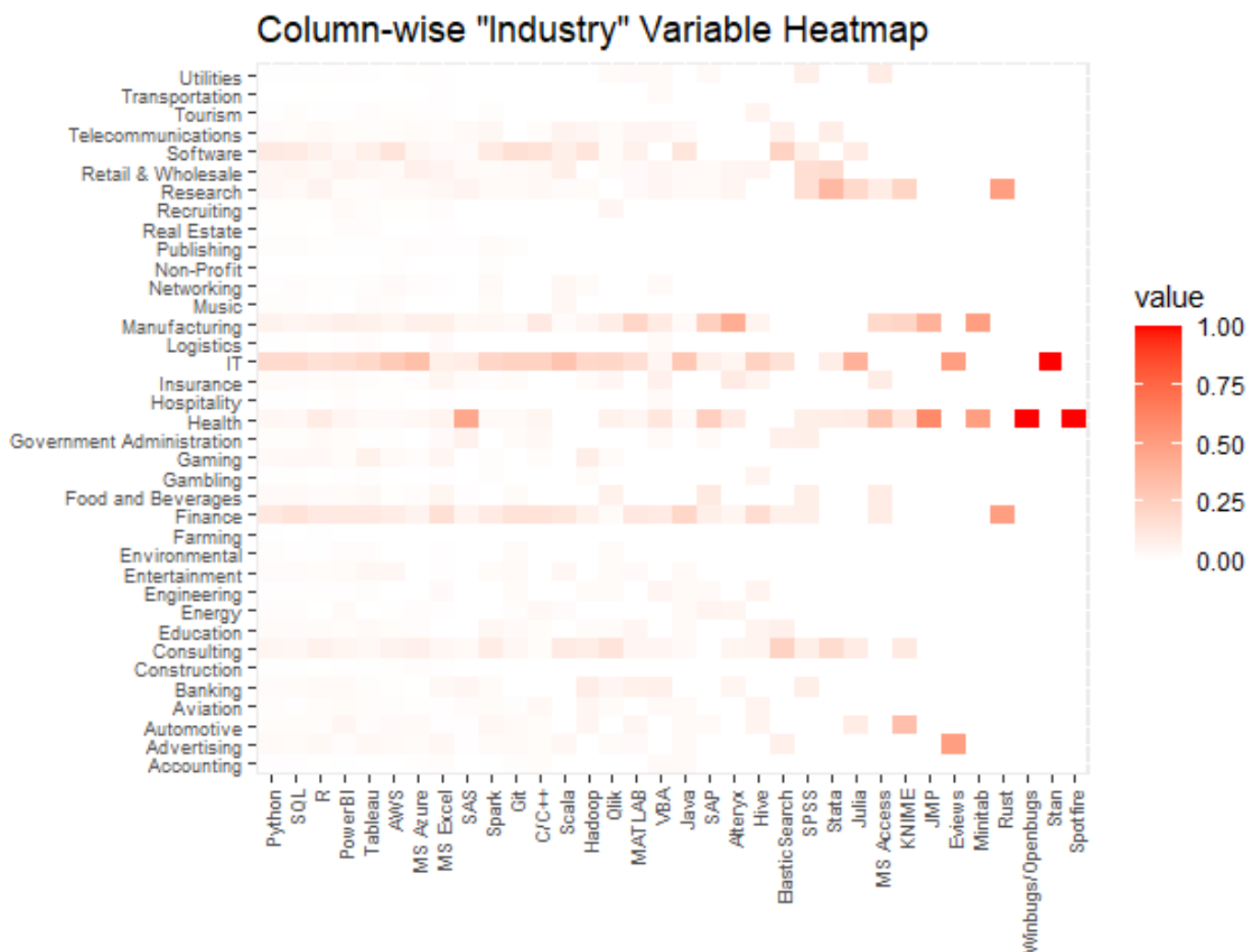
There does not appear to be any surprises in general and the skills seem to be equally distributed among the levels of the variable.

Therefore, in general, it is expected that all the people that work within a firm, shall know more or less the same statistical tools.

4.8 The “Industry” Variable

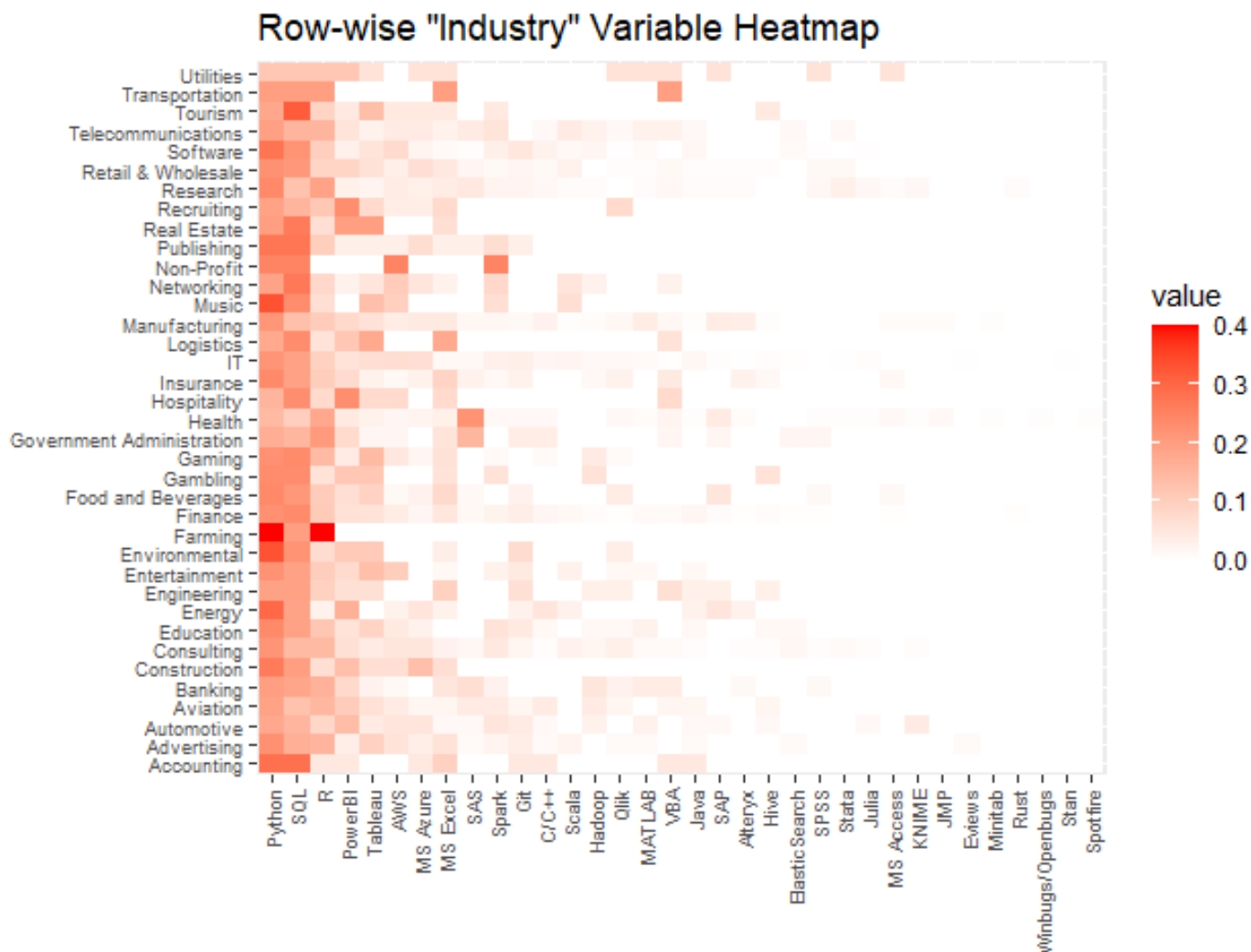
Another variable which was observed in the process of collection, was the “Industry” variable. It consisted of 37 levels, which are summarized in *Table 6* that captured the broader purpose of each company.

Much like before, the following heatmaps can provide some insight as to what is going on with it and the statistical tools that are used:



Graph 13: Is a plot of column-wise proportions for the variable “Industry”. Each column of the graph above adds to 1. The statistical tools (on the x-axis) were ordered by frequency.





Graph 14: Is a plot of row-wise proportions for the variable “Industry”. Each row of the graph above adds to 1. The statistical tools (on the x-axis) were ordered by frequency.

Looking at *Graph 13* and *Graph 14*, one can see that in general, industries tended to prefer the most frequent and popular software. In *Graph 13*, it is noticeable that the majority of the most popular software tended toward the IT, Software or Finance industry, while the Health industry used SAS the most of all other industries. In *Graph 14*, right below, it is clear that SAS was also used in the Government Administration industry, but certainly to a lesser degree than in the Health industry. Or it would also appear that Transportation industry seemed to avoid proprietary programs, as much as it could and that the Farming industry used only the three most frequent statistical tools.

Unfortunately, this does not look good at all. Should we want to separate these combinations of software based on the “Industry” variable, then theoretically we would have a lot of trouble, as many of these groups would merge into one. Looking at the column-wise graph and due to the fact that this factor has too many levels, categories such as Farming, Real Estate and Publishing, for

example, have no perceptible differences and albeit being very different in their function, they appear inseparable in this set of data. At times it would appear that firms used one type of software consistently, such as SAS, which was mainly used by the Healthcare and Pharmaceutical sectors. And of course, there were other times, when it appeared that companies from entirely different industries would use the same combination of statistical tools.

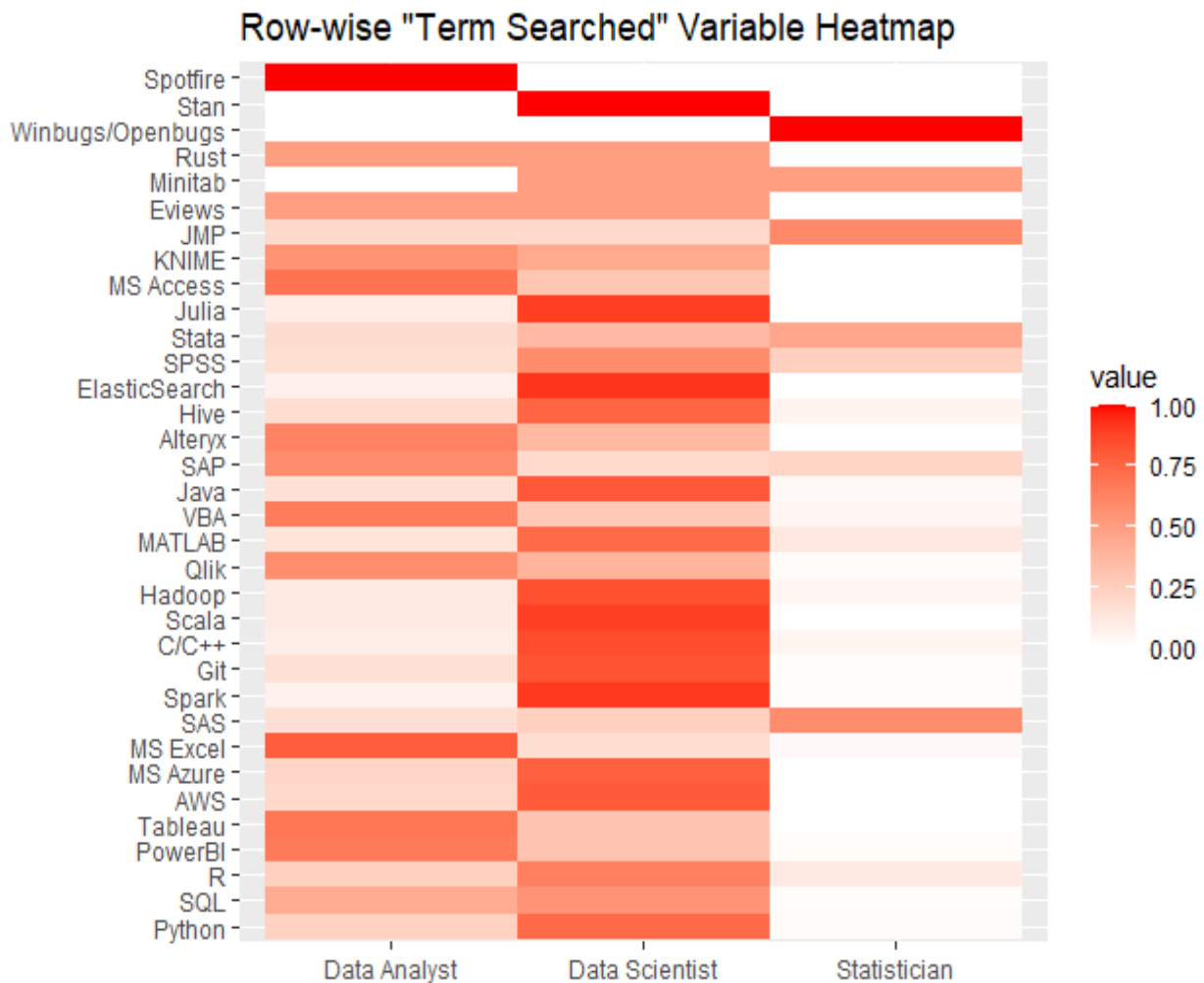
This could signify that, although some industries do have statistical software that they use more than others, the majority of these sectors use the same tools. Thus, seemingly, it matters very little in which industry you work, as it is possible that you will be using the same statistical software as the other industries. And therefore, a Statistician who works in the Construction industry could be using the same software as a Statistician who works for the IT industry, which stands to reason.



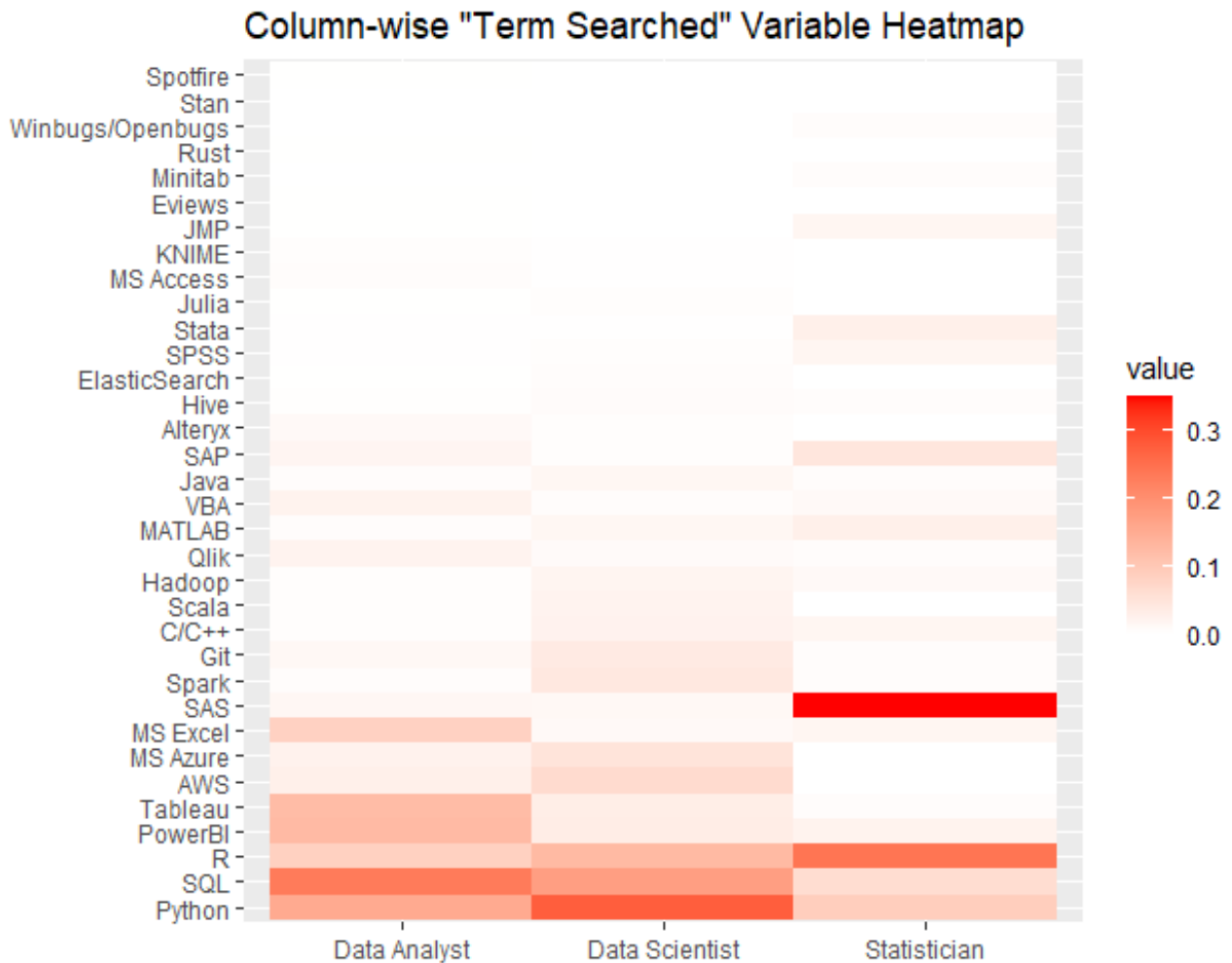
4.9 The “Term Searched” Variable

This variable took note of the term that was used to search for various, statistically related job positions. Three terms in total were deemed proper to use for such queries. While the data was being collected it was apparent that some positions used similar arrangements of statistical tools. It could be that the job of a Statistician in one company used the same software as the job that was looking for a Data Scientist in another. Clearly that could potentially pose a problem, should we want to divide that said software based upon these three categories.

Thus, let us plot heatmaps in order to make clear whether we can perceive with the naked eye if there are groups that can be formed based on the levels of this variable.



Graph 15: Is a plot of row-wise proportions for the variable “Term Searched”. Each row of the graph above adds to 1. The statistical tools (on the y-axis) were ordered by frequency.



Graph 16: Is a plot of column-wise proportions for the variable “Term Searched”. Each column of the graph above adds to 1. The statistical tools (on the y-axis) were ordered by frequency.

It appears that the three categories in *Graph 15* and *Graph 16* are separated, even though not perfectly. As it can be seen especially in *Graph 16*, all three groups use Python for instance, although Data Scientists seem to use it more frequently. Data Analysts seem to use software that is aimed mostly at data analysis and data management, like Tableau, PowerBI, Alteryx, MS Access, etc. Data Scientists do much more programming, as it appears that they are first in Python, R, Julia, etc. Finally, Statisticians have a clear preference for SAS and to a much lesser degree, Stata and JMP, while Minitab seems to be equally separated among them and the Data Scientists.

Here the groups, while not always perfect are better separated from each other, than in the previous three cases of grouping variables. Once again, the crimson red variables in the row-wise heatmap are the rare cases that appear to provide adequate separation, for those few cases where they were observed.

