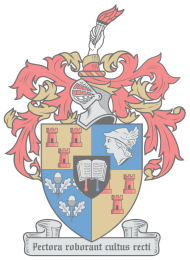
CLUSTERING METHODS WITH A FOCUS ON SELF-ORGANISING MAPS AND AN IMPLEMENTATION ON RETAIL BANK TRANSACTIONAL DATA

Chrismarie Enslin



UNIVERSITEIT
iYUNIVESITHI
STELLENBOSCH
UNIVERSITY

Thesis presented in partial fulfilment
of the requirements for the degree of
Master of Commerce in the Faculty of
Economic and Management Sciences
at Stellenbosch University

Supervisor: Professor S.J. Steel	December 2018

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: December 2018

Abstract

The aims of this study is to provide an overview of traditional clustering methods, as well as introduce and discuss self-organising maps (SOMs) in detail. This study wants to convince the reader of the usefulness of self-organising maps as a dimension reduction tool. The batch SOMs algorithm was found to be the most appropriate SOM to use in practice, together with random initialisation of the prototypes. Ward linkage hierarchical clustering was found to perform the best on multivariate Gaussian simulated data and it was also found to be the most appropriate traditional clustering method to fit on top of the SOM. Banking transactional data was investigated for client behavioural clusters and the clusters of lower socio-economic class clients, technologically sophisticated clients, older and more traditional clients and low financial activity clients were found. These clusters emerged consistently throughout 9 different data samples.

Key Words:

Unsupervised learning, Self-organising maps, Clustering, K-means clustering, K-medoids clustering, hierarchical clustering, CLARA

Opsomming

Die doel van hierdie studie is om 'n oorsig oor tradisionele groeperings metodes saam te stel, sowel as om selforganiserende kaarte (SOK) ("self-organising maps") te bespreek. Hierdie studie wil die leser oortuig van die bruikbaarheid van SOK as 'n dimensie-vermindering tegniek. Die bondel-SOK algoritme is die metode wat in die praktyk aanbeveel word, saam met lukrake inisialisering van die prototipes. Ward-koppeling ("Ward linkage") hiërargiese groepering het die beste presteer op multivariaat-Gaussies gesimuleerde data. In hierdies studie is ook gevind dat Ward-koppeling die mees toepaslike tradisionele groeperingsmetode was om bo-op die SOK aan te wend. Data uit die transaksionele bank omgewing is ondersoek om kliënt gedragsgroepe te vind. Hierdie gedragsgroepe is geïdentifiseer as laer sosio-ekonomiese klas kliënte, tegnologies gesofistikeerde kliënte, ouer en meer tradisionele kliënte en ook 'n groep met lae finansiële aktiwiteit. Die ontleding het hierdie groepe konsekwent oor 9 verskillende datastelle geïdentifiseer.

Sleutelwoorde:

Leer sonder toesig ("unsupervised learning"), selforganiserende kaarte, groepering, K-gemiddelde groepering, K-"medoid" groepering, hiërargiese groepering, CLARA

Acknowledgements

My sincerest thank you to Prof S.J. Steel for supporting, guiding and motivating me throughout the writing of this study. I appreciate the feedback that was required at a very late stage and the thorough comments that improved the quality of the final product.

Thank you to August, from Bank C, for the discussions around the investigation of the data and setting up of desired outcomes.

A very sincere thank you to Bank C, for allowing me the usage of their transactional data in this study. The data problem was incredibly interesting and I hope the feedback from this study can feed back into bettering the lives of the Bank C clients.

I would also like to thank Marius from Bank C, for brainstorming with me on some of my variable cleaning problems and teaching me some SQL shortcuts.

A big thank you to the examiners for their input that contributed to an improved final product. It is greatly appreciated.

Contents

Declaration i

Abstract ii

Opsomming iii

Acknowledgements iv

List of Tables vi

List of Figures vii

CHAPTER 1 INTRODUCTION 1

1 Introduction 1
- 1.1 Research Objectives and Design . . . 1
- 1.2 Chapter Outline . . . 2

CHAPTER 2 LITERATURE REVIEW (PART 1) 3

2 Traditional Clustering Methods 3
- 2.1 Introduction . . . 3
  - 2.1.1 Proximity and Dissimilarity . . . 3
  - 2.1.2 Combinatorial Algorithms . . . 5
- 2.2 $\boldsymbol{K}$-means Clustering . . . 6
- 2.3 $\boldsymbol{K}$-medoids clustering . . . 9
- 2.4 Hierarchical Clustering . . . 12
  - 2.4.1 Agglomerative Hierarchical Clustering . . . 13
  - 2.4.2 Divisive Hierarchical Clustering . . . 17
- 2.5 Practical Issues . . . 18
  - 2.5.1 Missing Values . . . 18
- 2.6 How to choose $\boldsymbol{K}$ . . . 20
  - 2.6.1 Standardisation of Variables . . . 23
  - 2.6.2 Weighting of Variables . . . 24
- 2.7 Cluster Validity . . . 26
  - 2.7.1 External Criteria . . . 26
  - 2.7.2 Relative Criteria . . . 27
- 2.8 Summary . . . 28

CHAPTER 3 LITERATURE REVIEW (PART 2) 29

3 Clustering Large and High-Dimensional Datasets 29
- 3.1 Introduction . . . 29
- 3.2 Clustering Large Applications (CLARA) . . . 29
- 3.3 Self-Organising Maps (SOMs) . . . 30

3.3.1 Competitive Learning . . . 30
3.3.2 Details about SOMs . . . 32
3.3.3 Initialisation of the SOM . . . 35
3.3.4 The Batch SOM . . . 37
3.3.5 Combining SOMs and other clustering methods . . . 39
3.3.6 Growing Self-organizing Maps (GSOMs) and Growing Hierarchical Self-organizing Maps (GHSOMs) . . . 40
3.4 Summary . . . 43

CHAPTER 4 SIMULATION STUDY **44**

4 Data Simulation Study **44**
4.1 Introduction . . . 44
4.2 Design of the Datasets . . . 44
4.2.1 Datasets for Scenarios A and B . . . 46
4.2.2 Visualising **10**-dimensional data for Scenarios A and B . . . 47
4.3 Cluster Validation . . . 49
4.4 Scenario A . . . 50
4.4.1 $\boldsymbol{K}$-means Clustering . . . 50
4.4.2 $\boldsymbol{K}$-medoids Clustering . . . 55
4.4.3 Clustering Large Applications (CLARA) . . . 57
4.4.4 Agglomerative Hierarchical Clustering . . . 58
4.4.5 Divisive Hierarchical Clustering . . . 61
4.4.6 Self-Organising Maps (SOMs) . . . 64
4.4.7 Discussing the validity of the clusters for Scenario A . . . 70
4.5 Scenario B . . . 71
4.6 Summary . . . 76