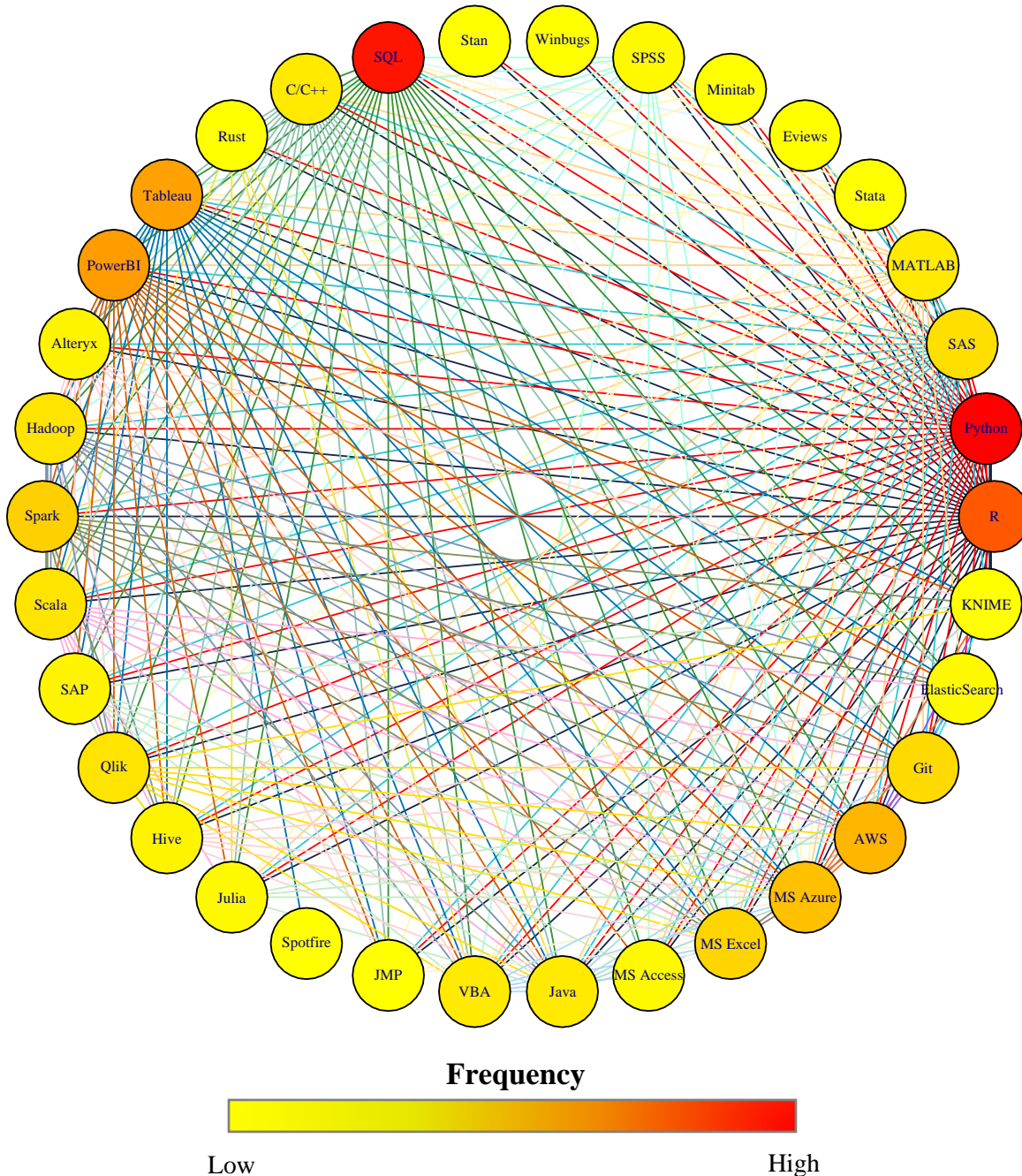
Thus, out of all the previous variables, this one gives us some indication that groups might be formed should these 34 statistical tools be clustered together having this variable as a separator,

although the separation itself might not be perfect, as in some cases, the job positions demanded some extra knowledge of other statistical tools that were observed in the other two terms that we re searched.

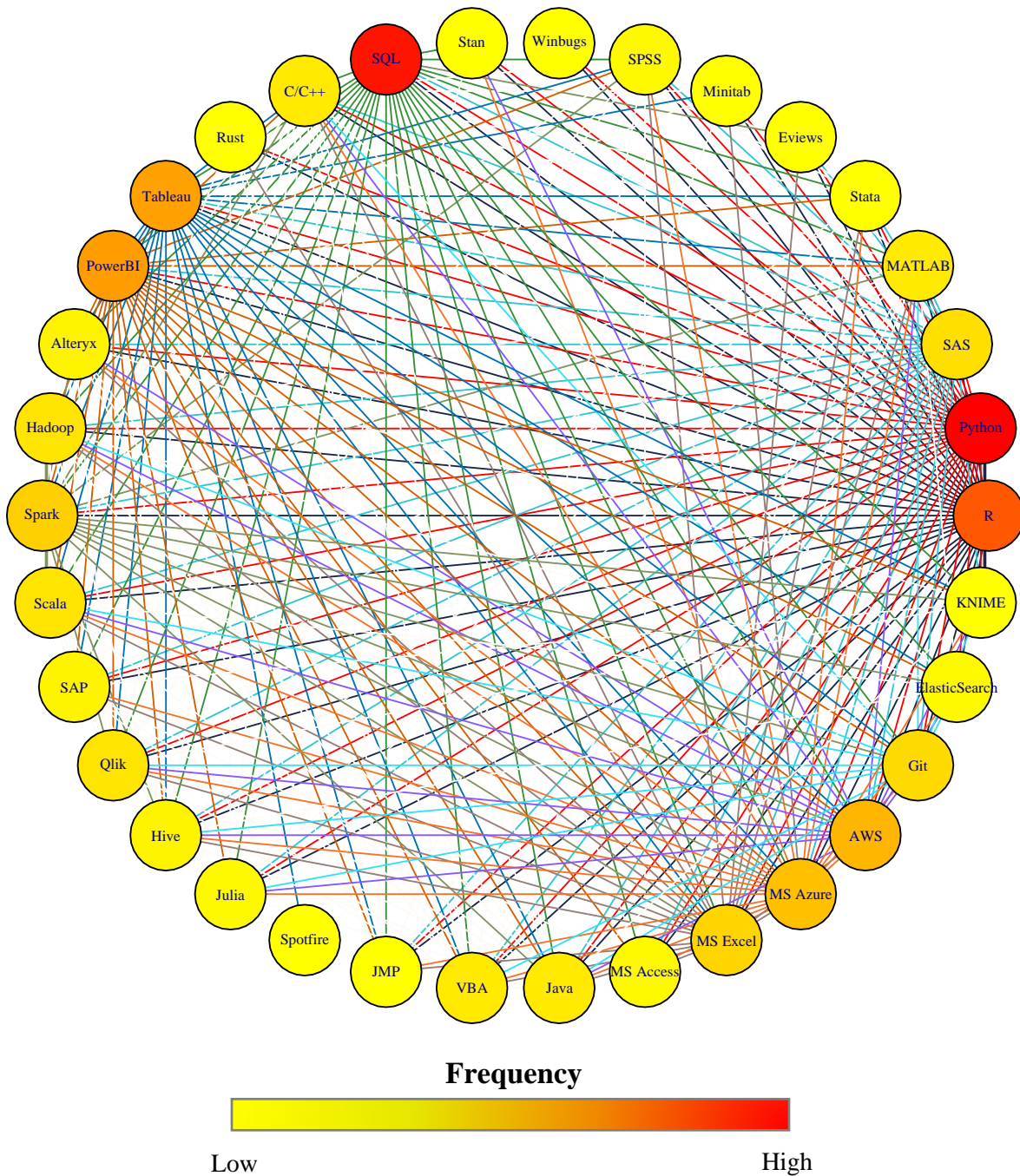


4.10 Network Analysis

It would also be interesting to visualize the way in which the statistical tools are connected to each other. This can be done by plotting a network of the observed data.



Graph 17: Is a plot of the network formed between the statistical tools. The nodes are coloured by frequency and so are the edges. Thus, more frequent tools have red nodes and bolder-coloured edges, while the less frequent they are, the yellow are the nodes and the less visible are the edges.



Graph 18: Is a plot of the network formed between the 11 most frequent statistical tools. The nodes are coloured by frequency and so are the edges. Thus, more frequent tools have red nodes and bolder-coloured edges, while the less frequent they are, the yellower are the nodes and the less visible are the edges.

Two graphs were presented above *Graph 17* and *Graph 18*, that depict the connections of statistical tools as they were found in the job descriptions. For instance, Winbugs/Openbugs was observed once, and that job description also indicated an interest in R, Python and Stata and that is why these four are connected by lines.

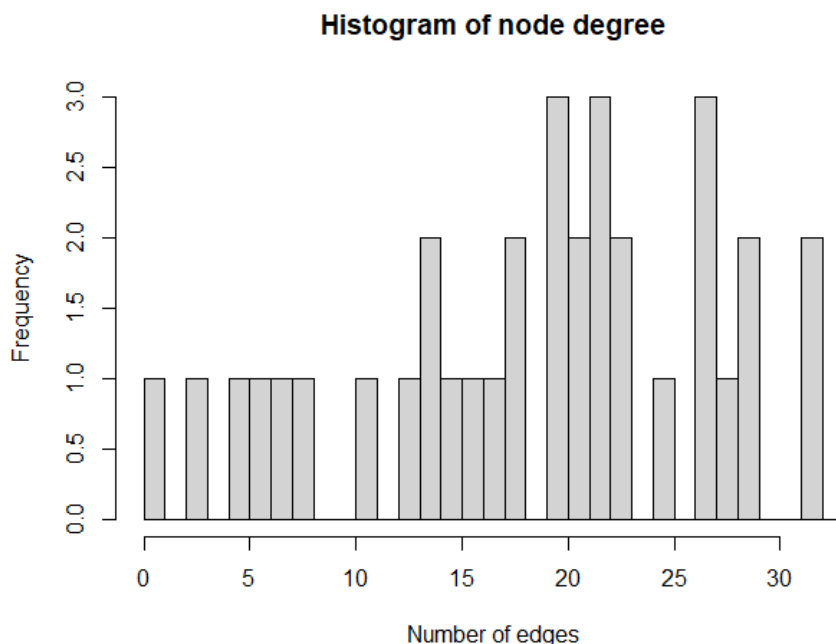
Admittedly these are very dense graphs that can be difficult to read. *Graph 18* was made especially to make things clearer, but as it stood to reason, the most frequent statistical tools were most likely to be seen with almost all the other ones, as they appeared in more and more job advertisements.

Looking at the network graphs above (*Graph 17* and *Graph 18*), we can see which statistical tools were mentioned together throughout the data that were collected. For instance, out of the 1000 observations in total, Spotfire was mentioned once along with Tableau, which can be seen because as there is one line connecting those two. Stan on the other hand was noticed to go along with SQL, Python, Julia, AWS and MS Azure. Python and R were seen with every single program, except for Spotfire. And similar observations can be drawn from the graph above for any of the other vertices.

Since the interest of this network is to see how the statistical tools were paired, this is an undirected network, meaning that these edges are simple lines and not arrows.

The diagonal of the adjacency matrix used was zero, which means that there were no self-edges. In other words, one would not see Python have a line that loops back into Python, but only lines that connect Python to other statistical tools.

Let us now present some measurements of connectivity:



Measurements of Connectivity

<i>Number of Vertices</i>	34
<i>Number of Edges</i>	318
<i>Density</i>	0.5668449
<i>Reciprocity</i>	1
<i>Transitivity</i>	0.7598408

Graph 19: This is a histogram of node degree for our network and right next to it, are the measurements of connectivity.



In total the network had 34 points that formed 318 connections between them. If this network was complete, then 561 connections would have to be formed, but here, only approximately 57% of connections were made. This network is a directionless network, it was designed that way, but it also can be confirmed from the reciprocity, which is 100%. Lastly, around 76% of all the nodes had transitive connections between them. For it to have been 100%, all the statistical tools would have had to be mentioned together at least once for each node in the network.

Looking at the histogram of *Graph 19*, it can be seen that most nodes had somewhere between 15 to 30 connections between themselves. Two of them (Python and R) formed a staggering 32 connections with the rest, while on the other end, Spotfire had only one connection.

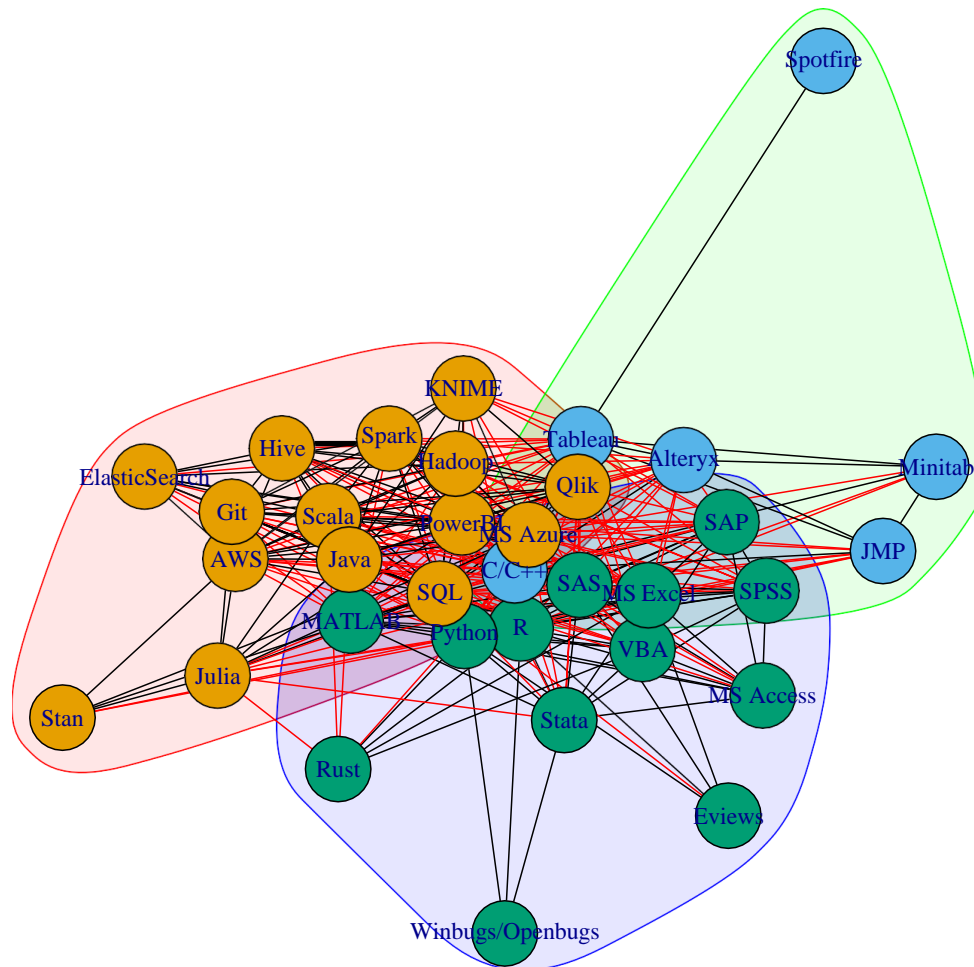


4.11 Network Clustering for Statistical Tools

These tools can potentially be grouped into clusters through Network Analysis. For this part of the thesis, two such algorithms shall attempt to do so: One that seeks structure in communities formed through a greedy optimization of modularity (*cluster_fast_greedy*); and another that tries to find latent positions of objects within a Euclidean space (*latentnet*).

4.11.1 Clustering through Greedy Optimization of Modularity

Should clustering in such a method be performed, then the graphical result shall look somewhat like this:



Graph 20: A plot of our network of statistical tools that was split into three clusters, through the method of greedy optimization of modularity.

As we can see from *Graph 20*, the clustering procedure seems to be a tough one, as these statistical tools get mingled together instead of forming their own distinct groups. Frankly, this is a logical result if one considers the nature of the data, as many job advertisements suggested that the ideal candidate should know a plethora of programs and software and thus, these formed the many connections that are seen right above.

It would appear that the algorithm has detected three groups in total, with Spotfire floating further away, as it has only one connection. The clustering itself is a bit unexpected, as for instance, it has put Tableau in the blue group, when it was mentioned many times along with SQL and PowerBI and it should have at least been considered to be a part of that group. Perhaps it was a fault of the hierarchical structure hypothesis, or maybe it was pulled away from the other cluster by rare connections such as Spotfire, or Minitab.

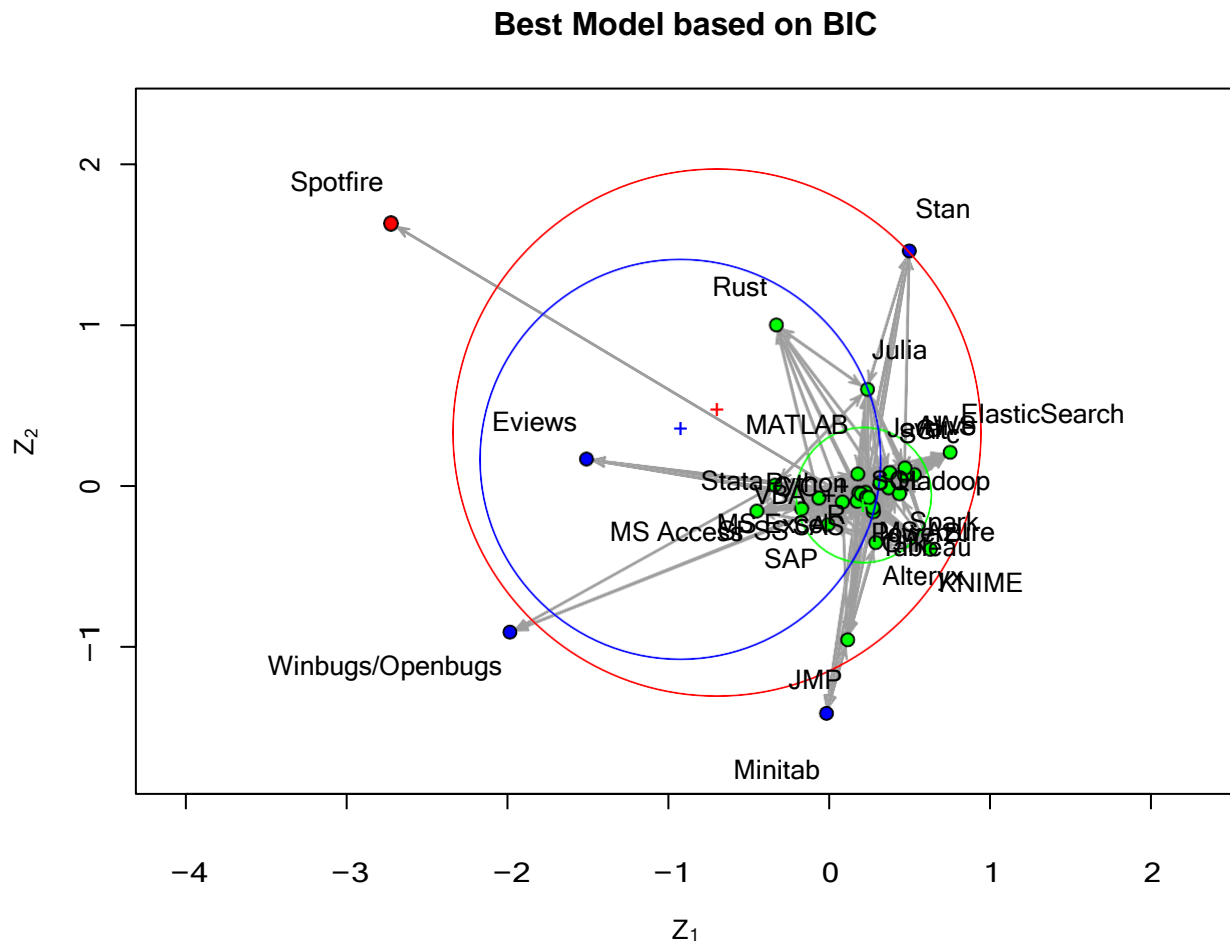
The green and orange groups seem to be alright. It is a bit odd to see SAS in the green group, as it would have been thought to be in a different category, but it was mentioned with R and Python quite a bit and thus could be considered a part of those two.

Other than that, the nodes in these categories do not perform the same functions (such as data management, or data analysis), nor are they a software vs frameworks vs programs type of separation and neither are these groups separated by their frequency (higher to lower for example), therefore, this clustering seems to be a bit faulty in that manner.



4.11.2 Clustering through Latent Positioning

Since our matrix contains frequencies and therefore counts for the weights of the edges, it would be prudent to take it into account. Furthermore, a model selection was attempted based on the Bayes Information Criterion (*BIC*), which admittedly is not as robust as the Integrated Completed Likelihood (*ICL*) criterion (P. Papastamoulis, M.-L. Martin-Magniette, and C. Maugis-Rabusseau, 2016; C. Biernacki, G. Celeux, and G. Govaert, 2000), however that was all that the *latentnet* package could provide when choosing a suitable model. Therefore, let us now apply it to our data and see what the results are:



Graph 21: A plot of our network of statistical tools that was split into three clusters, through the method of latent positioning.