CHAPTER 5 DATA EXTRACTION AND CLEANING **77**

5 Data Extraction and Cleaning **77**
5.1 Introduction . . . 77
5.2 Bank C Transactional Data . . . 77
5.3 Data Sampling . . . 78
5.4 Variable Set Description . . . 79
5.5 Variable Cleaning and Transformations . . . 80
5.6 Outlier Detection . . . 82
5.7 Dimension Reduction . . . 84
5.8 Data Normalisation and Final Dataset . . . 85
5.9 Summary . . . 92

CHAPTER 6 DATA MODELLING **94**

6 Data Modelling **94**
6.1 Introduction . . . 94
6.2 Data Summary and Visualisation . . . 94

6.3 SOM fitting 96
6.4 Deciding on the Appropriate Values for K 98
6.5 Fitting Ward linkage clustering on the SOMs 101
6.6 Client Behaviour Profiles 101
6.7 Summary 114

CHAPTER 7 CONCLUSION **115**

7 Summary, Conclusion and Recommendations **115**
7.1 Introduction 115
7.2 Summary of Main Findings 115
7.3 Recommendations 117

References **118**

Appendix A **121**

Appendix B **122**

Appendix C **123**

Appendix D **125**

Appendix E **131**

Appendix F **140**

Appendix G **143**

Appendix H **147**

Appendix I **152**

List of Tables

1 Values of $\boldsymbol{C}_{jih}$ 12
2 Values of $\boldsymbol{C}_{jih}$ 12
3 Linkage for Agglomerative Hierarchical Clustering 15
4 Scenario A and B overview 46
5 Scenario A Comparison of Clustering Algorithms 70
6 Scenario B Comparison SOM Algorithms and Initialisations 72
7 Scenario B Comparison of Clustering Algorithms 72
8 Account Balance History Variables (Last 3 months) 79
9 Outflows and Inflows History Variables (Last 3 months) 79
10 Definitions of Variable Names 80
11 Top 50 Variables (Last 3 months) 81
12 Summary of the 9 Datasets 94
13 Summary of PCs and SOM dimensions 96
14 Summary of Possible Values of K 99

List of Figures

1 Standard normal data, $p = 2$ and $K = 3$ . . . 7
2 Figures showing the iterations of the K-means clustering algorithm . . . 8
3 Different forms of linkage for $K = 3$ Toy Data . . . 14
4 Figures showing the evolution of K-means clustering for increasing values of K 19
5 The elbow method for choosing K' . . . 21
6 The gap statistic for choosing K' . . . 22
7 The average silhouette Method for choosing K' . . . 23
8 Illustration of scaled variables negatively affecting the outcome of K-Means clustering . . . 25
9 Multivariate Gaussian data, $p = 4$ and $K = 3$ . . . 33
10 Examples of the two types of SOM topologies, (a) Rectangular and (b) Hexagonal, with dimensions according to the ratio of the first two eigenvalues and the pie charts representing the weights of the prototype vectors . . . 36
11 Data Scenarios A and B plotted in terms of their first two PCs with given class labels . . . 48
12 The gap statistic (a) and (b), the elbow method (c) and the silhouette method (d) for choosing K for Scenario A using K-means clustering . . . 52
13 Scenario A plotted in terms of the first two PCs with class labels from K-means clustering (a) and the silhouette plot (b) . . . 54
14 Scenario A plotted in terms of the first two PCs with class labels from K-medoids clustering (a) and the silhouette plot (b) . . . 56
15 Scenario A plotted in terms of the first two PCs with class labels from the CLARA algorithm (a) and the silhouette plot for Scenario A (b) . . . 58
16 Dendrograms for different forms of linkage for Scenario A . . . 60
17 Agglomerative hierarchical clustering data plots and silhouette plots with (a)-(b) complete, (c)-(d) average, (e)-(f) McQuitty and (g)-(h) Ward linkage in Scenario A . . . 62
18 Divisive hierarchical clustering (a) dendrogram, (b) cluster plot and (c) silhouette plot for Scenario A . . . 63
19 SOM U-matrices for (a) the batch SOM with random initialisation, (b) the online SOM with random initilisation, (c) the batch SOM with linear initialisation and (d) the online SOM with linear initialisation for Scenario A . . . 67
20 Ward linkage clustering on top of the batch SOM with linear initialisation for Scenario A, (a) prototype grid with pie charts representing prototype weights, (b) dendrogram of Ward linkage clustering on top of the batch SOM, (c) colour-coded cluster outcome and (d) silhouette plot . . . 68
21 The U-matrix for a batch SOM with random initialisation on Scenario B . . . 73
22 Dendrogram of Ward linkage on a batch SOM in Scenario B . . . 74
23 Ward linkage on top of the batch SOM (a) cluster plot and (b) silhouette plot for Scenario B . . . 75
24 CLARA clustering for (a) cluster plot and (b) silhouette plot for Scenario B 75

25 Boxplots of (a) Val_Ct_Tran_L3M, (b) Val_Dt_Tran_L3M, (c) Num_Ct_Tran_L3M and (d) Num_Dt_Tran_L3M 83
26 Correlation plot of 52 variables to be excluded 86
27 Correlation plot of 41 variables to keep 87
28 Histograms for the monetary values, transaction counts and monetary averages variables 88
29 Histograms for the variables containing information on monthly frequency of the different channel usage 89
30 Histograms for the percentage and ratio variables 90
31 Histograms for the variables containing information on sizes of inflows and outflows 91
32 Histograms for the indicator variables 92
33 Normalised data plot on first two PCs for all 9 datasets 95
34 U-Matrix of the SOM of 3M_SAMP1 97
35 Gap statistic, elbow method and average silhouette method plot for the 3M_SAMP1 dataset 98
36 Dendrograms for (a) 3M_SAMP1, (b) 3M_SAMP2, (c) 3M_SAMP3, (d) 6M_SAMP1, (e) 6M_SAMP2, (f) 6M_SAMP3, (g) 12M_SAMP1, (h) 12M_SAMP2 and (i) 12M_SAMP3 100
37 Colour-coded cluster plots on the first two PCs for (a) 3M_SAMP1, (c) 3M_SAMP2, (e) 3M_SAMP3 and Silhoutte plots for (b) 3M_SAMP1, (d) 3M_SAMP2, (f) 3M_SAMP3 102
38 Colour-coded cluster plots on the first two PCs for (a) 6M_SAMP1, (c) 6M_SAMP2, (e) 6M_SAMP3 and Silhoutte plots for (b) 6M_SAMP1, (d) 6M_SAMP2, (f) 6M_SAMP3 103
39 Colour-coded cluster plots on the first two PCs for (a) 12M_SAMP1, (c) 12M_SAMP2, (e) 12M_SAMP3 and Silhoutte plots for (b) 12M_SAMP1, (d) 12M_SAMP2, (f) 12M_SAMP3 104
40 Histograms for the monetary values, transaction counts and monetary averages variables, colour-coded according to clusters for 3M_SAMP1 105
41 Density plots for the monetary values, transaction counts and monetary averages variables, colour-coded according to clusters for 3M_SAMP1 105
42 Histograms of the variables containing information on monthly frequency of the different channel's usage, colour-coded according to clusters for 3M_SAMP1 106
43 Density plots of the variables containing information on monthly frequency of the different channel's usage, colour-coded according to clusters for 3M_SAMP1 106
44 Histograms of percentage and ratio variables, colour-coded according to clusters for 3M_SAMP1 107
45 Density plots of percentage and ratio variables, colour-coded according to clusters for 3M_SAMP1 108
46 Histograms of the variables containing information on sizes of inflows and outflows, colour-coded according to clusters for 3M_SAMP1 109
47 Density plots of the variables containing information on sizes of inflows and outflows, colour-coded according to clusters for 3M_SAMP1 110

48 Histograms of indicator variables, colour-coded according to clusters for 3M_SAMP1 . . . 111
49 Density plots of indicator variables, colour-coded according to clusters for 3M_SAMP1 . . . 111
50 Variable 1 to 15 . . . 132
51 Variable 19 to 36 . . . 133
52 Variable 37 to 54 . . . 134
53 Variable 55 to 69 . . . 135
54 Variable 70 to 84 . . . 136
55 Variable 85 to 99 . . . 137
56 Variable 100 to 114 . . . 138
57 Variable 115 to 127 . . . 139
58 U-matrices of (a) the SOM of 3M_SAMP2 and (b) the SOM of 3M_SAMP3 144
59 U-Matrix of (a) the SOM of 6M_SAMP1, (b) the SOM of 6M_SAMP2 and (c) the SOM of 6M_SAMP3 . . . 145
60 U-Matrix of (a) the SOM of 12M_SAMP1, (b) the SOM of 12M_SAMP2 and (c) the SOM of 12M_SAMP3 . . . 146
61 Gap Statistic, Elbow method and Average Silhouette method plot for the 3M_SAMP2 dataset . . . 148
62 Gap Statistic, Elbow method and Average Silhouette method plot for the 3M_SAMP3 dataset . . . 148
63 Gap Statistic, Elbow method and Average Silhouette method plot for the 6M_SAMP1 dataset . . . 149
64 Gap Statistic, Elbow method and Average Silhouette method plot for the 6M_SAMP2 dataset . . . 149
65 Gap Statistic, Elbow method and Average Silhouette method plot for the 6M_SAMP3 dataset . . . 150
66 Gap Statistic, Elbow method and Average Silhouette method plot for the 12M_SAMP1 dataset . . . 150
67 Gap Statistic, Elbow method and Average Silhouette method plot for the 12M_SAMP2 dataset . . . 151
68 Gap Statistic, Elbow method and Average Silhouette method plot for the 12M_SAMP3 dataset . . . 151
69 Histograms for the monetary values, transaction counts and monetary averages variables, colour-coded according to clusters for 3M_SAMP2 . . . 153
70 Density plots for the monetary values, transaction counts and monetary averages variables, colour-coded according to clusters for 3M_SAMP2 . . . 154
71 Histograms of the variables containing information on monthly frequency of the different channel's usage, colour-coded according to clusters for 3M_SAMP2 154
72 Density plots of the variables containing information on monthly frequency of the different channel's usage, colour-coded according to clusters for 3M_SAMP2 . . . 155
73 Histograms of percentage and ratio variables, colour-coded according to clusters for 3M_SAMP2 . . . 156

74 Density plots of percentage and ratio variables, colour-coded according to clusters for 3M_SAMP2 157
75 Histograms of the variables containing information on sizes of inflows and outflows, colour-coded according to clusters for 3M_SAMP2 158
76 Density plots of the variables containing information on sizes of inflows and outflows, colour-coded according to clusters for 3M_SAMP2 158
77 Histograms of indicator variables, colour-coded according to clusters for 3M_SAMP2 159
78 Density plots of indicator variables, colour-coded according to clusters for 3M_SAMP2 159

CHAPTER 1: INTRODUCTION

1 Introduction

Self-Organising Maps (SOMs) were introduced in 1982 by Teuvo Kohonen and the original paper has been cited 9 800 times since then. The book called *Self-Organizing Maps*, written by Teuvo Kohonen himself, released its third edition in 2001. Since then it has been cited 23 487 times. These numbers are a testament to the popularity of SOMs and the value it adds to the field of unsupervised learning.

There are a large number of traditional clustering methods available that are suited to small datasets with low dimensions. However, the nature of the data that is available in the world today is more likely to be extremely large and high-dimensional. These traditional methods break down in high-dimensions and can become time-consuming and unreliable.

SOMs is a method that maps high-dimensional data down to lower dimensions and is particularly useful in the current data climate. After dimensionality of the data has been reduced the possibility to apply traditional clustering methods becomes available again. This means that SOMs can be used to preserve the traditional clustering methods, by making the data in the modern world accessible to them. For this reason it is important to understand the traditional clustering methods, along with the SOMs. Even though modern alternatives for dimension reduction also exist, for example t-Distributed Stochastic Neighbor Embedding (t-SNE), the focus in this study will be on SOMs.

It is important to apply the methods to simulated and real-world data to understand how they work. For this study we have access to Bank C's transactional history data for their savings clients. There is a need to separate these clients into different financial behavioural groups, each group with specific needs and requirements from their primary bank. These different groups of clients can be presented with different banking products, can be supplied with relevant financial education material and can be addressed in the communication channels they prefer. It is very important for Bank C that their clients have a good experience and relationship with the bank, and the clusters formed from a combination of SOMs and a traditional clustering method can aid in this endeavour.