As it can be seen from *Graph 21* above, this algorithm has split the dataset into three clusters. From the outside in, we can see that there is one red point of a cluster, which is the one belonging to Spotfire, because it had only one edge, then four points that were aggregated together into the blue cluster and finally the rest were all put into the green one.

This separation is a logical one and appears to be based upon the distance of nodes from the center, although not as useful as it could have been, if it also had separated the more dense, blue cluster into other, smaller clusters. However, this result is to be taken with a grain of salt, as it was already mentioned that *BIC* is not a suitable measure to separate clusters, however it was all that was provided from the package used.

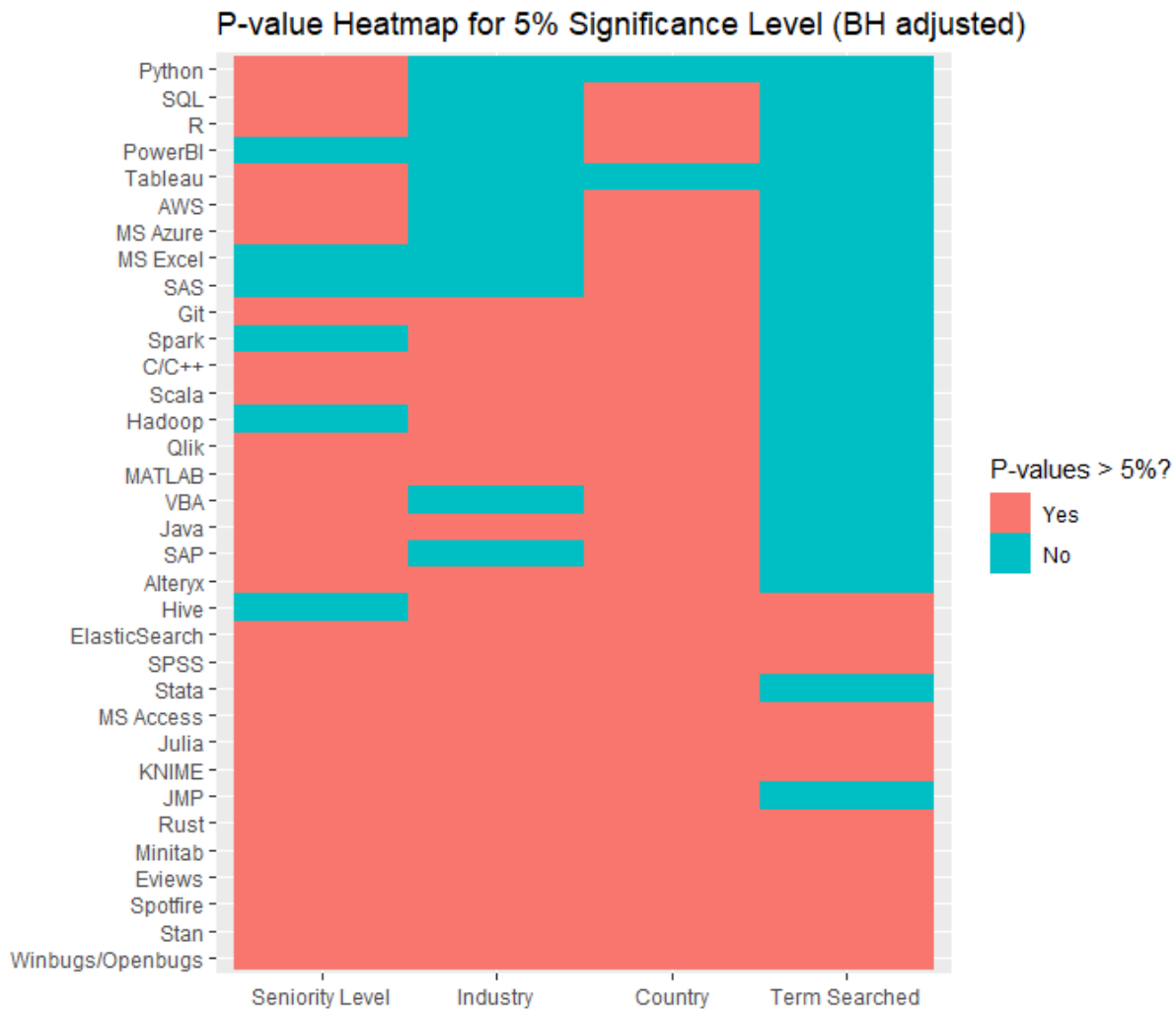


4.12 Testing for Statistical Significance

Our variables are nominal, categorical variables and thus should one need to test the statistical significance between them, then the standard thing one could do would be a *Chi-Square Test*.

As it was explained in 2.1.3 *Testing for Statistical Significance* a *Chi-Square* test for independence between nominal variables shall be performed, using the *MCMC* algorithm to estimate the *p-values*, because some variables are populated sparsely by data for the levels that they have.

In total there are 136 tests to conduct. Below is a heatmap that synopsizes the results of these tests, as far as *p-values* are concerned at a statistical significance level of $\alpha = 0.05$.



Graph 22: A heatmap that depicts the results of 136 Chi-Square Tests between each of the 34 statistical tools (that are present on the y-axis) and the 4 grouping variables (that are shown on the x-axis) for a significance level of $\alpha = 0.05$. The y-axis was ordered based on frequency.

The hypotheses which were checked at each iteration, were:



$H_{0(ij)}$: The variables ST_i and GV_j are independent of each other.

$H_{1(ij)}$: The variables ST_i and GV_j are related to one another.

where:

- ST_i : Is the *Statistical Tool* i , for $i = 1, 2, \dots, 34$. And it includes all the statistical tools that were monitored in the dataset except for JAGS, as it was uninformative.
- GV_j : Is the *Grouping Variable* j , for $j = 1, 2, 3, 4$. Which includes the so-called grouping variables that were observed during data collection, meaning “*Seniority Level*”, “*Industry*”, “*Country*” and “*Term Searched*”.

All the p -values of the hypotheses above have been corrected through the Benjamini – Hochberg Procedure (as described in 2.1.5 *Benjamini-Hochberg Multiple Hypothesis Testing Procedure*).

As always, if the p -value is lesser than or equal to the significance level α that was set (here it was 5%), then the null hypothesis (H_0) is rejected, otherwise, we fail to reject the null hypothesis.

Specifically, looking at the first three grouping variables from left to right, one can see that only a handful of statistical tools were characterized dependent with them according to the results of the *Chi-Square* test for statistical significance level of $\alpha = 0.05$. For example, the “*Seniority Level*” variable was found to be related to SAS, PowerBI, Hadoop, Spark, Hive and finally MS Excel. Was it surprising that Python or SQL was not included? Not as much, because during the process of collection of data, it was apparent that both of these tools were present in all the levels of that variable, and thus were thought to be independent to its levels.

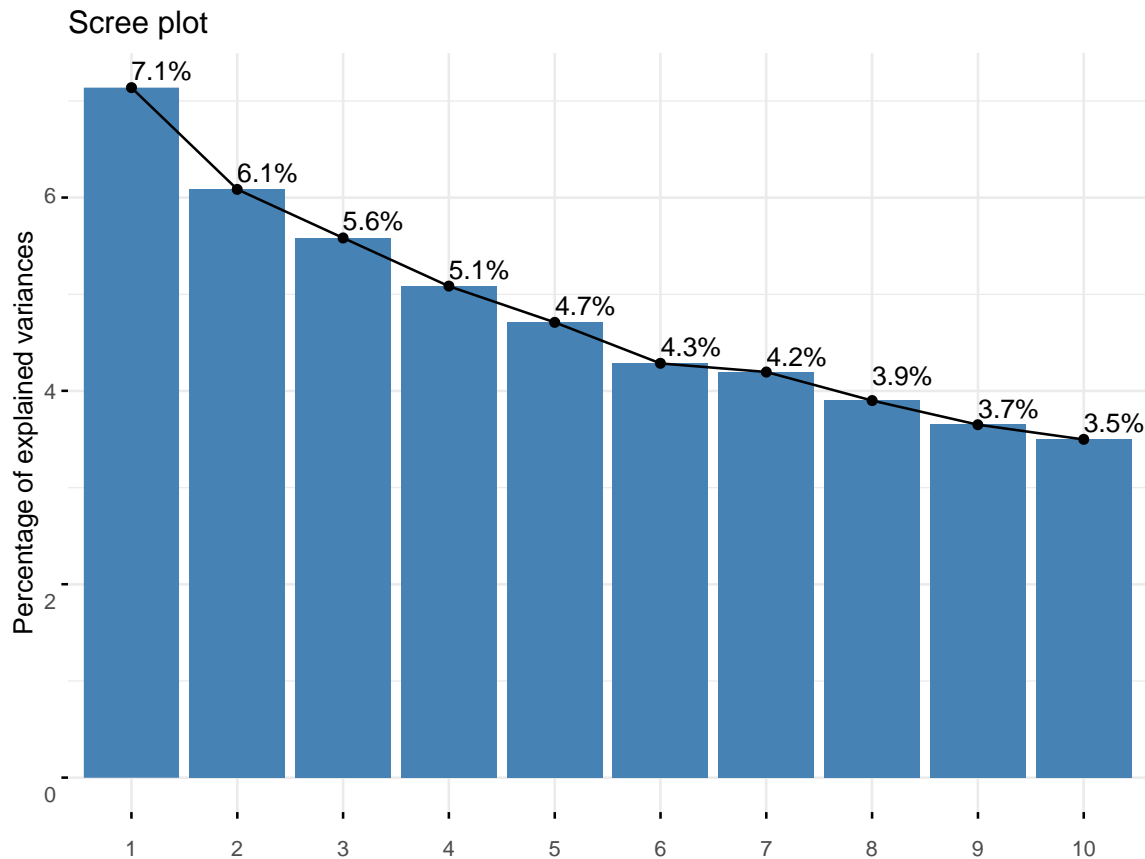
The exact opposite can be said about the last grouping variable, as most of the statistical tools were found to be dependent with it. Here the most significant tools were characterized as dependent to this variable. For instance, SAS which clearly defined the role of a Statistician or Biostatistician was included in the list of dependent statistical tools, which makes sense.

Generally speaking, it can be seen from *Graph 22* that the most frequent statistical tools were almost always dependent to the grouping variables, and those that were not as frequent, were mostly considered independent.



4.13 Multiple Correspondence Analysis (MCA)

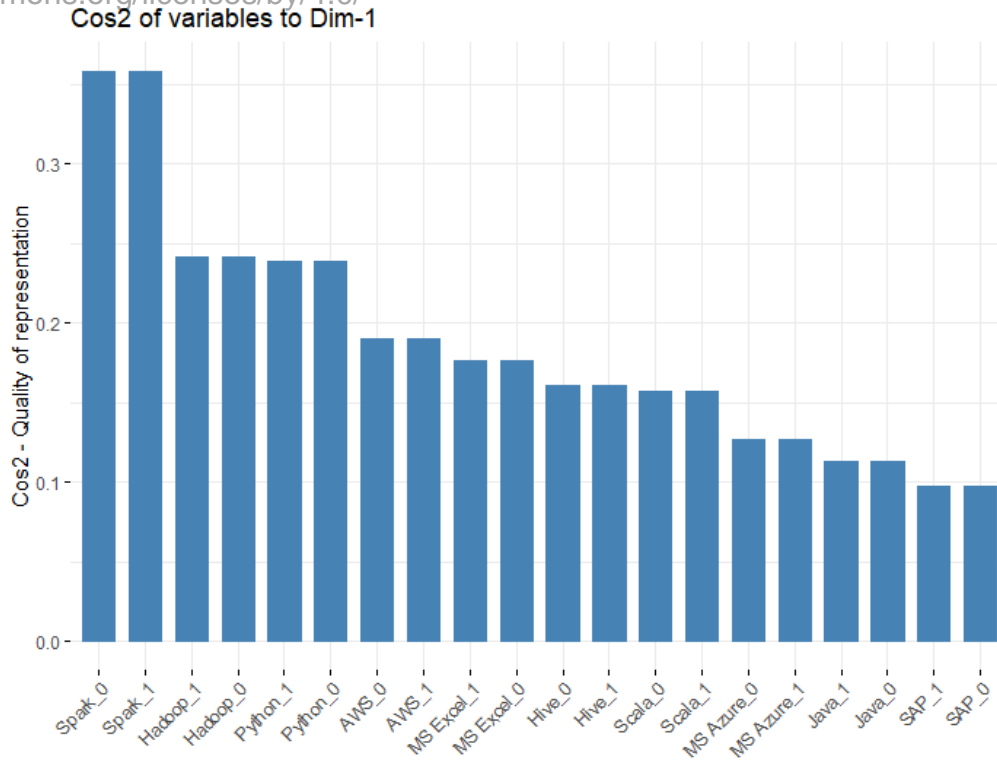
Another technique to analyze, detect and depict hidden structures within our data, is the so-called Multiple Correspondence Analysis (*MCA*). If we try to depict all of our statistical tools through it, these are the results that we shall get:



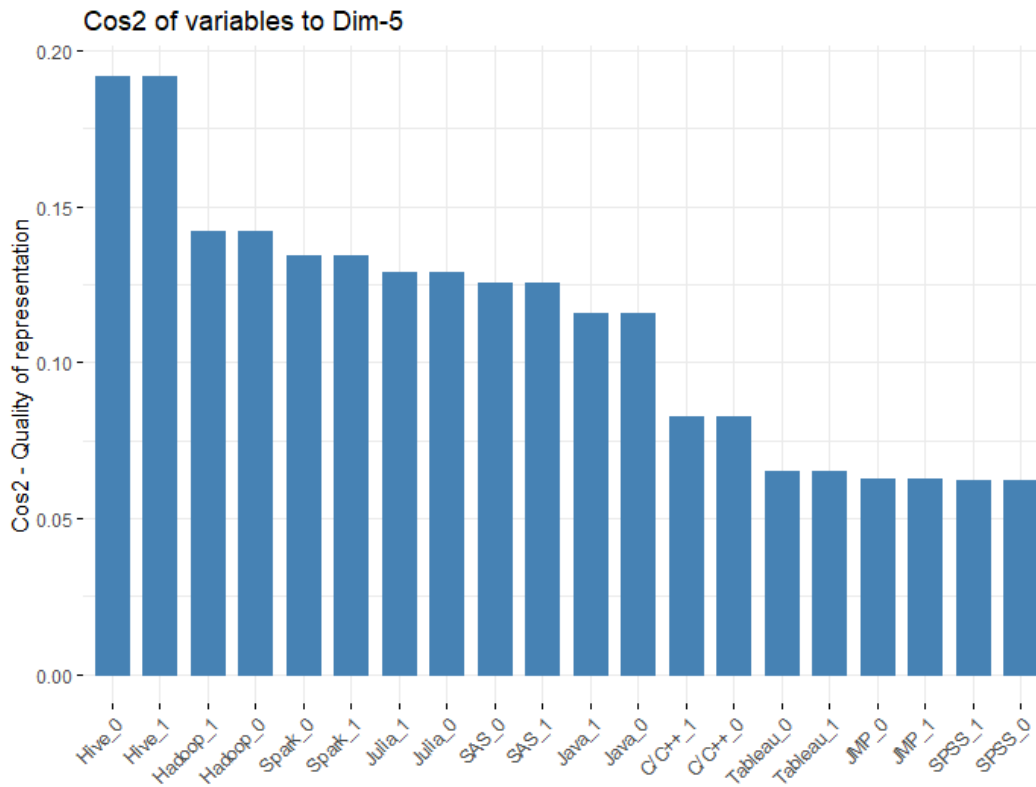
Graph 23: Scree-plot of all 34 statistical tools present within our data.

Immediately from *Graph 23* we see that these 10 dimensions are arguably not enough, as they are interpreting around 48% of the variability within our dataset. It means that should we need to adequately explain the variability, we might need even more dimensions to do so. That problem was of course obvious, as it was explained (in *2.1.4 Multiple Correspondence Analysis*) that unlike PCA, this technique studies a more general relationship between the variables.

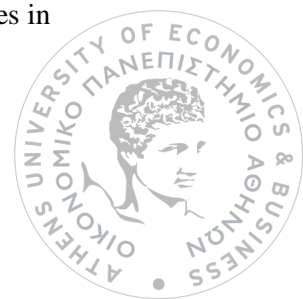
This problem is even more apparent if we plot the diagrams that show how well each dimension represents each variable within it, which we will do right below for dimensions 1 and 5:



Graph 24: A plot that shows the quality of representation of the 10 best represented variables in dimension 1.



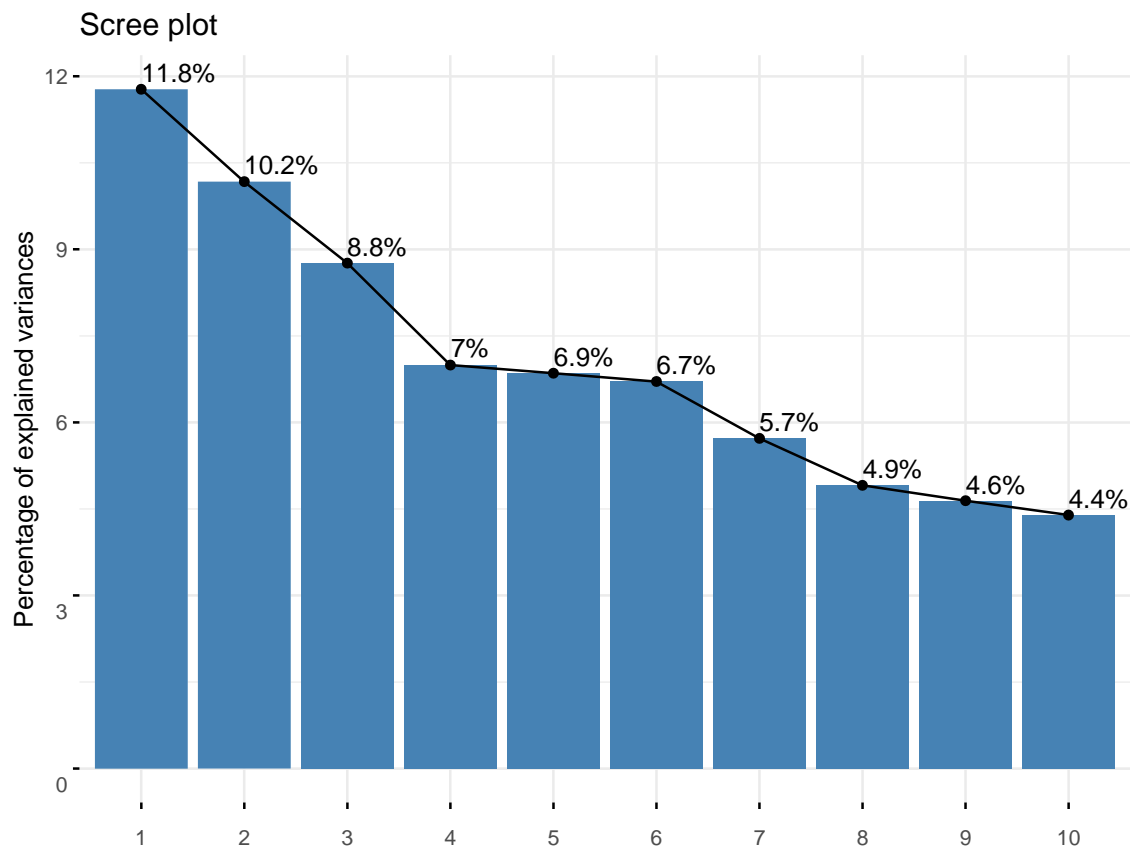
Graph 25: A plot that shows the quality of representation of the 10 best represented variables in dimension 5.