1.1 Research Objectives and Design

There are several aims in this study. Firstly, this study wants to convince the reader of the usefulness of SOMs as a dimension reduction tool. Secondly, this study wants to find the optimal traditional clustering method to fit onto the prototype vectors of a SOM. Thirdly, we want to find client behavioural clusters in Bank C's transactional data. These groups of clients would have different requirements for a good banking experience and have to be communicated with in different ways.

The testing of the different methods in this project will be done through a simulation study

in the R programming environment and the data extraction will be done using the SQL Server query language.

1.2 Chapter Outline

Chapter 2 provides a literature review on traditional clustering methods, originally designed for small and low-dimensional datasets. We will also investigate different methods for choosing the appropriate number of clusters and how to evaluate the quality of the clustering outcome.

Chapter 3 provides a literature review on clustering methods that are more appropriate for large and high-dimensional datasets. Specifically we will investigate SOMs, which is the main focus of the study.

In Chapter 4 a simulation study on the clustering methods discussed in the literature review will be provided. The relevant R functions, packages and code will also be discussed and will form part of the text. In the simulation study we will decide on the clustering methods to apply to the real-world data.

In Chapter 5 we will discuss the data extraction, cleaning and normalisation process followed on the real-world data. The data consists of transactional histories of the savings accounts of Bank C's clients.

Chapter 6 shows the implementation of the SOM combined with a traditional clustering method on the cleaned transactional data. The clusters that resulted from this implementation will be discussed in detail.

Chapter 7 provides a summary of the results found in the simulation study, as well as recommendations gathered from the literature. A conclusion on the results of the clustering of the transactional data will be given. It serves as a summary of the entire findings of this report and also recommendations for future investigations and research.

CHAPTER 2: LITERATURE REVIEW (PART 1)

2 Traditional Clustering Methods

2.1 Introduction

Cluster analysis is the statistical process of identifying groups or clusters of observations in a dataset so that the observations in any given group or cluster are more similar to each other than to the observations in the other groups. It is an unsupervised method because no outcome variable is used to train the model or test its accuracy. The homogeneous groups found in the data are called clusters and are mostly non-overlapping. Although there are methods, such as Fuzzy C-Means clustering, that allow overlapping clusters, these will not be discussed here. In high-dimensional data, cluster analysis can assist with the understanding and visualisation of relationships between the variables and between the observations.

One application of cluster analysis is that different treatments can be applied to the different clusters that have been identified in a dataset. The underlying assumption is that the individuals in a cluster will respond similarly. For example, in the banking environment, a group of clients can be assigned a financial sophistication label. This label can be used to reflect anticipated good or bad financial behaviour. If the anticipated behaviour is of good quality, certain rewards can be put in place. On the other hand, if the anticipated behaviour might jeopardise the financial health of a client, financial education can be made available to the client.

2.1.1 Proximity and Dissimilarity

The term proximity refers to how close one observation is to another and can be defined in terms of similarity or dissimilarity. In cluster analysis dissimilarity is used as the definition for proximity. The input into any cluster analysis algorithm is a proximity (or, equivalently, dissimilarity) matrix D, rather than the raw observation matrix X. The matrix X is an $N \times p$ matrix, where p is the number of variables and N is the number of observations. In such a case the dissimilarity matrix will be of size $N \times N$. Entries in this matrix will be $d_{ii'}$, which is the dissimilarity between the observation vector $\boldsymbol{x}_i$ and $\boldsymbol{x}_{i'}$. The diagonal of D will contain zeroes. If proximity had been defined in terms of similarity, the diagonal of D would contain ones. The matrix D also has to be symmetric; if this requirement is not met, D can be replaced by $\frac{D+D^T}{2}$ (Hastie, Tibshirani & Friedman, 2009:503). Some clustering functions in R, for example *kmeans()* from the package **stats**, can create the matrix D as part of the clustering process. These functions accept the raw observation matrix X as an input.

Whenever $p > 1$ the dissimilarity between two observation vectors is defined in terms of dissimilarities for the separate variables. We therefore first consider dissimilarity between two observations with respect to a given variable before discussing how to measure dissimilarity between objects. Pairwise dissimilarity uses the notation $d_j(x_{ij}, x_{i'j})$ as we briefly defined

before. This denotes the dissimilarity between observation i and observation i' measured in terms of variable j. A choice has to be made regarding the form of the measure d_j and this will be determined by the types of variables occurring in the dataset.

For quantitative variables the dissimilarity can be specified to look like a loss function $\ell(.)$, *i.e.* $d_j(x_{ij}, x_{i'j}) = \ell(|x_{ij} - x_{i'j}|)$. This "loss function" will assign larger values to observation pairs that lie further apart, *i.e.* are more dissimilar. Most often this function is chosen to be a squared error, $d_j(x_{ij}, x_{i'j}) = (x_{ij} - x_{i'j})^2$, or an absolute error, $d_j(x_{ij}, x_{i'j}) = |x_{ij} - x_{i'j}|$. These are well known loss functions that are easily understood and frequently used in other statistical learning methods. For categorical variables the $0-1$ loss function is often chosen, $d_j(x_{ij}, x_{i'j}) = I(x_{ij} \neq x_{i'j})$, where $I(.)$ denotes the indicator function (Hastie *et al.*, 2009:503-504).

We now define a measure of dissimilarity between objects based on a set of p variables. We define $D(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \sum_{j=1}^{p} d_j(x_{ij}, x_{i'j})$ as the dissimilarity between two rows of X. There are certain requirements that $D(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ has to satisfy to be considered a distance metric:

1. Symmetry: $D(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = D(\boldsymbol{x}_{i'}, \boldsymbol{x}_i)$
2. Positivity: $D(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \geq 0$, $\forall \boldsymbol{x}_i, \boldsymbol{x}_{i'}$
3. Triangle Inequality: $D(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \leq D(\boldsymbol{x}_i, \boldsymbol{x}_k) + D(\boldsymbol{x}_k, \boldsymbol{x}_{i'})$, $\forall \boldsymbol{x}_i, \boldsymbol{x}_{i'}, \boldsymbol{x}_k$
4. Reflexivity: $D(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = 0$ if and only if $\boldsymbol{x}_i = \boldsymbol{x}_{i'}$.

Frequently weights are attached to some of the d_j-values, *i.e.* to the dissimilarities of certain variables (Hastie *et al.*, 2009:505; Xu & Wunsch II, 2009:21). This will be discussed later. Distance measures between clusters will also be discussed later.

The measure of dissimilarity chosen might have a larger impact on the resulting clusters than the clustering method itself. Therefore, care has to be taken when this measure is specified. For this research project we will focus on Euclidean distance as dissimilarity measure, defined by

$$D(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \sqrt{\sum_{j=1}^{p} (x_{ij} - x_{i'j})^2}.$$

Euclidean distance is a popular distance measure as it is easy to understand and finds a straight line between two observations. In some cases squared Euclidean distance, $D(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \|\boldsymbol{x}_i - \boldsymbol{x}_{i'}\|^2$, will be used. Euclidean distance can only accommodate quantitative variables. This choice can be made because of *a priori* knowledge regarding the type of variables available for the real-world dataset that will be investigated. In this study the real-world dataset will only contain binary, ordinal and continuous quantitative input variables.

When categorical variables or variables containing strings are present in the input dataset, other distance measures need to be considered. For strings, it is quite popular to use an edit distance measure. This refers to the number of edits required for one string to look exactly like the other. The simplest distance measure for non-ordinal categorical variables would be to provide a distance of 0 if two categories are the same and a value of 1 if they are different.

This is called measuring the *overlap* and more distance measures based on this have been developed. These distance measures will not be discussed further, but deserve consideration if categorical or text variables are present in the data.

2.1.2 Combinatorial Algorithms

Clustering can be done in several ways, for example using combinatorial algorithms, mixture models or bump hunting. In this discussion we will focus on combinatorial algorithms using the observed data without any assumptions about underlying distributions. More specifically, K-means clustering, K-medoids clustering, hierarchical clustering, Density-based clustering and Self-Organising Maps (SOMs) will be explored.

The number of clusters will be denoted by $K < N$. For certain algorithms the value of K has to be specified beforehand, for example K-means clustering. Other approaches, for example hierarchical clustering, requires the value of K to be specified after the algorithm has been completed. In every case we require an encoder, denoted by $C(.)$, that maps the observations to the clusters, $k = C(i), k \in \{1, 2, ..., K\}, i \in \{1, 2, ..., N\}$. It is clear that $C(i)$ therefore denotes the cluster index for observation i (Hastie *et al.*, 2009:509).

When a cluster analysis is performed we would like the subgroups identified in the data to have two important properties. Firstly, we would like the observations inside any subgroup to be homogeneous and to display a low within-cluster dissimilarity. Secondly, we would also like different subgroups to be as heterogeneous as possible and thus have a high between-cluster dissimilarity. These two requirements can have different levels of desirability in different applications.

A cluster algorithm requires specification of a measure of within-cluster variation as well as a measure of between-cluster variation. The between- and within-cluster variation can assist in the comparison of clustering algorithms, in choosing an algorithm to use, as well as to decide when a clustering algorithm has stabilised. The ratio of between- and within-cluster variation is frequently used for this last purpose.

The total-cluster variation is defined by

$$\begin{aligned} T(C) &= \frac{1}{2}\sum_{i=1}^{N}\sum_{i'=1}^{N} d_{ii'} \\ &= \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\left(\sum_{C(i')=k} d_{ii'} + \sum_{C(i')\neq k} d_{ii'}\right) \\ &= W(C) + B(C). \end{aligned}$$

Here $W(C)$ refers to within-cluster variation,

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} d_{ii'}$$

and $B(C)$ refers to between-cluster variation,

$$B(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')\neq k} d_{ii'}.$$

Either $W(C)$ or $B(C)$ can be chosen as the main focus of a clustering problem (Hastie *et al.*, 2009:507-508). As an example, consider again clustering in a banking environment, specifically the problem of classification of transactions into the two groups of recurring and non-recurring transactions on the monthly bank statement. Here it is more important that the within-cluster variation be minimised and the transactions in a certain recurring cluster be almost identical. If a transaction is incorrectly labelled as recurring and an interaction with the client follows from this, there might be a reputational risk for the bank. The between-cluster variation is of less importance as the different recurring transactions can be quite similar as well.

2.2 *K*-means Clustering

K-means clustering is a well-established and widely used procedure, having been proposed in the 1950s. Early references to this approach are Lloyd (1957) and Forgy (1965). The term "K-means" was proposed by MacQueen (1967).

In K-means clustering the value of the parameter K (the number of clusters to be identified in the data) has to be specified upfront. This is a difficult task, which will be discussed in more detail later. The K-means algorithm assigns each of the N observations in the dataset to one of the K non-overlapping clusters.

To illustrate the application of the K-means clustering algorithm a toy dataset generated from two standard bivariate Gaussian distributions is considered. In this dataset there are $p = 2$ variables and three distinct classes of observations, $K = 3$. The dataset is illustrated in Figure 1.

Now consider the cluster encoder, $k = C(i)$, in more detail. Define C_k to represent the indices of the observations in the k^{th} cluster and $C = \{C_1, C_2, ..., C_K\}$. There are two requirements the K clusters found by K-means have to adhere to:

1. $C_1 \cup C_2 \cup ... \cup C_K = \{1, 2, ..., N\}$ (Each observation has to belong to at least one cluster)
2. $C_k \cap C_{k'} = \varnothing, \forall k \neq k'$ (Each observation can belong to no more than one cluster)

This means that there are $\frac{1}{K!}\sum_{k=1}^{K}(-1)^{K-k}\binom{K}{k}k^N$ possible cluster assignments C (Jain & Dubes, 1988). There are no specific constraints on the number of observations allowed or required per cluster. This amount of calculations are completely infeasible. The iterative K-means algorithm is used as an approximation to the true search through all possible clusters (Hastie *et al.*, 2009:508; James, Witten, Hastie & Tibshirani, 2013:386). For clarity, C_k returns the indices of the observations in the k^{th} cluster and $C(i)$ returns the cluster label of the i^{th} observation.

Figure 1: Standard normal data, $p = 2$ and $K = 3$

Squared Euclidean distance, defined before, is used as the dissimilarity measure of the K-means algorithm. The within-cluster variation when using squared Euclidean distance as dissimilarity measure is

$$\begin{aligned} W(C) &= \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2 \\ &= \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k}\|\boldsymbol{x}_i-\boldsymbol{x}_{i'}\|^2. \end{aligned}$$

This expression for $W(C)$ can be simplified to

$$W(C) = \sum_{k=1}^{K} N_k \sum_{C(i)=k}\|\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k\|^2$$

where $\bar{\boldsymbol{x}}_k = (\bar{x}_{1k}, ..., \bar{x}_{pk})$ and $N_k = \sum_{i=1}^{N} I(C(i) = k)$ (Weatherwax & Epstein, 2018:110-112). The proof of this result can be found in Appendix A.