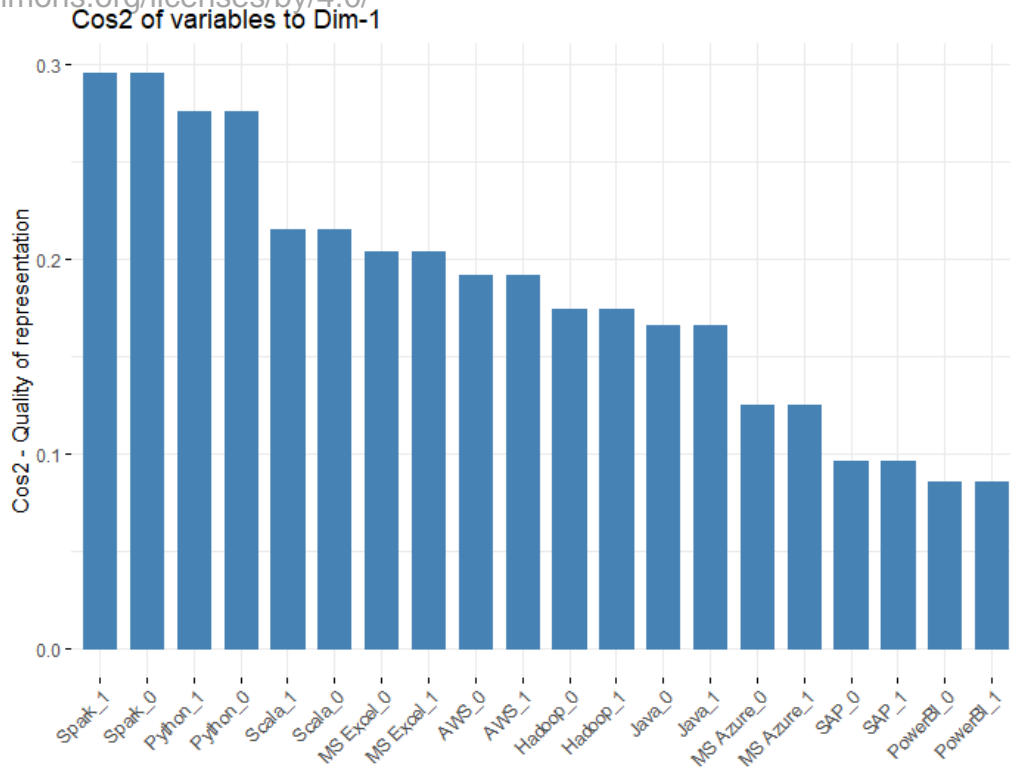
Since we have 34 statistical tools in total, it gradually becomes harder to represent them all in just a few dimensions. The problem is apparent firstly from the Scree plot (*Graph 23*) and afterwards from *Graph 24* and *Graph 25* right below it. Although the contribution of the top variables towards forming the first dimension is okay, it is problematic that the quality of representation of those variables is this low. Especially if it is generally considered that anything below 0.2 is not represented very well. What we expect; and is apparent in *Graph 25*, is that at higher dimensions, these statistical software will lose in the quality of representation. We can see in *Graph 25* that the quality has already fallen below the 0.2 threshold and although dimension 5 does mainly represent Hive (and maybe Hadoop), they are not represented very well, which also means that they can be found in perhaps similar proportions in other dimensions as well.

What we have to understand is that Multiple Correspondence Analysis (*MCA*) is strongly affected by all of its data and thus if there are rare observations here and there, then it will of course take them into consideration at the cost of the quality of representation of the data. Therefore, what we could do is look at the Multiple Correspondence Analysis (*MCA*) of the most popular statistical tools and ignore some that are kind of obscure.

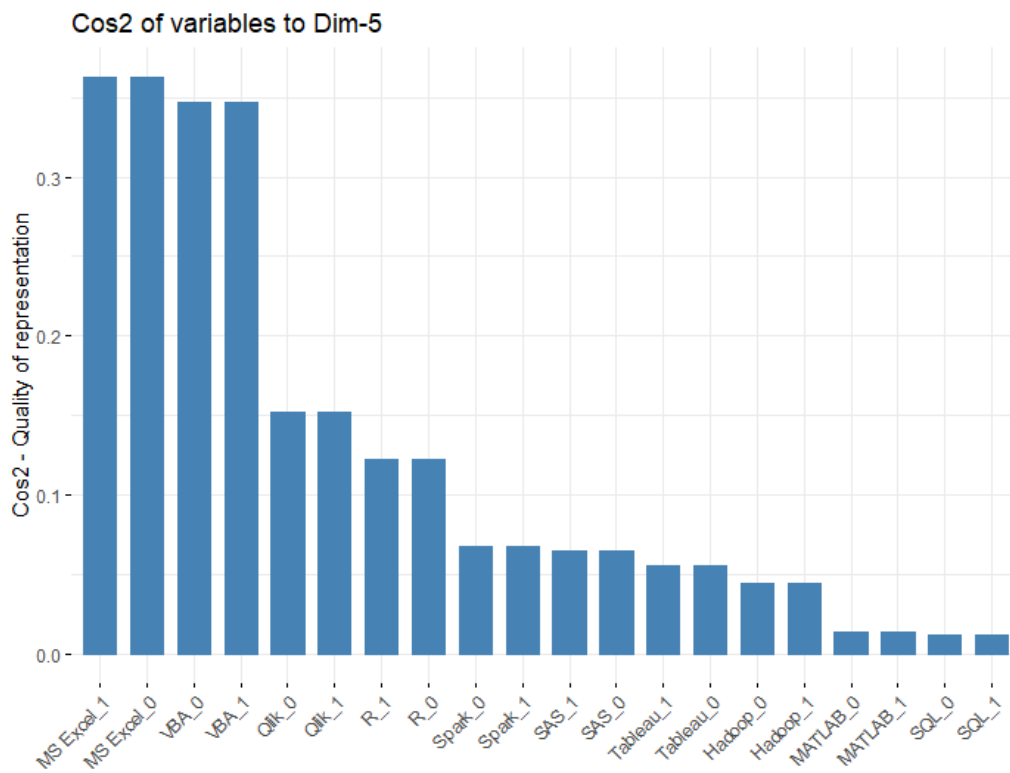
Therefore, let us then consider anything that had a frequency of 20 or less as trivial and apply *MCA* for the rest. If we do that, then we get:



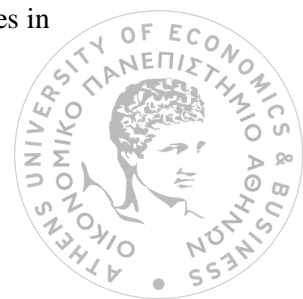
Graph 26: Scree-plot of the 19 most frequent statistical tools in our dataset.



Graph 27: A plot that shows the quality of representation of the 10 best represented variables in dimension 1.

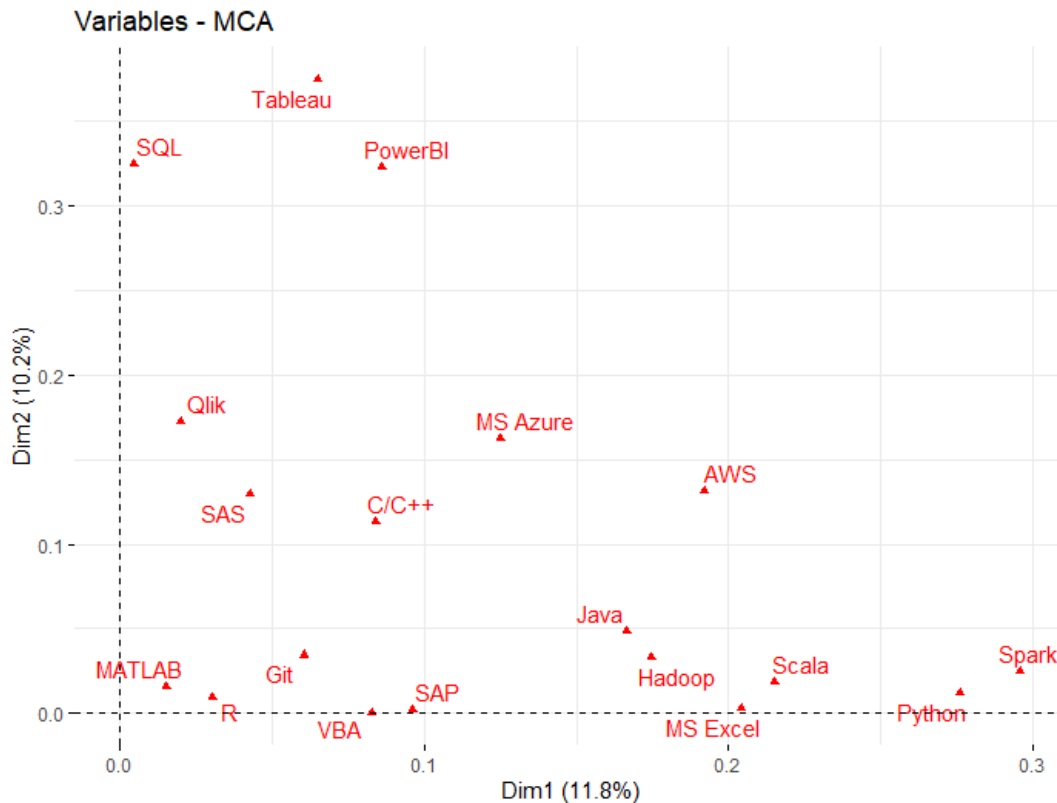


Graph 28: A plot that shows the quality of representation of the 10 best represented variables in dimension 5.



Here the program chose 19 statistical tools to be analyzed and although not perfect, it still is an improvement on the previous results. The Scree plot in *Graph 26*, tells us that in total, the 10 dimensions can account for approximately 71% of all the variability of our data and that perhaps 8 dimensions are enough to describe our data adequately. *Graph 27* shows that the first dimension is an adequate representation for Spark, Python, and maybe, Scala, while *Graph 28* tells us that the fifth dimension is mostly a good representation of MS Excel and VBA.

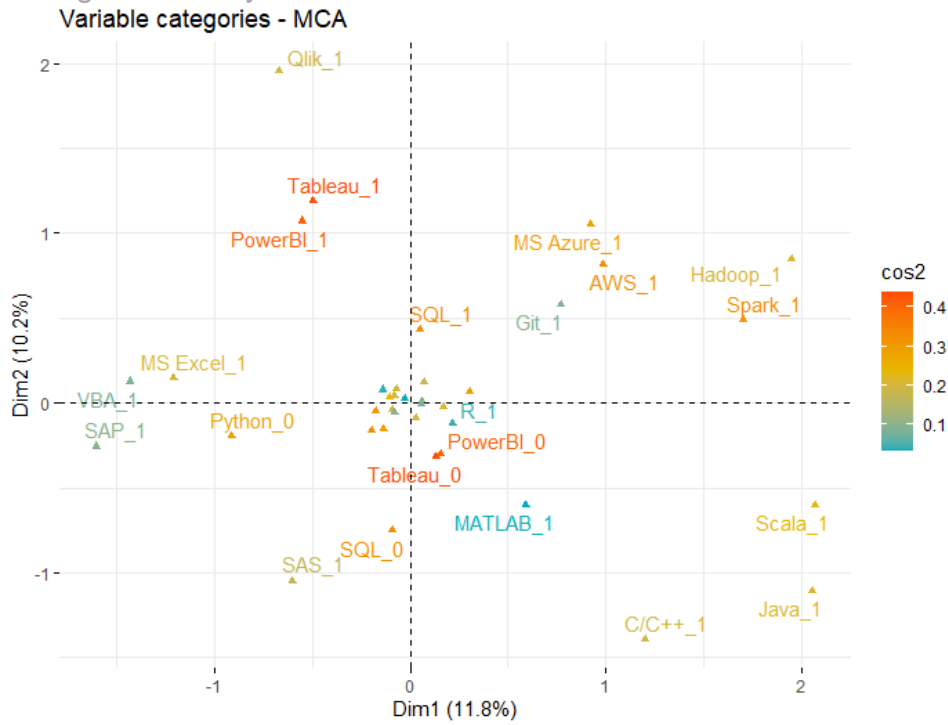
Let us now see how variables relate to each other, as far as their correlation to each other is considered:



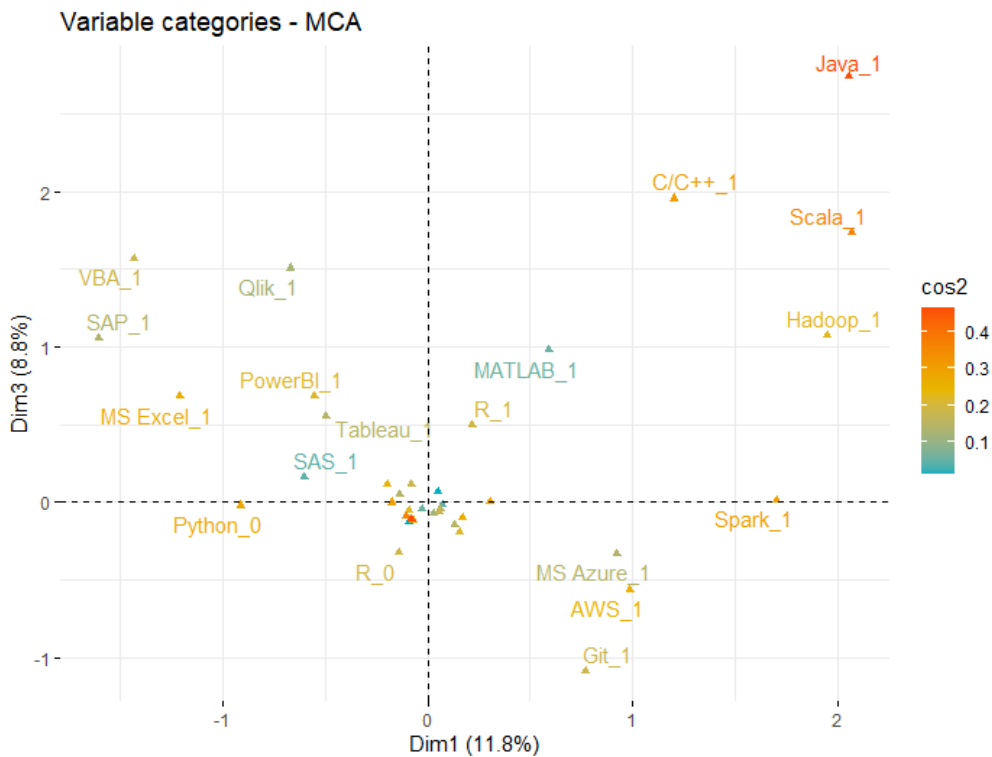
Graph 29: A plot of correlations between the 19 most frequent variables presented in two dimensions

In *Graph 29*, points that are close to each other are considered to have a stronger relationship than if they are further apart. Therefore, Python and Spark seem to have a close relationship in our dataset, and so are SQL, Tableau and PowerBI that were seen to be requested together lots of times, sometimes even as alternatives to each other. And so it is as well for the rest of the points within this dataset. There do not seem to be any surprises here, except for maybe that SQL and Python are so far apart, although we must consider that SQL's function differs greatly to the one Python does to the data and perhaps has more things in common with Tableau and PowerBI that are used in data analysis and data management.

Let us see how well these variables are represented within the first five dimensions (keeping dimension 1 constantly on the x-axis for all the subsequent graphs):

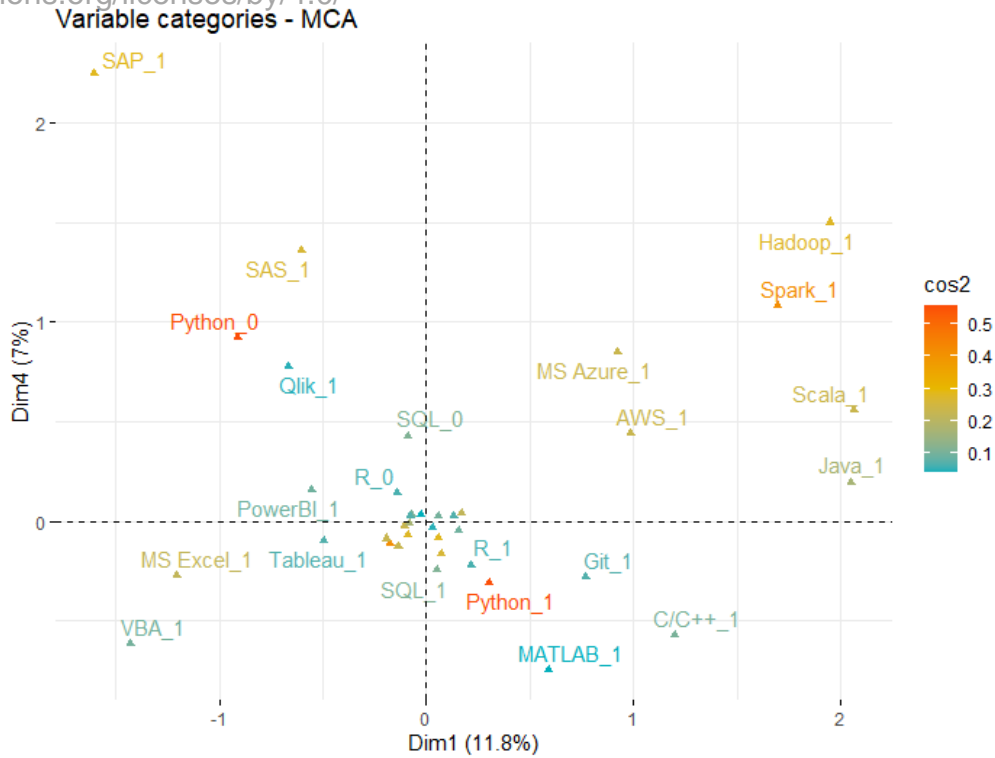


Graph 30: A plot of variable categories between dimensions 1 and 2, coloured by their quality of representation.

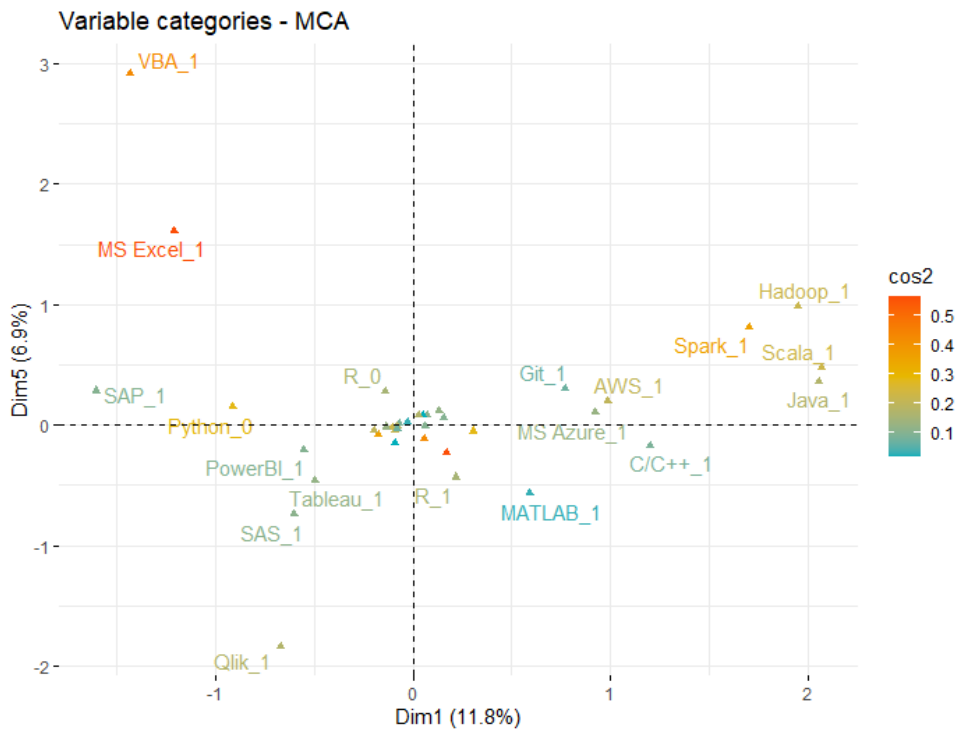


Graph 31: A plot of variable categories between dimensions 1 and 3, coloured by their quality of representation.