$W(C)$ defined here is the quantity that K-means clustering would like to minimise. This means that the total distances between the cluster mean $\bar{\boldsymbol{x}}_k$ and the N_k observations in the cluster are minimised. This leads to the definition of a criterion for optimising the cluster encoder and the mean values of the clusters at the same time. The criterion has the form

$$C^* = \min_{C,\{\boldsymbol{m}_k\}_1^K} \sum_{k=1}^{K} N_k \sum_{C(i)=k}\|\boldsymbol{x}_i-\boldsymbol{m}_k\|^2,$$

where C is the set of cluster index values as defined above and $\boldsymbol{m}_k$ is the mean vector of cluster k. The notation $\boldsymbol{m}_k$ is used, instead of $\bar{\boldsymbol{x}}_k$, as this refers to a more general definition

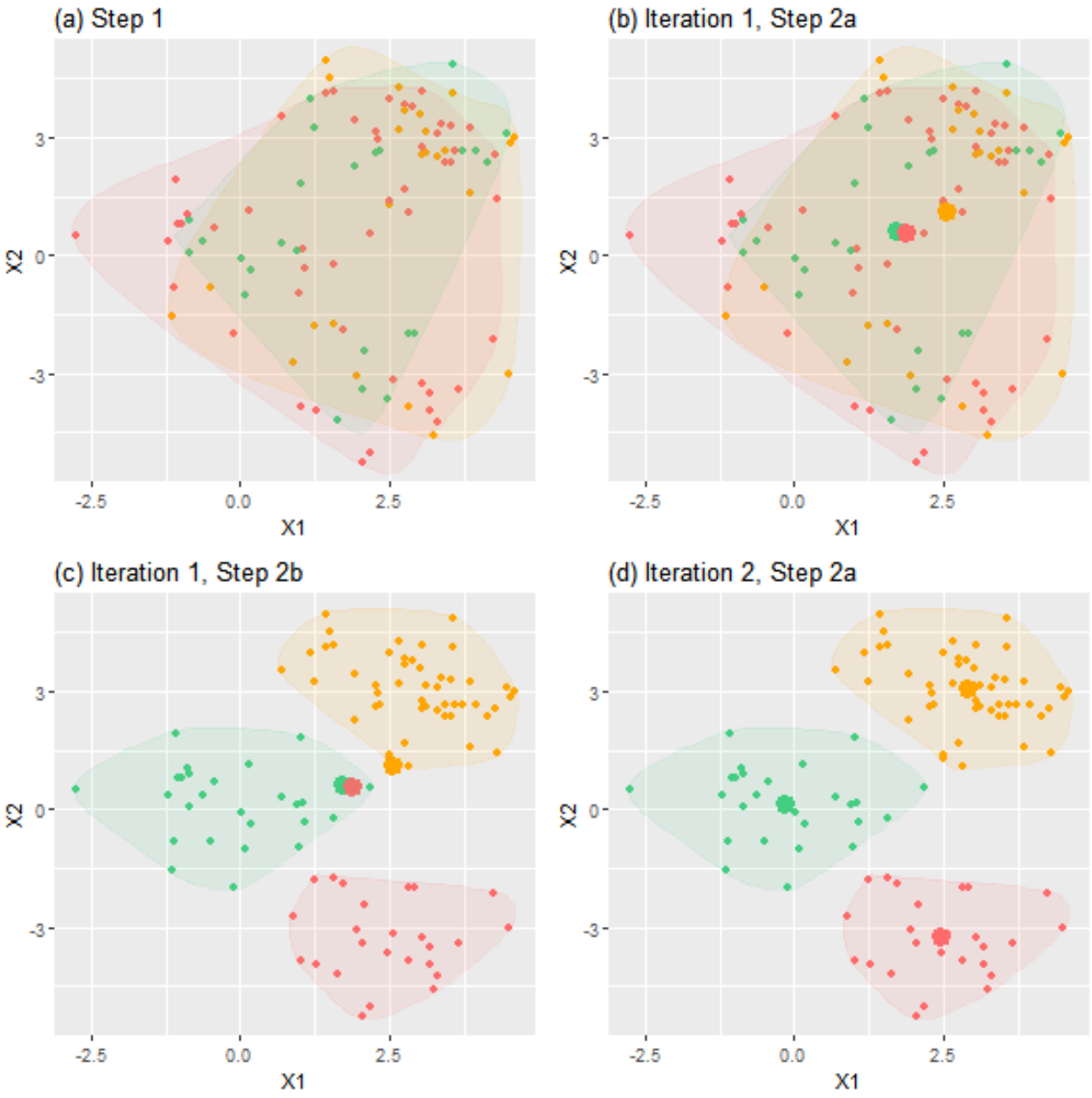


Figure 2: Figures showing the iterations of the K-means clustering algorithm

of a cluster center. For K-means clustering it happens to represent the mean vector, but it could also be the median vector, for example (Hastie *et al.*, 2009:509-510). Below we find the iterative algorithm for K-means clustering.

Algorithm for K-Means Clustering:

1. Specify the value for K.
2. Initialise $C = \{C_1, ..., C_K\}$ by randomly assigning every observation to one of the K clusters.
3. Iterate until the cluster assignment C stabilises:
   a. Calculate the mean vectors for the K clusters through $\boldsymbol{m}_k = \frac{1}{N_k} \sum_{\boldsymbol{x}_j \in C_k} \boldsymbol{x}_j$, yielding $\{\boldsymbol{m}_1, \boldsymbol{m}_2, ..., \boldsymbol{m}_K\}$.

   b. Update C by assigning each observation to the cluster label of the closest mean vector $\boldsymbol{m}_k$, using squared Euclidean distance:

$$C(i) = \underset{1 \leq k \leq K}{\operatorname{argmin}} \|\boldsymbol{x}_i - \boldsymbol{m}_k\|^2.$$

The K-means algorithm performs well when the underlying clusters in the data are spherical, but will struggle to identify elongated cluster shapes. K-means also performs well when the underlying clusters are compact, meaning the observations in a cluster lie very close together. In Figure 2(a)-(d) we can clearly see how the iterations of the K-means cluster algorithm evolve. The toy dataset was used for this visualisation with only one random initialisation. It almost seems that the algorithm has already converged after only two iterations, but this will not be the case if p and K increase.

A property of the K-means algorithm is that it will always reach a point where the cluster labels C stabilise. We are not, however, guaranteed that this is the optimal solution. The K-means algorithm may return different results for different random initialisations of the cluster means $\{\boldsymbol{m}_1, \boldsymbol{m}_2, ..., \boldsymbol{m}_K\}$. To counter this problem a wide range of initial cluster assignments should be made and the final within-cluster variation for each model should be noted. The best model can be chosen as the model that has the lowest final value for $W(C)$ (Hastie *et al.*, 2009:510; James *et al.*, 2013:388; Xu & Wunsch II, 2009:68).

2.3 K-medoids clustering

K-medoids clustering is similar to K-means clustering as both methods address the same optimisation problem. The difference lies in the fact that the centers of the clusters for K-medoids clustering are not taken as the mean vectors but as observations from the dataset. This means that we will have K of the N observations in the dataset chosen as the cluster centers. These K center observations will be called the *representative objects* or *medoids* and

can already contain a lot of information about the data that are being investigated. We can consider these medoids as a sample reduction from N to K and we hope that the medoids summarise the characteristics of their closest neighbours to a certain extent (Kaufman & Rousseeuw, 1990:68).

The clusters are formed around these medoids by including the medoid itself and all the observations lying closest to it. In this study closeness is defined by squared Euclidean distance. By using representative observations as the cluster centers, the clusters returned by the algorithm are more robust to outliers than K-means clusters. The clusters will also be spherical in nature and this method cannot be used to find irregularly shaped clusters (Hastie *et al.*, 2009:516).

Similar to K-means clustering, we find a few slightly different algorithms linked to the K-medoids problem. As an example of a K-medoids algorithm we will discuss the algorithm called Partitioning Around Medoids (PAM) suggested by Kaufman and Rousseeuw in 1987. PAM is known for its two phases, the BUILD phase and the SWAP phase. In the BUILD phase a collection of K medoids are put together as an initialisation for the SWAP phase. In the SWAP phase each cluster is investigated and different observations are tested to possibly replace one of the current medoids (Kaufman & Rousseeuw, 1990:102).

We will now discuss the PAM algorithm in more detail. There are two groups of indices that we need to keep track of:

S: set of indices ($S \leq K$) of observations that have been chosen as medoids
U: set of indices of observations that have not been chosen as medoids.

These two sets are defined such that $O = S \cup U$ is the set of all N observation indices. There are also two dissimilarity measures we need to calculate and update for each observation index $\ell \in O$:

D_ℓ: dissimilarity between $\boldsymbol{x}_\ell$ and the closest medoid indexed in S
E_ℓ: dissimilarity between $\boldsymbol{x}_\ell$ and the second closest medoid indexed in S.

Every time S or U changes in the algorithm, D_ℓ and E_ℓ have to be updated. The part of the PAM algorithm that is associated with the BUILD phase can be found below.

Algorithm for Partitioning Around Medoids (PAM) BUILD phase:

1. Specify the value of K and note that $U = O$.
2. Identify an observation with index h from U, such that this observation has the smallest sum of dissimilarities from all other observations, $\min_h \sum_{j=1 \neq h}^{N} d_{hj}$.
3. Initialise the set S by placing the index h in S. Note that now $S = \{h\}$ and $U = \{1, ..., h-1, h+1, ..., N\}$ and $U \neq O$.
4. Randomly choose an index from U, say i. $\boldsymbol{x}_i$ can now be tested as the next medoid, as described in Step 5 to Step 9.

5. For an observation with index $j \neq i \in U$ calculate D_j, as defined above, and d_{ji}, from the chosen dissimilarity measure for the algorithm.
6. Calculate $C_{ji} = \max\{(D_j - d_{ji}), 0\}$ for every $j \in U \neq i$.
7. Calculate $g_i = \sum_{\{j \neq i\} \in U} C_{ji}$ for all $i \in U$.
8. Choose i so that the value g_i is maximised, $\max_i g_i$.
9. Add i to index set S.
10. Iterate Step 4 to 8 until set S is of size K.

In Step 3 of the BUILD phase, S is initialised with the observation that acts as the center, or the medoid, of the whole observation set. We also look a little closer at the value C_{ji} calculated in Step 6 of the BUILD phase. When $C_{ji} > 0$ and therefore $D_j > d_{ji}$, it indicates that $\boldsymbol{x}_j$ lies closer to $\boldsymbol{x}_i$ than to any of the medoids indexed in S. This means that $\boldsymbol{x}_j$ provides evidence that $\boldsymbol{x}_i$ should also be included in the set of medoids. When $C_{ji} = 0$ and therefore $D_j \leq d_{ji}$, it means that $\boldsymbol{x}_j$ lies closer to a medoid indexed in S than to $\boldsymbol{x}_i$. This provides evidence against $\boldsymbol{x}_i$ and recommends that it should not be included in the set of medoids (Kaufman & Rousseeuw, 1990:102-103; Simovici, 2011:1-2).

After the BUILD phase we have a set of medoid indices S of size K and a set of indices U of size $N - K$ that represent the observations not selected as medoids. In the SWAP phase we will now identify index pairs $(i, h) \in S \times U$, where $i \in S$ and $h \in U$ such that placing i in U and h in S will improve the clustering. The part of the PAM algorithm that is associated with the SWAP phase can be found below.

Algorithm for Partitioning Around Medoids (PAM) SWAP phase:

1. Choose an observation index, say i, from S and an observation index, say h, from U, to be considered for the swap.
2. Choose a third observation index, say $j \neq h$, from set U.
3. Calculate D_j and E_j, as defined above, and d_{ji} and d_{jh}, from the chosen dissimilarity measure for the algorithm.
4. Calculate C_{jih} according to the conditions set out in Table 1 and Table 2.
5. Calculate $T_{ih} = \sum_{\{j \neq h\} \in U} C_{jih}$.
6. Choose i and h so that the value T_{ih} is minimised, $\min_{i,h} T_{ih}$.
7. If $T_{ih} < 0$, carry out the swap and return to Step 1. If $T_{ih} \geq 0$ end the algorithm.