Graph 32: A plot of variable categories between dimensions 1 and 4, coloured by their quality of representation.



Graph 33: A plot of variable categories between dimensions 1 and 5, coloured by their quality of representation.



As it can be seen from *Graph 30*, dimension 1 represents well the presence of Spark and perhaps Scala and the absence of Python. This means that mostly Spark appeared whenever Python was not present in job advertisements. Dimension 2 in that same diagram, represents adequately SQL, Tableau and PowerBI, both their presence and absence from the dataset. It means that in the majority of cases, these statistical tools were seen together in a job description, even as alternatives in case of Tableau and PowerBI. Even though the other tools are not represented too well, we can see that AWS, MS Azure and Git appear to be similar, as well as MS Excel and VBA, which makes sense considering that they were seen together many times during the collection of data.

Moving on to *Graph 31*, we see that dimension 3 represents Java and C/C++ decently as well as adding some information about Scala. In general, MCA does not fully represent all the statistical tools by one dimension or another. Much like in this new dimension, there appears to be an elevation of the *Scala_1* point (that represents the presence of Scala in our data), which means that it is found in both dimensions 1 and 3. Therefore, Java is represented by both these dimensions and the fact that it is close to C/C++ and Scala means that it is somewhat similar to these two statistical tool, whenever they are present in the dataset.

In *Graph 32*, one can see that dimension 4 best represents Python and SAS. It tells us that whenever Python was absent, it was more likely to see SAS in the required set of skills. But it is also apparent that it was more likely to see Python along with R and SQL, as they are linked, but not as strongly as the other variables in these two dimensions.

Finally, looking at *Graph 33*, it can be seen that MS Excel and VBA are well represented by dimension 5 and they are close to each other, meaning that they are quite similar. That stands to reason as VBA is part of Excel and so it is logical to see those two programs appear together.

Even in this reduced dataset with 19 variables, the separation is not very clear, but that is to be expected as different job positions have different demands and therefore might need an odd mixture of statistical tools to meet them.



CHAPTER 5: CLUSTERING FOR BINARY DATA

5.1 Hierarchical Clustering

Clustering is only a part of Machine Learning and is called Unsupervised because it has no response vector to refer to. What it is, is a process that seeks a grouping of data (if it exists) into an (unknown) number of classes from a given dataset. The goal of such algorithms is to identify latent groupings of observations, which allow us to predict the class of observations even without a labelled response vector.

Here, our data are binary, meaning that it consists of 0 and 1, while our variables are nominal, categorical, or dummy variables and therefore whatever clustering can be done will be quite limited.

Hierarchical Clustering for binary data is not an easy thing to do. First one must choose a suitable distance measure so as to interpret the similarity between the data (which is discussed in 2.2.1 *Hierarchical Clustering*).

Why choose the Jaccard Distance?

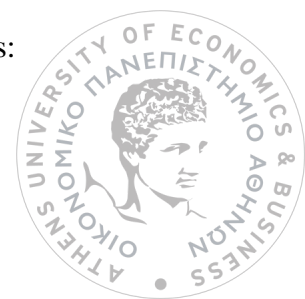
Should one take into account the zeroes when comparing how similar two observations are? Let us say that there are two jobs that are similar to each other, which means that they have the same arrangement of statistical tools that they use (Python, R, etc.) and let us suppose that VBA is missing from both of them. Now let us suppose that a new observation comes and has the same assortment of statistical tools, plus VBA. Shouldn't these three job descriptions be in the same category, despite this? After all, VBA could have been omitted from the other two job descriptions, but implied in the text and thus it would not be added in the dataset as 1, but as a 0 in both of these cases. Therefore zeroes should probably be omitted from the distance measure.

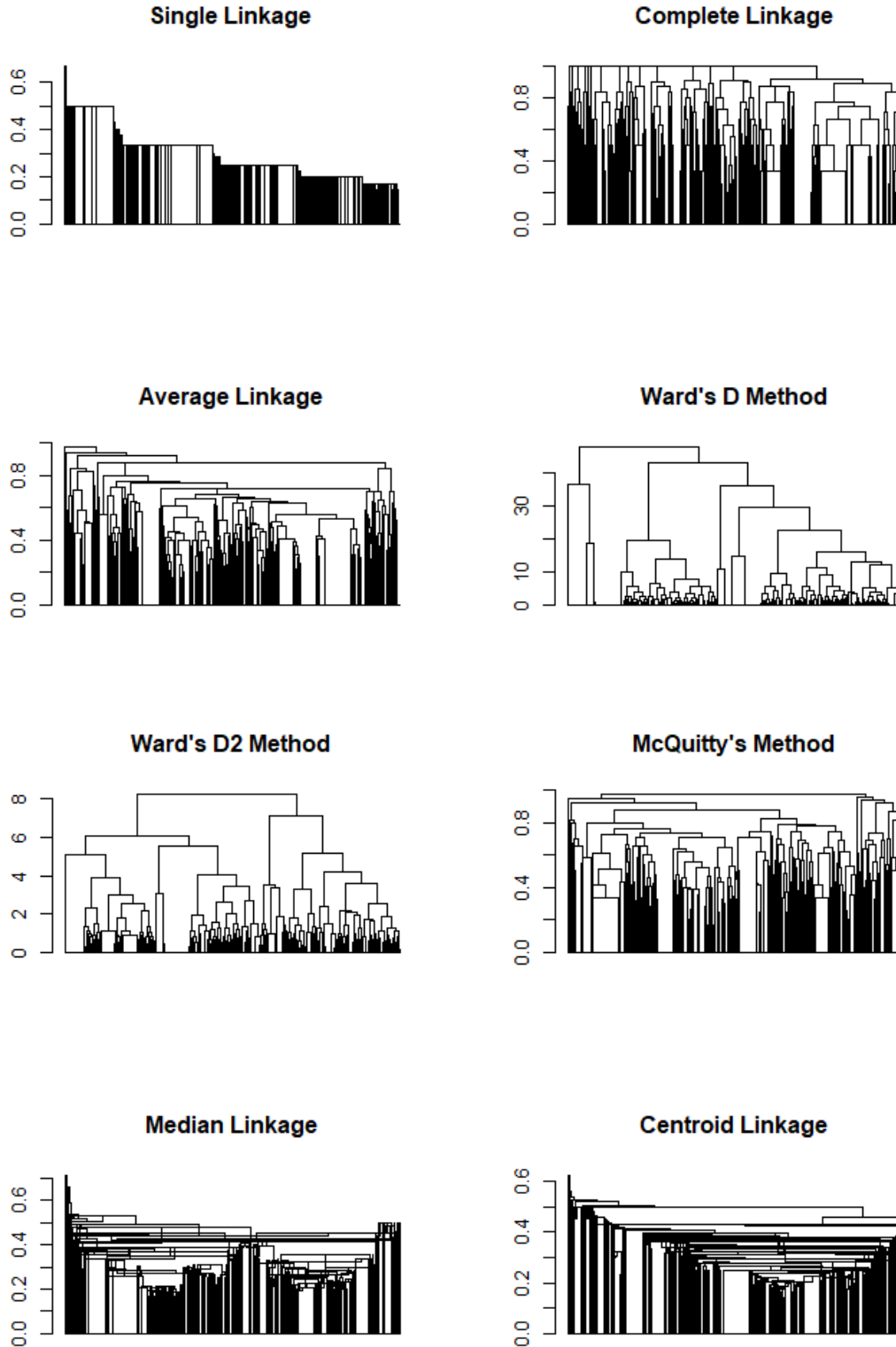
The ones should definitely be there, as they are rightfully in every distance measure, because more similar observations will definitely have the same tools present.

Now what about the dissimilarities? Should they be treated in a different way than the similarities of the dataset? That is to say, should we consider that two observations are further apart if at least one of the statistical tools are not present? If two observations are similar in their tools that they use, but one does not have, say R, should these two be closer than normal, or further apart? Perhaps in this case it is not prudent to tinker with the dissimilarities and it would be more logical to leave this measure as is, as it would be pretty hard to detect clusters if these dissimilarities were too close and it would not be too reasonable to put a greater distance between them and the similarities.

In conclusion, what we need is a distance measure that handles the similarities and dissimilarities in a similar manner, while also not relying on the common zeroes of the observations, which is the *Jaccard Distance*.

Let us then plot the dendrograms for the Jaccard distance for all the available methods:





Graph 34: Dendrograms for the *Jaccard Distance*, of all the available methods.



What we need from *Graph 34* are the general shapes of the dendrograms, as it would be unreasonable to make sense of all the 1000 observations. What we can therefore see, is that not all methods produced coherent results. Single Linkage Method seems to have an issue, with the so called: “*chaining effect*”, which means that it does not form proper clusters, but merges chunks of observations into many small clusters. Median and Centroid Linkage Methods produced bizarre shapes that do not resemble clusters at all, while the Average Linkage Method seems to have some outliers that for a cluster of their own. Apart from that, the other methods appear to have tried to form some sort of logical connections between the observations. Just by looking at the coherent dendrograms of *Graph 34*, we can see that our data are separated either into 2, 3, 4, 6 or 16 clusters.

Let us try to discern which ones of these are more logical separations, by applying some indices that shall indicate the optimal number of clusters per each of the coherent methods (as discussed in 2.2.5 *Indices for Detecting Clusters through Hierarchical Clustering Methods*):

Optimal Number of Clusters					
Indices	Methods				
	Complete	Average	Ward’s D	Ward’s D2	McQuitty
Ratkowsky-Lance	16	31	4	4	24
Friedman-Rubin	384	384	384	384	384
Davies-Bouldin	384	384	384	384	384

Table 11: A table that shows the optimal number of clusters chosen by each of the three indices per method chosen. The maximum number of clusters was set at 500.

As it can be seen from *Table 11* above, only the Ratkowsky-Lance index functioned as it should, as the other indices stopped at 384 clusters, when their index value reached zero. Therefore, we should be looking only at row 1, where it can be seen Ward’s D and Ward’s D2 Methods chose 4 clusters, the Complete Method chose 16, McQuitty’s Method chose 24 and finally, the Average method chose 31 as the optimal number of clusters.

In conclusion, it appears that in this case, clustering does not come easy for *Hierarchical Clustering* and although it does give us some results, they are not as concrete as we would hope.

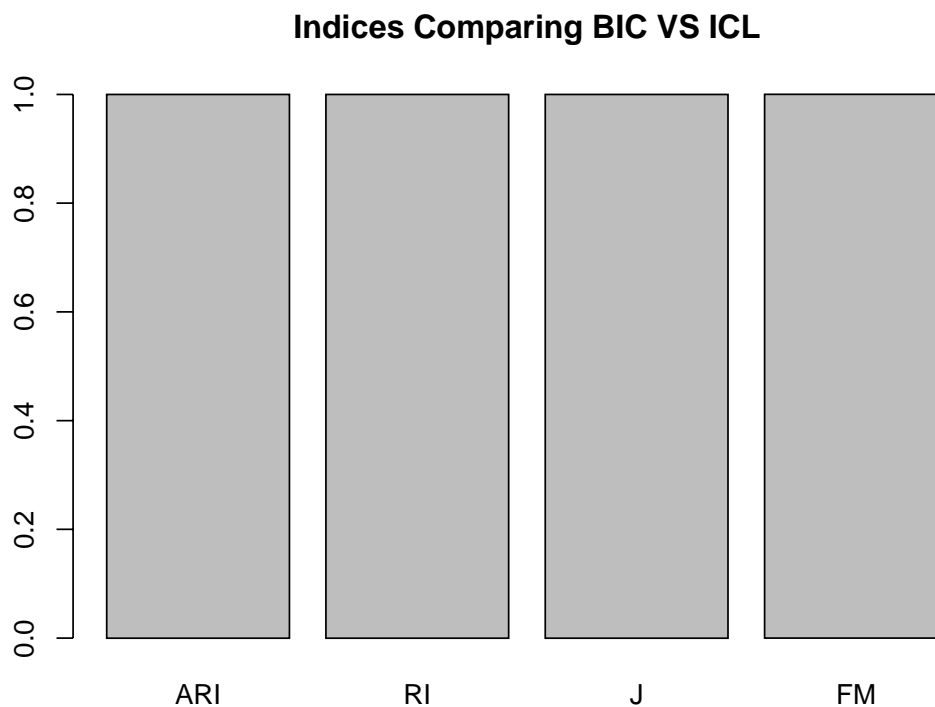


5.2 Finite Mixture Models for Model-Based Clustering (FlexMix)

One of the applications of finite mixture models is their ability to detect clusters within a group of data.

The package that is used in R (*FlexMix*), gives the users an opportunity to choose the best model however they see fit. Therefore, one could say that among the options present, they could use Bayes Information Criterion (*BIC*), or an even better option would be the Integrated Completed Likelihood (*ICL*) criterion, which has been shown to be more robust (P. Papastamoulis, M.-L. Martin-Magniette, and C. Maugis-Rabusseau, 2016; C. Biernacki, G. Celeux, and G. Govaert, 2000).

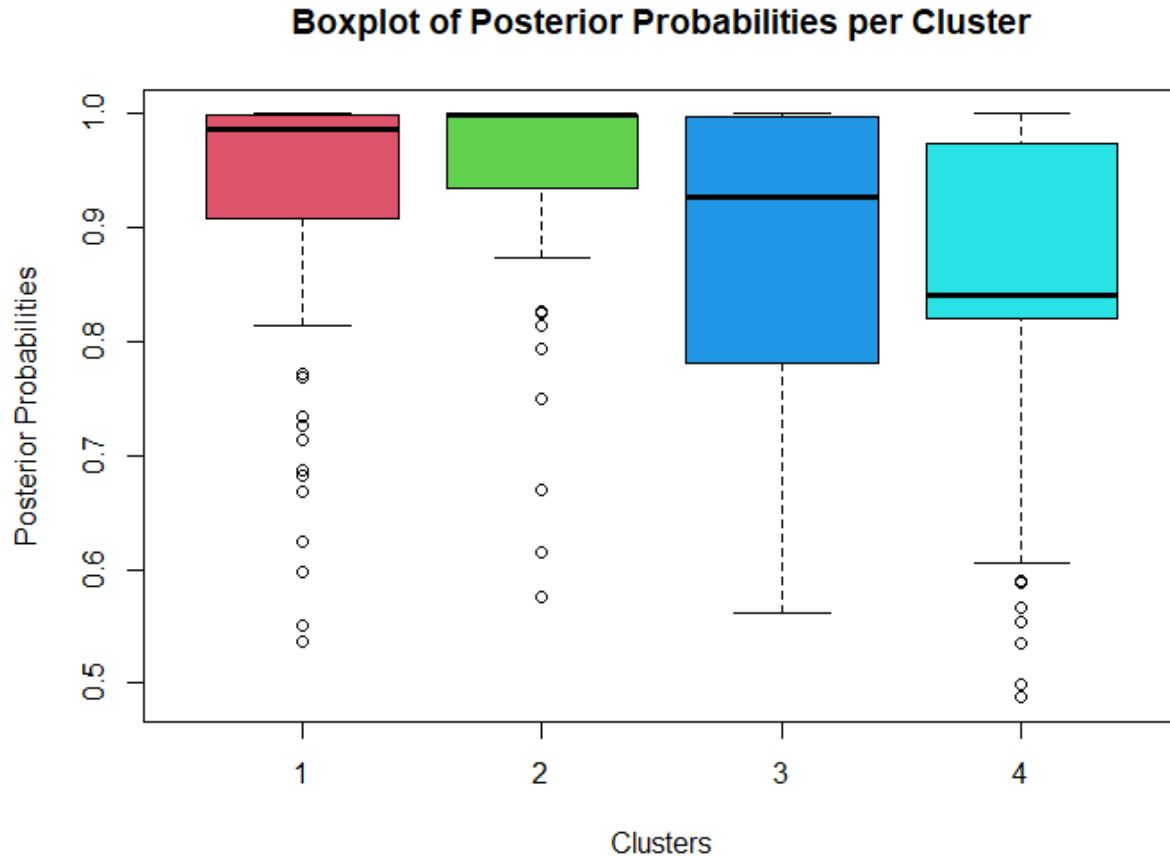
For our dataset, it does not matter much, as after many repetitions of the algorithm (around 100), they both reach the same conclusion as to the optimal number of clusters. In order to show just that, let us plot a few indices that show how similar are the two methods used:



Graph 35: A bar-plot of indices comparing the two partitions of clusters formed by choosing the best model through BIC and through ICL.

Therefore looking at *Graph 35*, it is clear that these two methods (*BIC* and *ICL*) produce identical results as they have reached the same maxima.

After running this procedure for around 100 repetitions for each potential number of clusters, the results showed to favour 4 as the optimal number of clusters for our data. First of all, here is a plot of the maximum posterior probabilities in a boxplot per cluster chosen:

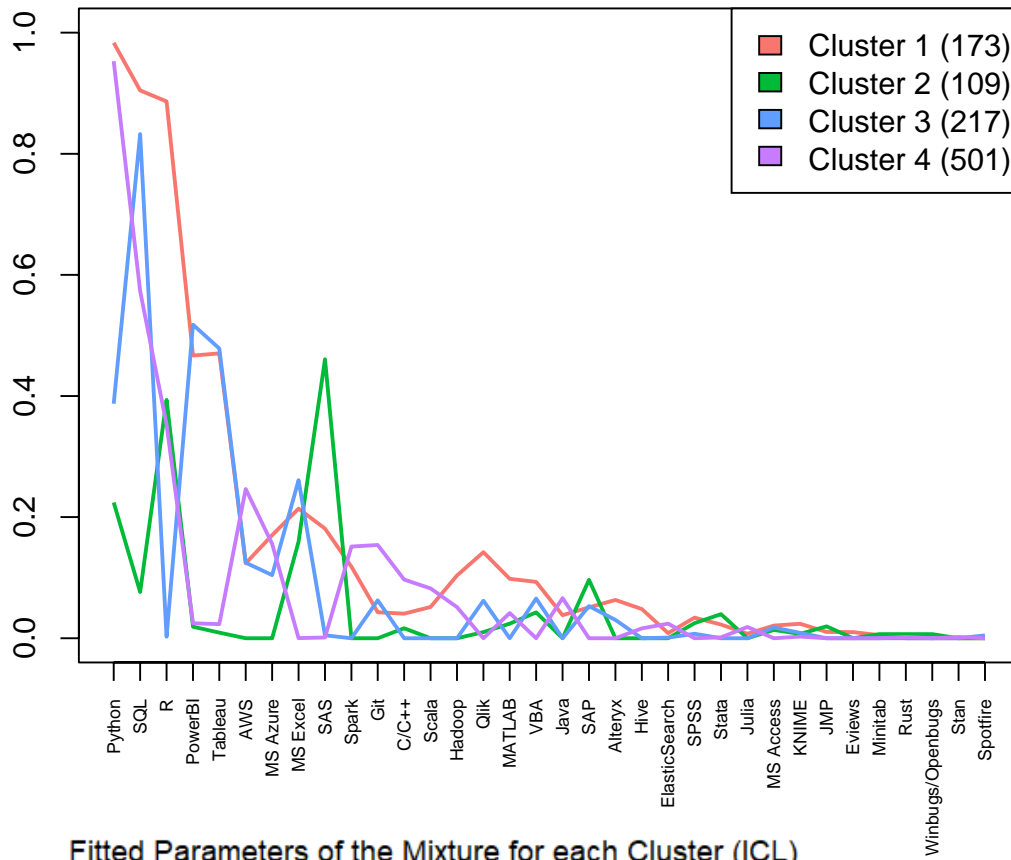


Graph 36: A box-plot that depicts the posterior probabilities for each of the 4 clusters.

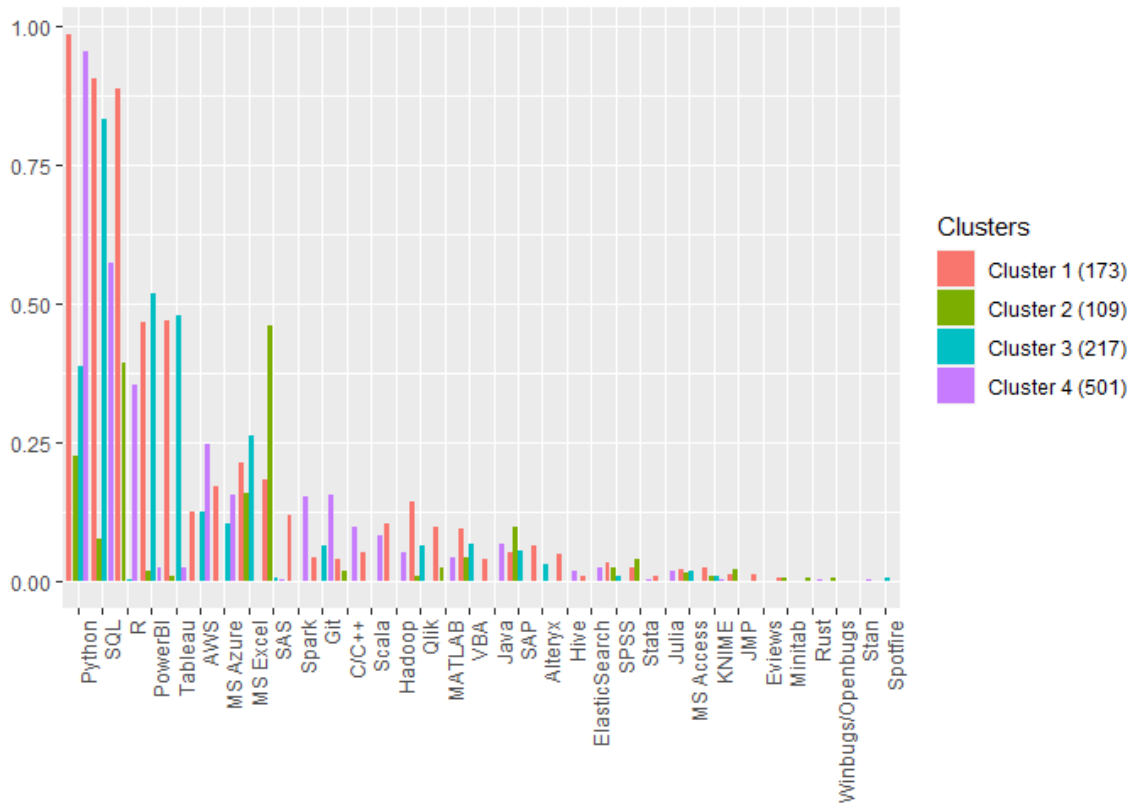
In this plot (*Graph 36*), we can see that the majority of clusters chosen were of a high probability, meaning that there were only a few second guesses whether an observation belonged to another cluster or not, but it was mostly definite that the assignment was correct. There are of course around 20 – 30 observations that seem to have had hard time discerning whether they actually belonged to one cluster or another. Furthermore, apart from the two observations of Cluster 4, which is the far right one (coloured in purple), none of these probabilities went lower than 50%. The most uncertainty seems to lie in the last two clusters, as the majority of observations in the first two are classified with high posterior probability per cluster (with a few outliers, of course).

The results of this clustering technique are summarized in the two diagrams below:

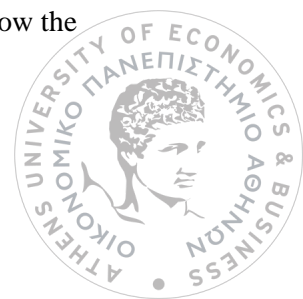
Fitted Parameters of the Mixture for each Cluster (BIC)



Fitted Parameters of the Mixture for each Cluster (ICL)



Graph 37: A line and a bar graph that both depict the fitted parameters of the mixture for each cluster. The statistical tools are ordered by frequency from most, to least frequent. The parentheses show the number of observations per cluster.



What can be gathered from the graphs above, is the following:

1. *Cluster 1*: The one that is coloured in red, dominates over others when it comes to the following statistical tools: Python, SQL, R, Hadoop, Qlik, MATLAB, VBA, Alteryx and Hive.
2. *Cluster 2*: The one that is coloured in green, dominates over others when it comes to the following statistical tools: SAS, Stata and SAP.
3. *Cluster 3*: The one that is coloured in blue, dominates over others when it comes to the following statistical tools: PowerBI, Tableau and MS Excel.
4. *Cluster 4*: The one that is coloured in purple, dominates over others when it comes to the following statistical tools: AWS, Azure, Scala, Git, Spark, C/C++, Java and Julia.
5. Clusters that had high demand of SQL, also had high demand in Tableau, PowerBI and a smaller demand in MS Excel (meaning Clusters 1 and 3). If either SQL, Tableau, or PowerBI was low in demand then the other two were most likely to be low as well (much like in Clusters 2 and 4). This was also observed consistently during the collection of the data, that there existed a grouping or at least an analogous relationship between of SQL, Tableau and PowerBI.
6. There appears to also be an analogous relationship between AWS, MS Azure and Git, as these three statistical tools were observed together during the collection of data and this is also reflected in Clusters 1, 3 and 4.
7. R and SAS seem to have a minimal relationship as well, as whenever SAS is present, it is also associated with big spikes in R as can be seen in Clusters 1 and 2, versus in Clusters 3 and 4, although not always.
8. Cluster 2 seems to be the most specialized of them all, because it mostly demands very specific statistical tools to be known from the candidate, which can be seen by the green line that majorly stays close to the x-axis.

In summary, looking at these clusters, it is highly probable that job advertisements that demand the knowledge SQL in their description, will probably, also mention Tableau, PowerBI and MS Excel, or one that refers to in AWS, in all probability shall also contain MS Azure, or Git and finally those job ads that mention SAS, will probably mention R as well. Now some might be mentioned as alternatives, like Tableau and PowerBI, but others will definitely be there as statistical tools that the candidate must know to do the job.



Word clouds for each cluster:

Next, we can take the clusters found by FlexMix and group them in a word cloud per each of the grouping variables (Country, Seniority Level, Industry and Term Searched), thus showing the categories in which they have the most observations. First up clusters for the “Country” variable:

United Kingdom



Graph 38: A word cloud for cluster 1, of the “Country” variable.

United Kingdom



Graph 39: A word cloud for cluster 2, of the “Country” variable.

United Kingdom



Graph 40: A word cloud for cluster 3, of the “Country” variable.

United Kingdom



Graph 41: A word cloud for cluster 4, of the “Country” variable.

Unfortunately, almost no interesting information can be gained when looking at this separation, except that maybe cluster 4 had the lowest number of observations in the other countries apart from the UK. United Kingdom was the cluster with the most observations, followed by Germany, Poland, Ireland and Sweden. Perhaps this means that it does not matter what country the candidates are, as they will probably use the same software.

Let us then continue onto the “Seniority Level” variable:



Graph 42: A word cloud for cluster 1, of the “Seniority Level” variable.



Graph 43: A word cloud for cluster 2, of the “Seniority Level” variable.



Graph 44: A word cloud for cluster 3, of the “Seniority Level” variable.



Graph 45: A word cloud for cluster 4, of the “Seniority Level” variable.

What *Graphs 42-45* tell us, is that cluster 1, which dominated over other clusters in Python, SQL, R Hadoop Qlik, MATLAB, VBA, Alteryx and Hive, had most observations in the Junior category, cluster 2, which was first in SAS, Stata and SAP, was mostly comprised of Associate, followed by Senior level positions, cluster 3 which was the best in PowerBI, Tableau and MS Excel was made of Associate, and followed by Entry-level positions and finally, cluster 4 that was dominant in AWS, Azure, Scala, Git, Spark, C/C++, Java and Julia was made of Associate and Mid-level positions.

This shows that there were very few jobs that demanded the knowledge of PowerBI, Tableau and MS Excel from Senior positions, or that SAS, Stata and SAP did not have much demand for Junior level experience from the candidates. It can be seen that Associate level positions are like the jacks-of-all-trades, they have a significant number of observations in each cluster. Whereas Senior level positions seemed to be more specialized in each cluster and mostly avoided tools that are used for data management and initial data analysis (which were dominant in cluster 3).

Next, let us see the same thing for the “Industry” variable:



Graph 46: A word cloud for cluster 1, of the “Industry” variable.



Graph 47: A word cloud for cluster 2, of the “Industry” variable.



Graph 48: A word cloud for cluster 3, of the “Industry” variable.



Graph 49: A word cloud for cluster 4, of the “Industry” variable.

From the *Graphs 46-49* one can see that the most dominant Industry in them all was IT, except for cluster 2, which was specialized towards Health. Therefore, it appears that Pharmaceutical and Healthcare companies have a preference towards SAS, Stata and SAP. IT, Finance and Software companies had the biggest demand for jobs that needed statistical tools. Apart from that, we can see that Cluster 1 was also big on Consulting, Gaming, Retail & Wholesale and Banking industries. Cluster 3 had more to offer to the Finance industry, while cluster 4 to the Software industry. Clusters 1 and 3 had tools that mostly needed a good programming knowledge to use, while Cluster 3 focused more on programs for data management and data visualizations that do not give much emphasis on a user’s programming skills.

Finally, let us depict the “Terms Searched” variable in these four clusters:

Data Analyst
Data Scientist
Statistician

Graph 50: A word cloud for cluster 1, of the “Term Searched” variable.

Data Scientist
Statistician
Data Analyst

Graph 51: A word cloud for cluster 2, of the “Term Searched” variable.

Data Analyst
Data Scientist

Graph 52: A word cloud for cluster 3, of the “Term Searched” variable.

Statistician
Data Analyst
Data Scientist

Graph 53: A word cloud for cluster 4, of the “Term Searched” variable.

As it is clear from the *Graphs 50-53*, Cluster 1 is oriented mostly towards Data Analysts and Data Scientists, Cluster 2 towards Statisticians, Cluster 3 towards Data Analysts and Cluster 4 to Data Scientists. It is curious that cluster 3 had no Statisticians at all. Apparently they do not prefer tools for data management and data analysis such as MS Excel, Tableau and PowerBI, like Data Analysts do, and instead use tools like SAP, SAS and Stata. Data Scientists clearly need tools like Python or R to do their job, but it is also surprising how big they are in Cluster 4 that includes cloud-based technologies as its most frequently witnessed, statistical tools, but perhaps it was once again the high number of Python job descriptions in the dataset that were present in Cluster 4 as well.

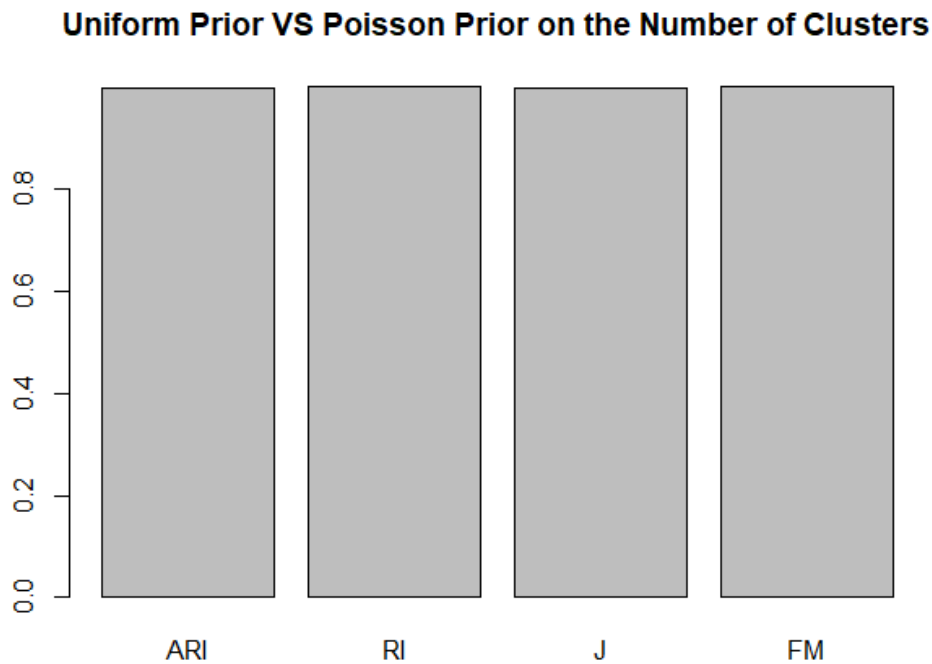
If we also compare *Graphs 46-49* to *Graphs 50-53*, we shall see that the Healthcare industry mostly asks for Statisticians, while Data Scientists and Data Analysts go mainly towards IT, Finance and Software industries.



5.3 Bayesian Clustering through Finite Mixture Models (BayesBinMix)

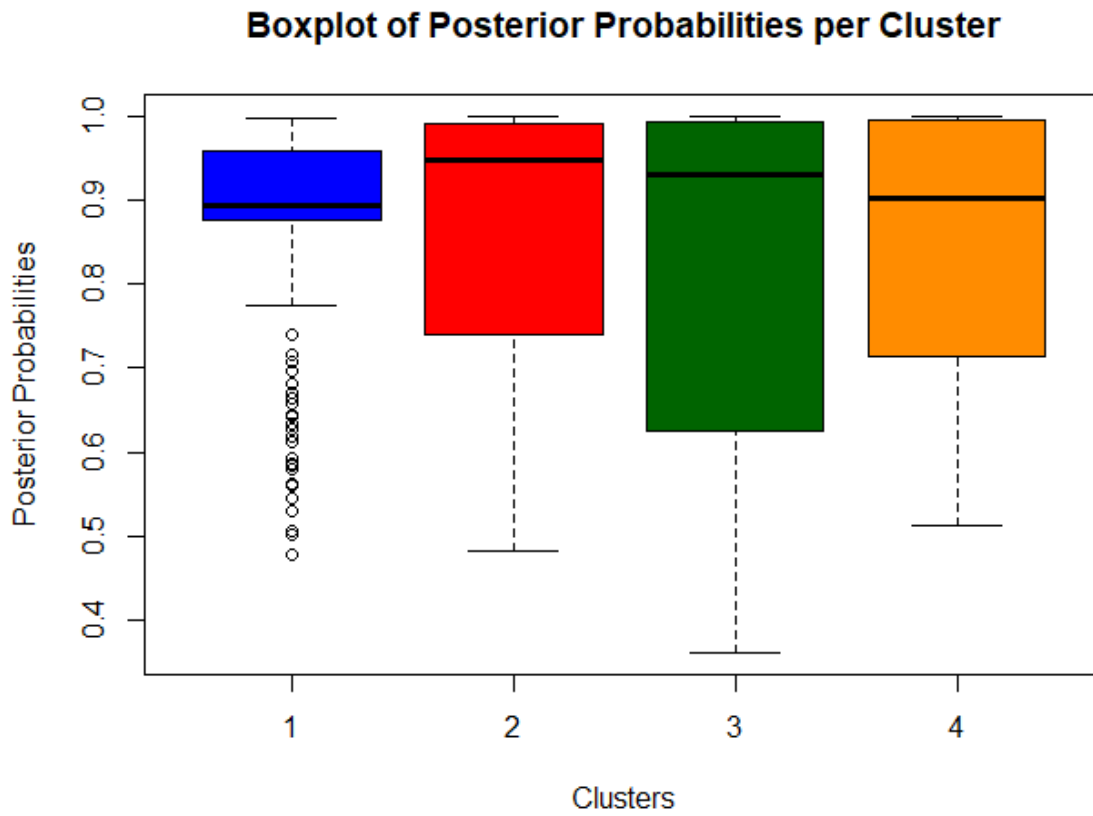
The previous approach had a minor disadvantage, which was that it could be stuck at a local maximum and never reach a global one, which happened in our case and thus, the number of repetitions had to be increased to 100, so as to get an adequate result. Moreover, it uses model selection measures like *BIC*, which is arguably not a suitable measure when choosing the optimal number of clusters. Thus, instead of using the *EM* algorithm, we could use *MCMC* sampling to come to the optimal number of clusters.

After applying the algorithm on our data, it was observed that this procedure was much slower than the one applied through the frequentist method (in the chapter right above, with the *FlexMix* package), in fact it took approximately 1-2 days for the algorithm to firmly decide which cluster it preferred. In the end, this program chose 4 clusters just like the one before it as the optimal number of clusters for our dataset. It did not matter whether we put Poisson Prior or Uniform Prior on the number of clusters, as they both end up with almost the same groupings (except for one observation that got grouped into another cluster), as can be seen from *Graph 54* right below.



Graph 54: A bar-plot of indices comparing the two partitions of clusters formed by choosing Poisson or Uniform Prior on the numbers of clusters.

Let us once again graphically represent the maximum posterior probabilities per cluster chosen, through a boxplot:

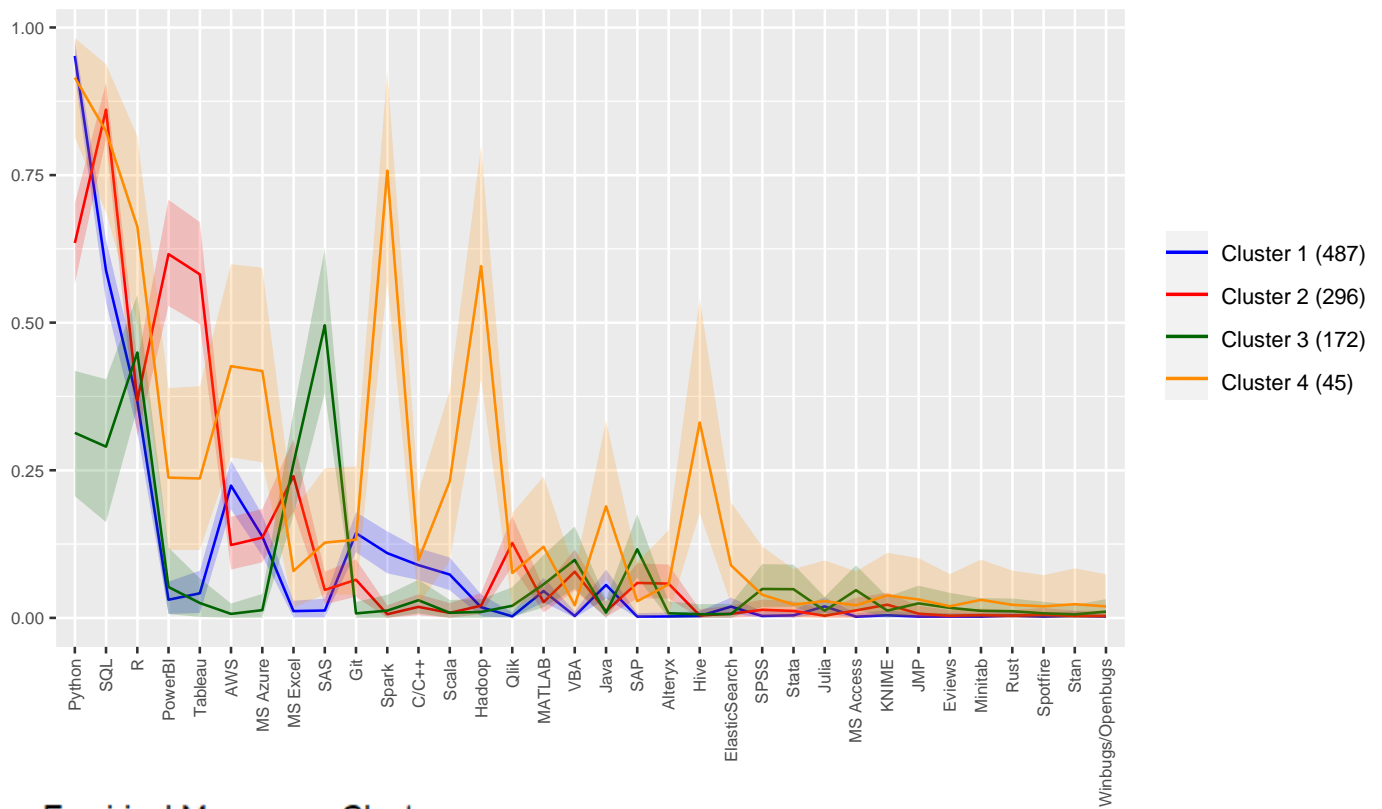


Graph 55: A box-plot that depicts the posterior probabilities for each of the 4 clusters.

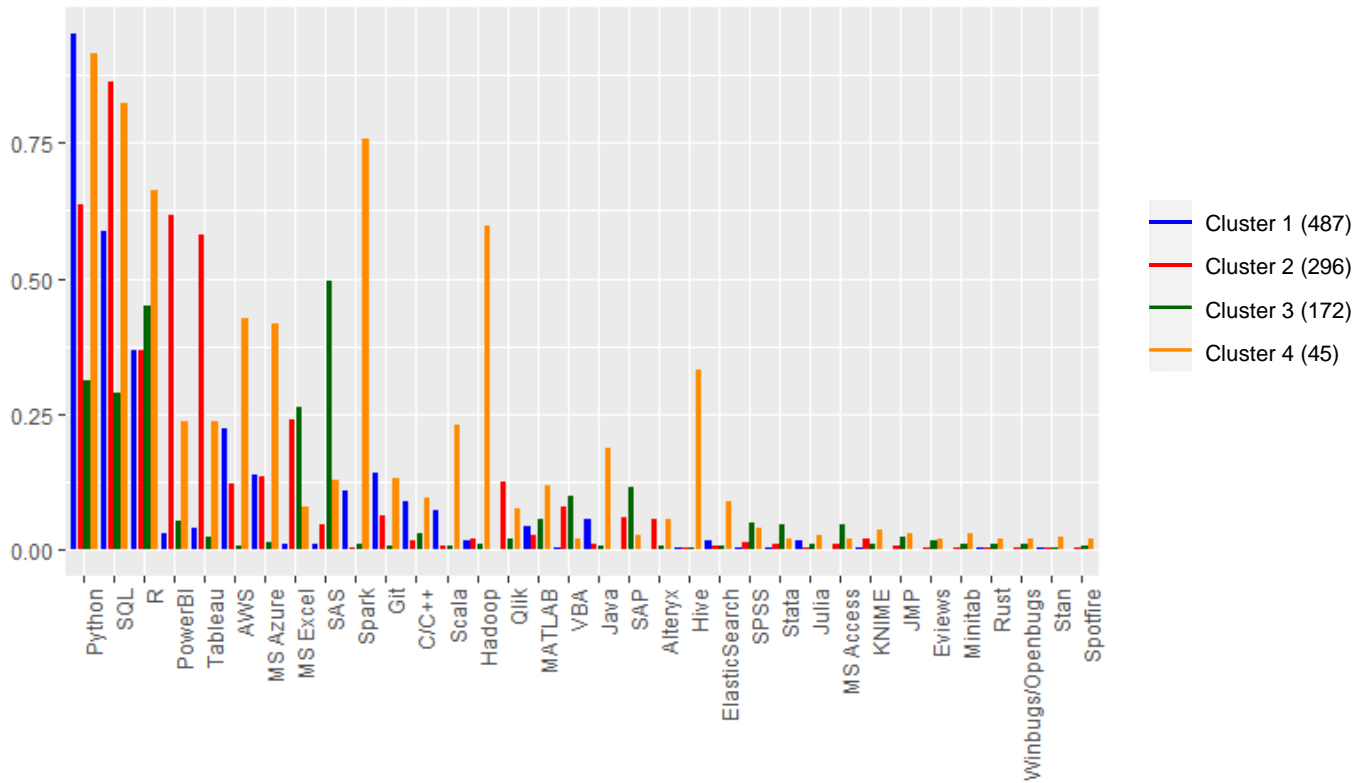
Looking at the median lines of *Graph 55*, we can see that the majority of clusters chosen were of a high probability, which was above, or around 90%. The only outliers appear in Cluster 1 (the one in blue colour), showing that it was perhaps the most certain of them all, apart from those 10-20 posterior probabilities that were out of the norm. Looking at the other clusters, it is apparent that it was a bit hard to be certain whether an observation belonged to one or another cluster, as many of the boxplots fall below 50% or even 40% in the case of Cluster 3, which account for a portion of the data contained in their lower quartiles. Although it was not as bad, it would be ideal to see them above at least 50% in these cases, in order to have some certainty when the algorithm chooses which cluster an observation belongs to.

The two following diagrams shall summarize the results of this clustering technique via empirical means and their 95% confidence intervals per cluster:

Empirical Means with 95% CI per Cluster



Empirical Means per Cluster



Graphs 56: A bar-plot and a line graph with 95% confidence intervals that both depict the empirical means of the mixture for each cluster. The statistical tools are ordered by frequency from most, to least frequent. The parentheses show the number of observations per cluster.



What can be seen from the graphs above (*Graphs 56*), is the following:

1. *Cluster 1*: The one that is coloured in blue, dominates over others when it comes to the following statistical tools only in Python and Git.
2. *Cluster 2*: The one that is coloured in red, dominates over others when it comes to the following statistical tools in: SQL, PowerBI, Tableau and Qlik.
3. *Cluster 3*: The one that is coloured in dark green, dominates over others when it comes to the following statistical tools in: MS Excel, SAS, VBA, SAP, SPSS, Stata and MS Access.
4. *Cluster 4*: The one that is coloured in dark orange, dominates over others when it comes to the following statistical tools in: R, AWS, MS Azure, C/C++, Spark, Scala, Hadoop, MATLAB, Java, Hive, ElasticSearch, Julia, KNIME, JMP, Minitab, Rust, Spotfire, Winbugs/Openbugs and Stan.
5. The most demanding cluster when it comes to empirical means of each program, is *Cluster 4* with the least number of observations, while the least demanding, is *Cluster 1* with the most observations.
6. Spikes in Tableau are frequently followed by spikes in demand for PowerBI and sometimes for SQL as well, something that can be confirmed in clusters 1,2 and 4.
7. AWS and MS Azure seem to be closely related. Looking at all clusters, when one of these tools appears in a cluster, so does the other.
8. SAS, Stata and SAP seem to have been organized in the same cluster yet again (*Cluster 3*), much like in the frequentist approach.
9. R, Python and SQL are present in all clusters, although in different quantities and have the highest demand overall.
10. High empirical mean for MS Excel meant high empirical mean for VBA, which was a bit expected, as these two are related. This can be seen in clusters 2 and 3.

It is thus organized in our data that if a candidate should run into AWS in a job add, it is highly probable that he will also see MS Azure. Or should they see SQL, then chances are that Tableau and PowerBI might follow. And of course, R, SQL and Python seem to be ever present.



Word clouds for each cluster:

Let us now organize these newfound clusters per grouping variable (meaning “Country”, “Seniority Level”, “Industry” and “Term Searched”), beginning with the “Country” variable:

United Kingdom



Graph 57: A word cloud for cluster 1, of the “Country” variable.

United Kingdom



Graph 58: A word cloud for cluster 2, of the “Country” variable.

United Kingdom



Graph 59: A word cloud for cluster 3, of the “Country” variable.

United Kingdom



Graph 60: A word cloud for cluster 4, of the “Country” variable.

Most of the observations are in the UK, as it was expected, because it had the most observations. All the countries that are a bit less bold than the United Kingdom, were the ones that had the biggest frequencies. Sweden is the second biggest country in Cluster 4, while Germany, in Cluster 2. All-in-all, this is a bit uninformative, as the groups show mostly the frequencies that were expected from the dataset.

Let us then continue onto the “Seniority Level” variable:



Graph 61: A word cloud for cluster 1, of the “Seniority Level” variable.



Graph 62: A word cloud for cluster 2, of the “Seniority Level” variable.



Graph 63: A word cloud for cluster 3, of the “Seniority Level” variable.



Graph 64: A word cloud for cluster 4, of the “Seniority Level” variable.

From the *Graphs 61-64* right above, we can see that the first cluster, which had the highest spikes in Python and Git, was addressing towards Associate to Mid-level experience positions. The second cluster was organized mostly around positions that needed at most 3 years of experience and its central tools that it used were: SQL, PowerBI, Tableau and Qlik. The third cluster was aimed mostly at job advertisements that needed some level of experience and mostly used MS Excel, SAS, VBA, SAP, SPSS, Stata and MS Access as its tools. And the fourth cluster was aimed at the most experienced positions of them all and was the first in: R, AWS, MS Azure, C/C++, Spark, Scala, Hadoop, MATLAB, Java, Hive, ElasticSearch, Julia, KNIME, JMP, Minitab, Rust, Spotfire, Winbugs/Openbugs and Stan. Once again, it seemed that Senior level positions mostly avoided tools that were needed for data management and data presentation, such as the ones in cluster 2, while Associate level positions had a bit of everything.

Next, is the “Industry” variable:



Graph 65: A word cloud for cluster 1, of the “Industry” variable.



Graph 66: A word cloud for cluster 2, of the “Industry” variable.



Graph 67: A word cloud for cluster 3, of the “Industry” variable.



Graph 68: A word cloud for cluster 4, of the “Industry” variable.

As it can be seen from *Graphs 65-68*, the IT industry was the first in most of the clusters, except for Cluster 3, where the Health industry took the first place among all observations. This cluster had MS Excel, SAS, VBA, SAP, SPSS, Stata and MS Access as the tools most used in it and therefore perhaps the Health industry makes use of the majority of those, much like in the frequentist approach through FlexMix, where it was SAS, SAP and Stata. Software was the next big thing in clusters 1 and 4 after IT and that had a lot of tools that needed programming knowledge in order to be used. While in Cluster 2, it was Finance, followed by Manufacturing industry and in this cluster, the most dominant tools were mostly programs for data management and data visualization, that did not need a lot of programming skills to be used.

Finally, let us also do word clusters for the “Term Searched” variable:

Data Analyst
 Data Scientist
 Statistician

Graph 69: A word cloud for cluster 1, of the “Term Searched” variable.

Data Scientist
 Data Analyst
 Statistician

Graph 70: A word cloud for cluster 2, of the “Term Searched” variable.

Statistician
 Data Analyst
 Data Scientist

Graph 71: A word cloud for cluster 3, of the “Term Searched” variable.

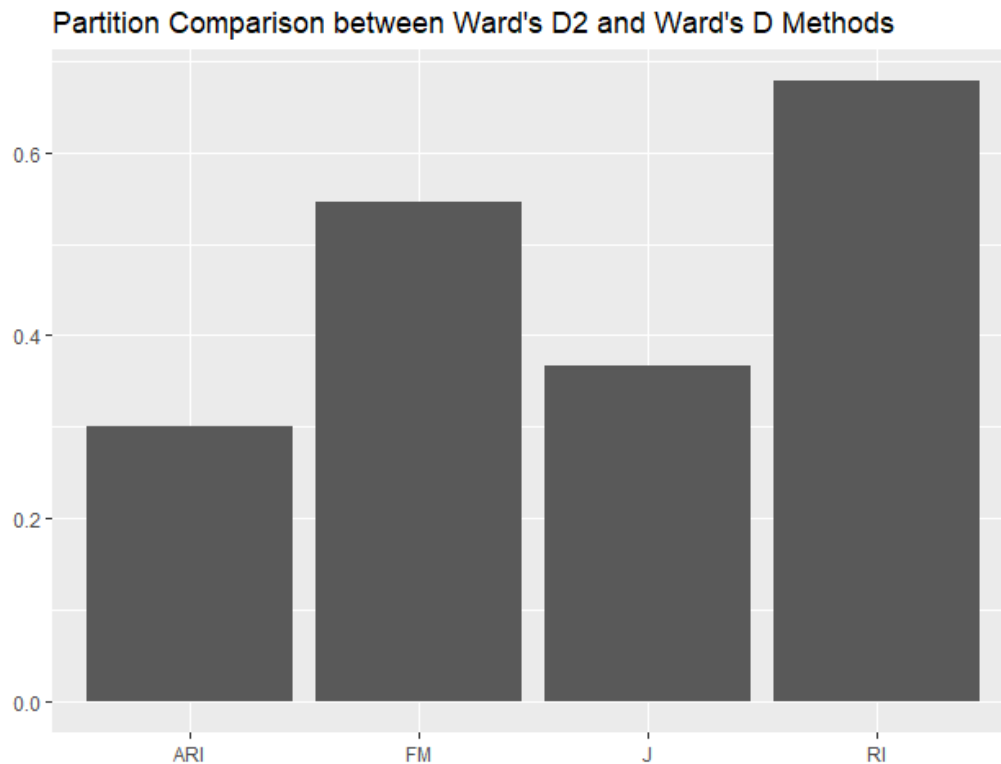
Statistician
 Data Scientist
 Data Analyst

Graph 72: A word cloud for cluster 4, of the “Term Searched” variable.

Looking at the *Graphs 69-72*, one can see that clusters 1 and 4 were mostly directed towards Data Scientists and that makes sense, considering that the programs they used can perform modelling, forecasting and everything else a Data Scientist needs. Cluster 2 was mostly oriented towards Data Analysts. With tools like SQL, PowerBI, Tableau and Qlik it seems natural, since these are mostly used to perform data manipulation and data analysis. The third cluster seemed to mix two roles in one, as it was addressing both Statisticians and Data Analysts. Knowing that lots of Statisticians used SAS, SAP and Stata, as it was apparent from the collection of our data, we can certainly say that they belong to this cluster and as for Data Analysts, they must certainly be using the Microsoft Office Suite to cover their needs.

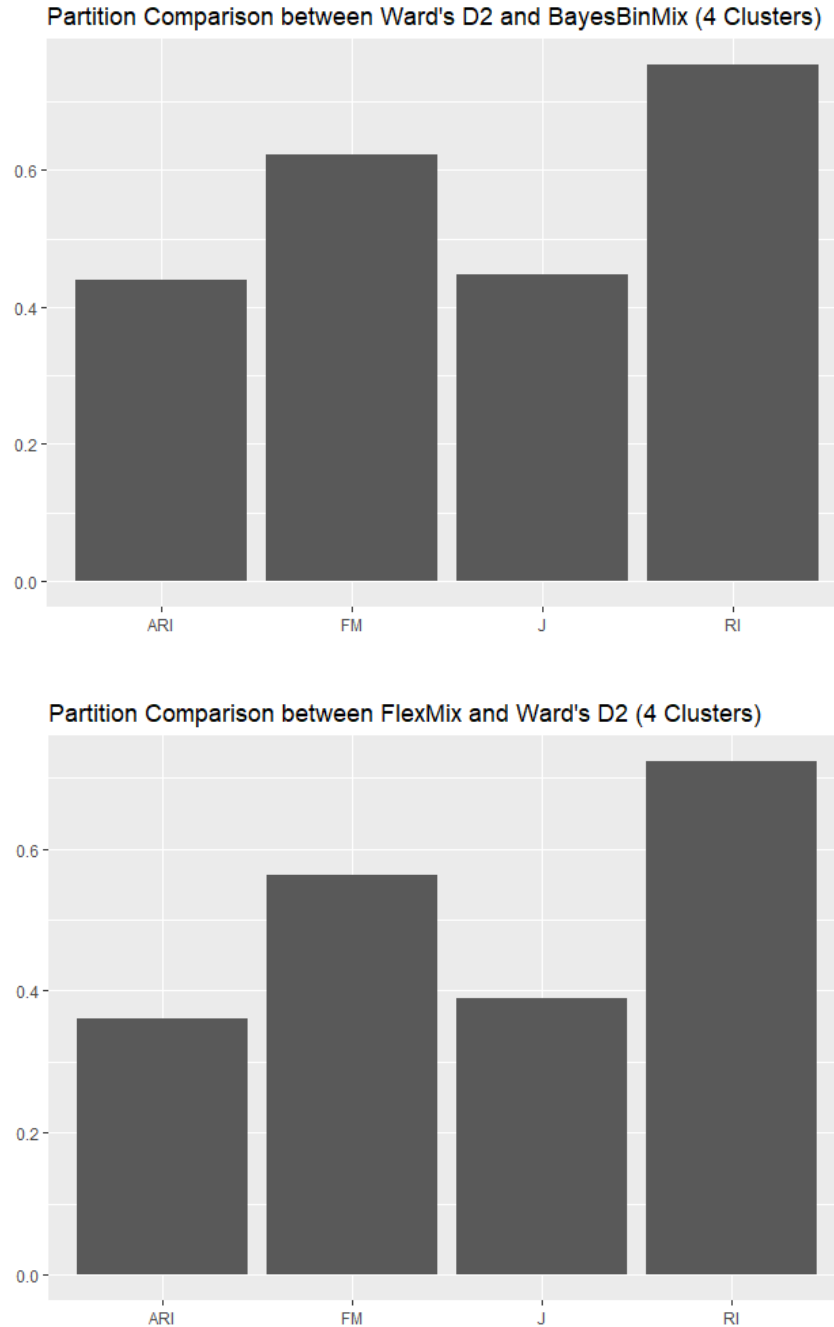
5.4 Comparing the Results of the Clustering Methods

Now after getting the clustering results for all the clustering methods and seeing that they ended up with the same number of clusters, even in the case of some of the hierarchical methods (Ward's D and Ward's D2), what we can do next is to take and compare those partitions in order to see how similar they are. If we first compare Ward's D and Ward's D2 partitions between themselves, then we will get the following:



Graph 73: A bar-plot of indices comparing the results of two partitions of clusters formed by Ward's D and Ward's D2.

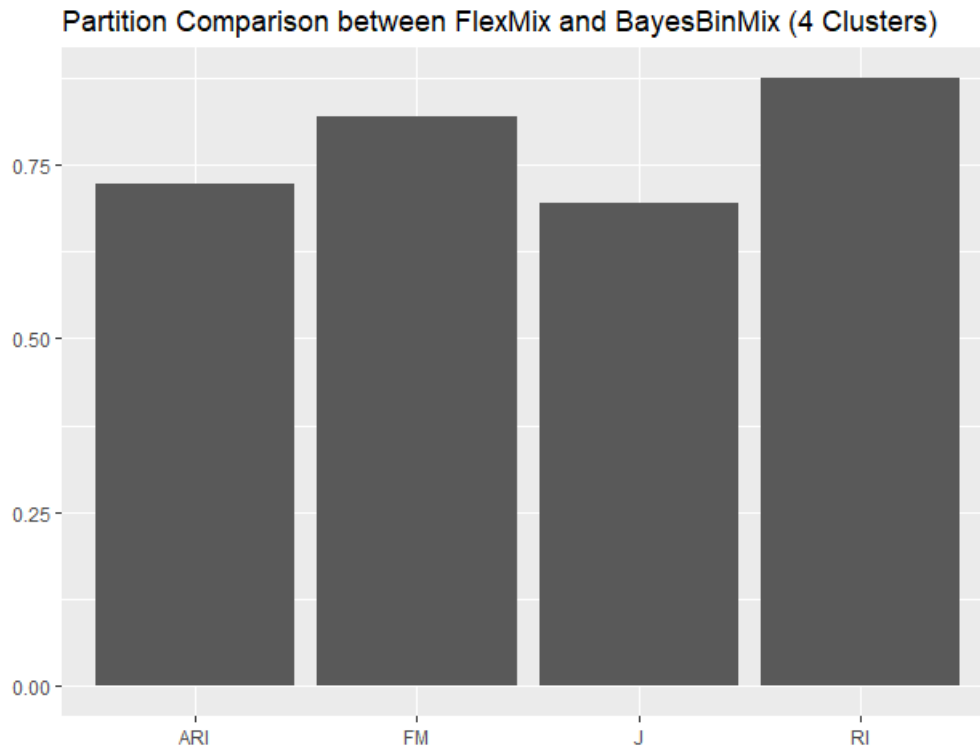
As it can be clearly seen from *Graph 73*, even though these two methods are close to one another, the results are not as similar as one would hope. Even looking back at *Graph 34*, which contained the dendrograms, one would see how weirdly Ward's D chose its cluster membership. Two of its clusters seem to have 3 elements in total and the other two split the rest, which is certainly not the outcome we would expect, based on the results of the other clustering techniques. Ward's D2 seems to be an okay clustering method, as it does split the data into seemingly logical clusters. And of course we can see how well it matches with the other two clustering methods right below.



Graphs 74: A bar-plot of indices comparing the results of two partitions of clusters formed by Ward's D2 and BayesBinMix, as well as Ward's D2 and FlexMix.

This is a little disheartening, although expected of hierarchical clustering. Looking at *Graphs 74*, one can see that it does have some coherence when compared to the results of other clustering techniques, although not as much as we would hope. This was however a logical result, as it is very difficult to form clusters in this data and thus maybe the hierarchical approach is a bit weaker.

when compared to other, model-based approaches. Let us now see how the last two clustering results compare to each other.



Graph 75: A bar-plot of indices comparing the results of two partitions of clusters formed by BayesBinMix and FlexMix.

As it can be seen from the bar-plot of *Graph 75*, while not identical, these two partitions are similar in their results. Certainly not all clusters contain the same observations, but the general groupings seem to agree with one another.

It was especially clear that in both cases there was a cluster that was oriented towards Healthcare and Pharmaceutical Companies and used similar statistical tools in both cases to get the job done. These tools that were in both clustering methods, were SAS, SAP and Stata. A big group of job searches in that cluster was the term of Statistician/Biostatistician, in both cases.

IT, Finance and Software companies were overall the biggest hiring firms for people who know these 34 monitored statistical tools in different combinations.

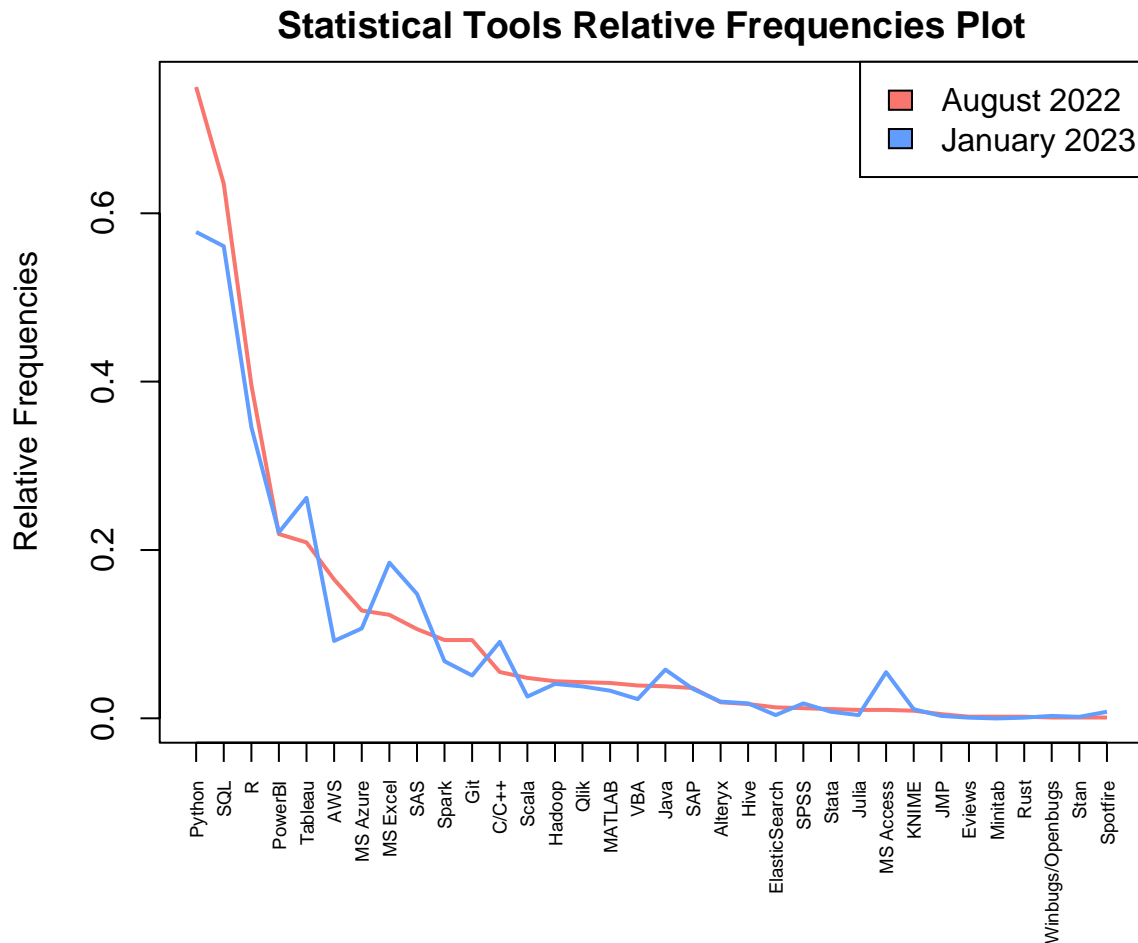
Data Analysts, appear to be gravitating towards SQL, Tableau, PowerBI, or tools that can do database management and do not require lots of programming knowledge. On the other hand, Data Scientists seem to gravitate towards statistical tools that do need to know how to program whatever it is that they are studying and depending on the company, these processes are either done by hand, through packages, or even through automated procedures.

Finally, it was also noticed that Senior level positions tended to avoid asking for programs that performed data management and data visualizations, as it was left to the more Junior, or better yet Associate level positions, which in both clustering techniques appeared to be something like all knowing positions, that required the knowledge of most of the 34 statistical programs that were set to be observed. Of course, it was not in all the Senior level positions, but the majority appeared to behave that way.

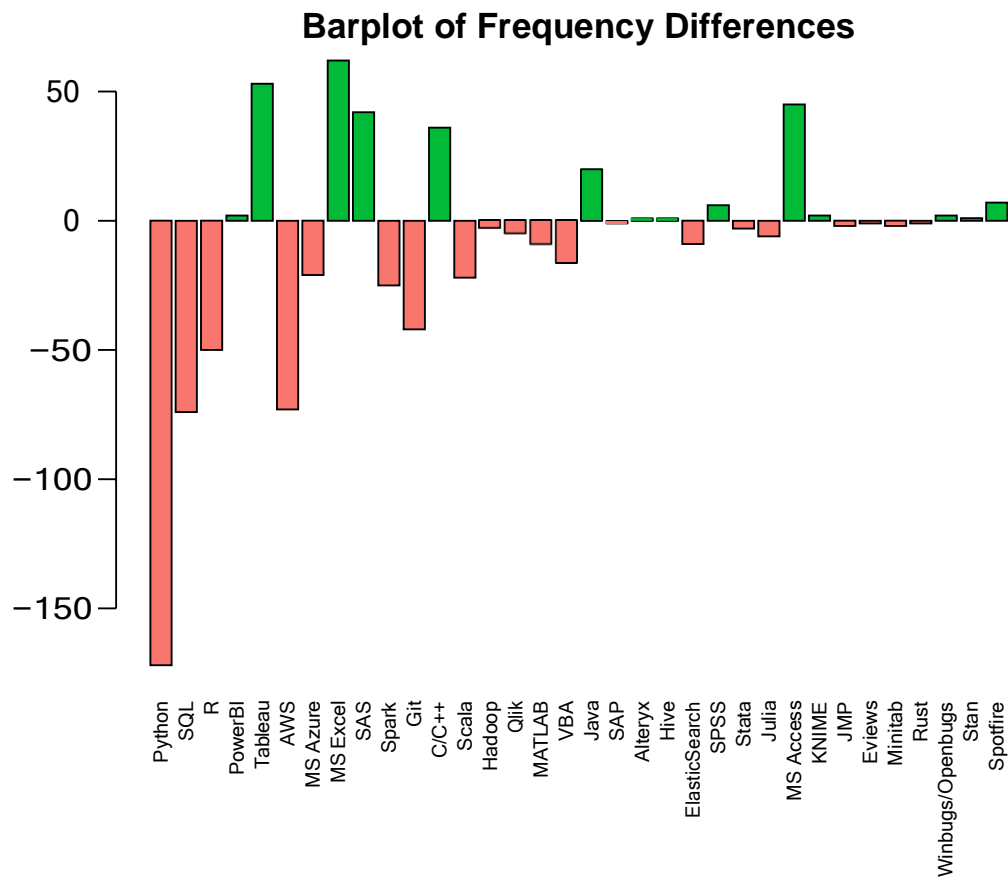


CHAPTER 6: TRENDS

Having developed an algorithm for relatively easy web scraping (discussed in 2.3 *Web Scraping*), it would certainly be interesting to see how the data have changed from the beginning of this thesis in August of 2022, until today, which is in January of 2023. Running our algorithm, we will get the following results:



Graph 76: A plot that shows relative frequencies for two periods of time. One when the data were first collected (August 2022) that is depicted by a red line and another coloured in blue, which is when the web scraping algorithm was run (January 2023). Both data sets contained 1000 observations each.



Graph 77: A plot that shows the differences between the two periods when data was collected. Therefore, it is a difference between the frequencies of January 2023 minus the corresponding frequencies of August 2022 for each of the statistical tools that were observed.

As it can be seen from the two graphs above (*Graph 76* and *Graph 77*), it does seem like some of the statistical tools have had some significant changes in their demand, but overall, they did not change too much. Even though, the popularity of Python had the biggest drop, which is apparent from *Graph 77*, it is still among the most popular pieces of statistical software that are available on the market. Perhaps the biggest boost in demand came from Microsoft Excel, which meant that more companies started asking that program from the potential job candidates. Tableau, as well as SAS enjoyed quite a rise in demand as well. These are both popular statistical tools and it was expected of Tableau, which is commonly mentioned along with tools such as PowerBI. As far as SAS is concerned, it probably means that many of the jobs that were monitored perhaps were those of the Healthcare sector, but as it was also mentioned, the web scraping algorithm does not discern between multiple job postings of the same position between countries and so it could be those of international firms (like IQVIA for example), which seek people for similar roles in different part of the world. Also, another significant rise in popularity came from MS Access, but this could also be attributed to the success of MS Excel, as when companies mention one product of the Microsoft Office Suite, they also tend to mention the other popular ones too. Thus, MS Excel, MS Access

and VBA are often seen together. Lastly, it should be mentioned that software like Winbugs/Openbugs and Spotfire, which were previously in the last places in terms of popularity, also saw some boost in their demand.



CHAPTER 7: CONCLUSION

In conclusion, we performed the ranking of software and programs used by statistician based on their frequencies and the results were obvious in *Graph 2*. Clearly the most sought-after programs were Python, SQL and R, in that order and these tools should definitely be known at least to some extent by practicing Statistician and Data Scientists as they are the ones that have the most demand among the companies. What is great is that they are free and open-source (at least the most used versions of SQL) and have lots of teaching material online, especially in the case of Python. What's more, is that Python, SQL and R still remain on top even after a five-month period.

Latent connections between the data were also explored, as well as a clustering was attempted to see if there were some logical groups that could be formed. After the analysis was done, it became clear that our observations could be divided into four clusters. What was also interesting to see, was that some groups of programs could be seen together for the majority of cases, like SQL, Tableau and PowerBI, or AWS and MS Azure. Now these programs were either complimentary to each other or substituted one another, but nonetheless, they could be seen together. What was really interesting, was to see how they were used in groups of jobs, like for instance, it was clear that Data Analyst jobs were referred to software that helped depict the data quickly, easily and without a lot of programming knowledge. While it appeared that Data Scientists really needed to know some programming skills in Python, R and other such tools that were needed for their trade. Statisticians or Biostatisticians actually had an inclination to using SAS, SAP or Stata as well as R (although not as often as SAS). We also saw that the majority of job positions for statisticians, data analysts, or data scientists were in IT, Financial, Software or Healthcare companies, with the last one having a clear preference for pure statisticians. It was also apparent that Senior level positions were mostly focused on modelling and interpreting the data, rather than managing and analyzing it, which are the first stages of any project. Those first stages were of course jobs for rookies and therefore, the lesser the number of years of experience needed, the more likely it was for companies to need more from a candidate. Therefore, all-in-all, it showed a logical view of our world with a few interesting insights.

It would certainly be interesting to repeat the study after some time has passed and see how and if things have changed as far as the usage of these tools is concerned, or even if they are still there, or have been replaced by something new and cutting edge and in that fashion, detect a trend for all the statistical tools that could be asked of a competent statistician within a job description. But for the moment, these are the most popular tools in 2022 and their results.



BIBLIOGRAPHY

- R. A. Muenchen. “*The Popularity of Data Analysis Software*”. 2013.
URL: <https://r4stats.com/articles/popularity/> [p.1]
- A. Agresti. “*An Introduction to Categorical Data Analysis*”. Wiley Series in Probability and Statistics, 2007.
URL: <https://doi.org/10.1002/0470114754> [p.3]
- H. Cramér. “*Mathematical Methods of Statistics*”. Princeton: Princeton University Press, 1946.
[p.4]
- A. M. Liebetrau. “*Measures of association*”. Sage Publications, 1983.
URL: <https://dx.doi.org/10.4135/9781412984942> [p.4]
- M. J. Newman. “*Networks: An Introduction*”. Oxford University Press, 2010.
URL: <https://doi.org/10.1093/acprof:oso/9780199206650.001.0001> [p.5]
- L. A. Douglas. “*A User's Guide to Network Analysis in R*”. Springer, 2015.
URL: <https://doi.org/10.1007/978-3-319-23883-8> [p.5]
- A. Clauset, M. E. J. Newman, and C. Moore. “*Finding Community Structure in Very Large Networks*”. Physical Review. E, Statistical, Nonlinear, and Soft Matter Physics, 24(6), 2005.
URL: <https://doi.org/10.1103/PhysRevE.70.066111> [p.6]
- M. Girvan and M. E. J. Newman. “*Community structure in social and biological networks*”. Proceedings of the National Academy of Sciences, 99(12): 7821-7826, 2002.
URL: <https://doi.org/10.1073/pnas.122653799> [p.6]
- M. E. J. Newman and M. Girvan. “*Finding and evaluating community structure in networks*”. Physical Review. E 69, 026113, 2004.
URL: <https://doi.org/10.1103/PhysRevE.69.026113> [p.6]
- P. N. Krivitsky and M. S. Handcock. “*Fitting Latent Cluster Models for Networks with latentnet*”. Journal of Statistical Software, 24(5): 1–23, 2008.
URL: <https://doi.org/10.18637/jss.v024.i05> [p.8]
- C. R. Mehta and N. R. Patel. “*IBM SPSS Exact Tests*”. IBM Corporation, 1989.
[p.9]
- A. Agresti, D. Wackerly, and J. M. Boyett. “*Exact conditional tests for cross-classifications: Approximations of attained significance levels*”. Psychometrika, 44(1): 75–83, 1979.
URL: <https://doi.org/10.1007/BF02293786> [p.9]
- A. C. A. Hope. “*A Simplified Monte Carlo Significance Test Procedure*”. Journal of the Royal Statistical Society. Series B (Methodological) 30(3): 582–98, 1968.
URL: <http://www.jstor.org/stable/2984263> [p.9]



W. M. Patefield. “*Algorithm AS 159: An Efficient Method of Generating Random $R \times C$ Tables with Given Row and Column Totals.*” *Journal of the Royal Statistical Society. Series C (Applied Statistics)* 30(1): 91–97, 1981.

URL: <https://doi.org/10.2307/2346669> [p.9]

Y. Benjamini and Y. Hochberg. “*Controlling the false discovery rate: a practical and powerful approach to multiple testing*”. *Journal of the Royal Statistical Society Series B*, (57): 289-300, 1995.

URL: <http://www.jstor.org/stable/2346101> [p.11]

H. Abdi, and L. J. Williams. “*Principal Component Analysis*”. John Wiley and Sons, Inc. *WIREs Computational Statistics* 2(4): 433–59, 2010.

URL: <https://doi.org/10.1002/wics.101> [p.11]

F. Husson, S. Le, J. Pagès. “*Exploratory Multivariate Analysis by Example Using R*”. Second Edition. Boca Raton, Florida: Chapman; Hall/CRC, 2017.

URL: <https://doi.org/10.1201/b21874> [p.11]

M. Greenacre and J. Blasius. “*Multiple Correspondence Analysis and Related Methods*”. Chapman & Hall CRC Statistics in the Social and Behavioral Sciences. Chapman and Hall/CRC, 2006.

[p.11]

G. James, T. Hastie, R. Tibshirani and D. Witten. “*An Introduction to Statistical Learning with Applications in R*”. Springer, 2015.

URL: <https://doi.org/10.1007/978-1-4614-7138-7> [p.15]

T. Hastie, R. Tibshirani, J. Friedman. “*The Elements of Statistical Learning - Data Mining, Inference, and Prediction*”. Springer, 2013.

URL: <https://doi.org/10.1007/978-0-387-84858-7> [p.15]

B. S. Everitt, S. Landau, M. Leese and D. Stahl. “*Cluster analysis*”. Wiley, 2011.

URL: <https://doi.org/10.1002/9780470977811> [p.16] [p.23]

F. Leisch. “*FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R*”. *Journal of Statistical Software*, 11(8), 1–18, 2004.

URL: <https://doi.org/10.18637/jss.v011.i08> [p.17]

P. Papastamoulis and M. Rattray. “*BayesBinMix: an R Package for Model Based Clustering of Multivariate Binary Data*”. *The R Journal*. 9(1): 403-420, 2017.

URL: <https://doi.org/10.32614/RJ-2017-022> [p.19]

J. M. Santos and M. J. Embrechts. “*On the Use of the Adjusted Rand Index as a Metric for Evaluating Supervised Classification*”. ICANN, 2009.

URL: https://doi.org/10.1007/978-3-642-04277-5_18 [p.22] [p.23]



M. Meila. “*Comparing clusterings—an information based distance*”. Journal of Multivariate Analysis, 98(5): 873-895, 2007.

URL: <https://doi.org/10.1016/j.jmva.2006.11.013> [p.22] [p.23]

L. Hubert and P. Arabie. “*Comparing partitions*”. Journal of Classification (2): 193–218, 1985.

URL: <https://doi.org/10.1007/BF01908075> [p.22]

E. Dimitriadou, S. Dolnicar and A. Weingessel. “*An Examination of Indexes for Determining the Number of Clusters in Binary Data Sets*”. Psychometrika 67: 137–159, 2002.

URL: <https://doi.org/10.1007/BF02294713> [p.24]

D. Ratkowsky, G. Lance. “*A criterion for determining the number of groups in a classification*”. Australian Computer Journal, 10(3):115-117, 1978.

URL: <http://hdl.handle.net/102.100.100/299282?index=1> [p.24]

H. P. Friedman and J. Rubin. “*On some invariant criteria for grouping data*”. Journal of the American Statistical Association, 62(320): 1159–1178, 1967.

URL: <https://doi.org/10.1080/01621459.1967.10500923> [p.24]

D. L. Davies and D. W. Bouldin. “*A Cluster Separation Measure*”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1(2): 224-227, 1979.

URL: <https://doi.org/10.1109/TPAMI.1979.4766909> [p.24]

C. Biernacki, G. Celeux, and G. Govaert. “*Assessing a Mixture Model for Clustering with the Integrated Completed Likelihood*”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22(7):719–725, 2000.

URL: <https://doi.org/10.1109/34.865189> [p.73] [p.88]

P. Papastamoulis, M.-L. Martin-Magniette, and C. Maugis-Rabusseau. “*On the estimation of mixtures of Poisson regression models with large number of components*”. Computational Statistics & Data Analysis, 93:97–106, 2016.

URL: <https://doi.org/10.1007/s11009-011-9238-7> [p.73] [p.88]