From Step 4 in the SWAP part of the PAM algorithm and Table 1 and Table 2 we consider different cases for computing the value of C_{jih}. Firstly, we consider the row of N/A values

Table 1: Values of $\boldsymbol{C}_{jih}$

	$\boldsymbol{D}_j < \boldsymbol{d}_{jh}$	$\boldsymbol{D}_j = \boldsymbol{d}_{jh}$	$\boldsymbol{D}_j > \boldsymbol{d}_{jh}$	**Summary**
$\boldsymbol{D}_j < \boldsymbol{d}_{ji}$	0	0	$d_{jh} - D_j$	$\min\{(d_{jh} - D_j), 0\}$
$\boldsymbol{D}_j = \boldsymbol{d}_{ji}$	Table 2	Table 2	Table 2	Table 2
$\boldsymbol{D}_j > \boldsymbol{d}_{ji}$	N/A	N/A	N/A	N/A

Table 2: Values of $\boldsymbol{C}_{jih}$

	$\boldsymbol{E}_j < \boldsymbol{d}_{jh}$	$\boldsymbol{E}_j = \boldsymbol{d}_{jh}$	$\boldsymbol{E}_j > \boldsymbol{d}_{jh}$	**Summary**
$\boldsymbol{D}_j = \boldsymbol{d}_{ji}$	$E_j - D_j$	$E_j - D_j$	$d_{jh} - D_j$	$\min\{d_{jh}, E_j\} - D_j$

in Table 1 which indicates that $D_j > d_{ji}$ cannot occur. As i is in the set S, $\boldsymbol{x}_j$ cannot be further away from its closest medoid in S than from $\boldsymbol{x}_i$.

Secondly, we consider the case where $D_j < d_{ji}$, which means that there is a medoid in S that is closer to $\boldsymbol{x}_j$ than to $\boldsymbol{x}_i$. We now look at the dissimilarity to $\boldsymbol{x}_h$ from $\boldsymbol{x}_j$ as well. If $\boldsymbol{x}_j$ lies closer to its closest medoid indexed in S than to $\boldsymbol{x}_i$ or $\boldsymbol{x}_h$, it will not provide evidence in favour of the swap of $\boldsymbol{x}_i$ and $\boldsymbol{x}_h$. The same goes for the case where $\boldsymbol{x}_j$ has the same distance to its closest medoid indexed in S than to $\boldsymbol{x}_h$. If however, $\boldsymbol{x}_j$ lies closer to $\boldsymbol{x}_h$ than to its closest medoid indexed in S, it will provide evidence of the size $d_{jh} - D_j$ for the swap of $\boldsymbol{x}_i$ and $\boldsymbol{x}_h$.

Thirdly, we consider the row in Table 1 for $D_j = d_{ji}$, referring to Table 2. When $D_j = d_{ji}$, then $\boldsymbol{x}_i$ is the closest medoid indexed in S to $\boldsymbol{x}_j$ and we now have to consider the second closest medoid indexed in S to $\boldsymbol{x}_j$. It is important to remember that we are looking for evidence that $\boldsymbol{x}_i$ and $\boldsymbol{x}_h$ should be swapped by investigating their proximities to $\boldsymbol{x}_j$. If $\boldsymbol{x}_i$ is already the closest medoid to $\boldsymbol{x}_j$, we can also investigate a second closest medoid to see if $\boldsymbol{x}_h$ lies even further away from $\boldsymbol{x}_j$ than it's second closest medoid.

Now if $\boldsymbol{x}_j$ is closer to the second closest medoid indexed in S than to $\boldsymbol{x}_h$, the evidence provided for the swap of $\boldsymbol{x}_i$ and $\boldsymbol{x}_h$ is $E_j - D_j$. The same evidence is provided if $\boldsymbol{x}_j$ is exactly the same distance from the second closest medoid indexed in S than from $\boldsymbol{x}_h$. The value of $E_j - D_j$ will always be positive as the dissimilarity of the second closest medoid indexed in S to $\boldsymbol{x}_j$ is necessarily larger than the closest medoid indexed in S. If $\boldsymbol{x}_j$ is closer to $\boldsymbol{x}_h$ than to the second closest medoid indexed in S the contribution to the swap will be $d_{jh} - D_j = d_{jh} - d_{ji}$. This value can be positive or negative. The order in which the swaps are considered for the algorithm should not affect the outcome as all combinations of $\{S, U\}$ are considered (Kaufman & Rousseeuw, 1990:103-104; Simovici, 2011:2-3).

2.4 Hierarchical Clustering

Hierarchical clustering is a collection of clustering methods that return nested clusters. This means that a hierarchical structure is imposed on the data. It is different to K-means or K-medoids clustering as there is no specification for the desired number of clusters before running the algorithm. The goal of hierarchical clustering is to place all the observations

in one cluster at the top of a tree structure and after following the clustering process every observation is in its own cluster at the bottom of the tree, or the other way around. The clusters containing only one observation are called singletons. The tree structure delivered by hierarchical clustering is called a dendrogram. It is a visual aid in making a decision on the appropriate number of clusters and it also captures all the information that is gathered in the clustering process (Hastie *et al.*, 2009:520-522; James *et al.*, 2013:390; Xu & Wunsch II, 2009:31).

The hierarchical structure brings with it an advantage of faster computing time, because of the successive nature of the splits and no concerns about the algorithm not converging. This has a drawback as well, because once a cluster is formed it cannot be split again and vice versa, once a cluster has been split it cannot be joined together again. There are no error-correction possibilities in hierarchical clustering (Everitt, Landau, Leese & Stahl, 2011:71; Kaufman & Rousseeuw, 1990:44-45; Xu & Wunsch II, 2009:40). We will discuss agglomerative hierarchical clustering, as well as divisive hierarchical clustering, which are two different approaches to growing a dendrogram.

2.4.1 Agglomerative Hierarchical Clustering

Agglomerative hierarchical clustering, also called Bottom-up clustering, starts at the bottom of a tree with N singletons and groups observations into clusters until it reaches one big cluster at the top. The agglomerative hierarchical clustering algorithm can be found below.

Algorithm for Agglomerative Hierarchical Clustering:

1. Choose a dissimilarity measure and view the dataset as N singletons.
2. Iterate until all observations are in one cluster, i.e. $i = N, N-1, ..., 2$:
   a. Calculate $\binom{i}{2} = \frac{i(i-1)}{2}$ pairwise dissimilarities among the i clusters.
   b. Identify the pair of clusters with the smallest pairwise dissimilarity.
   c. Create a new cluster by merging all the observations from the clusters identified in (b) together so that there are now $i-1$ clusters.

We need a strategy to determine the dissimilarity between two clusters if either of them, or both, include more than one observation. We introduce the concept of linkage to assist with this. In Table 3 and Figure 3 we can see the seven kinds of linkage that we will consider: complete, single, average, centroid, median, McQuitty and Ward linkage. Figure 3 is again based on the $K = 3$ toy dataset. The forms of linkage in Table 3 are explained in terms of two arbitrary clusters, G and H (Everitt *et al.*, 2011:76-79; James *et al.*, 2013:394-395).

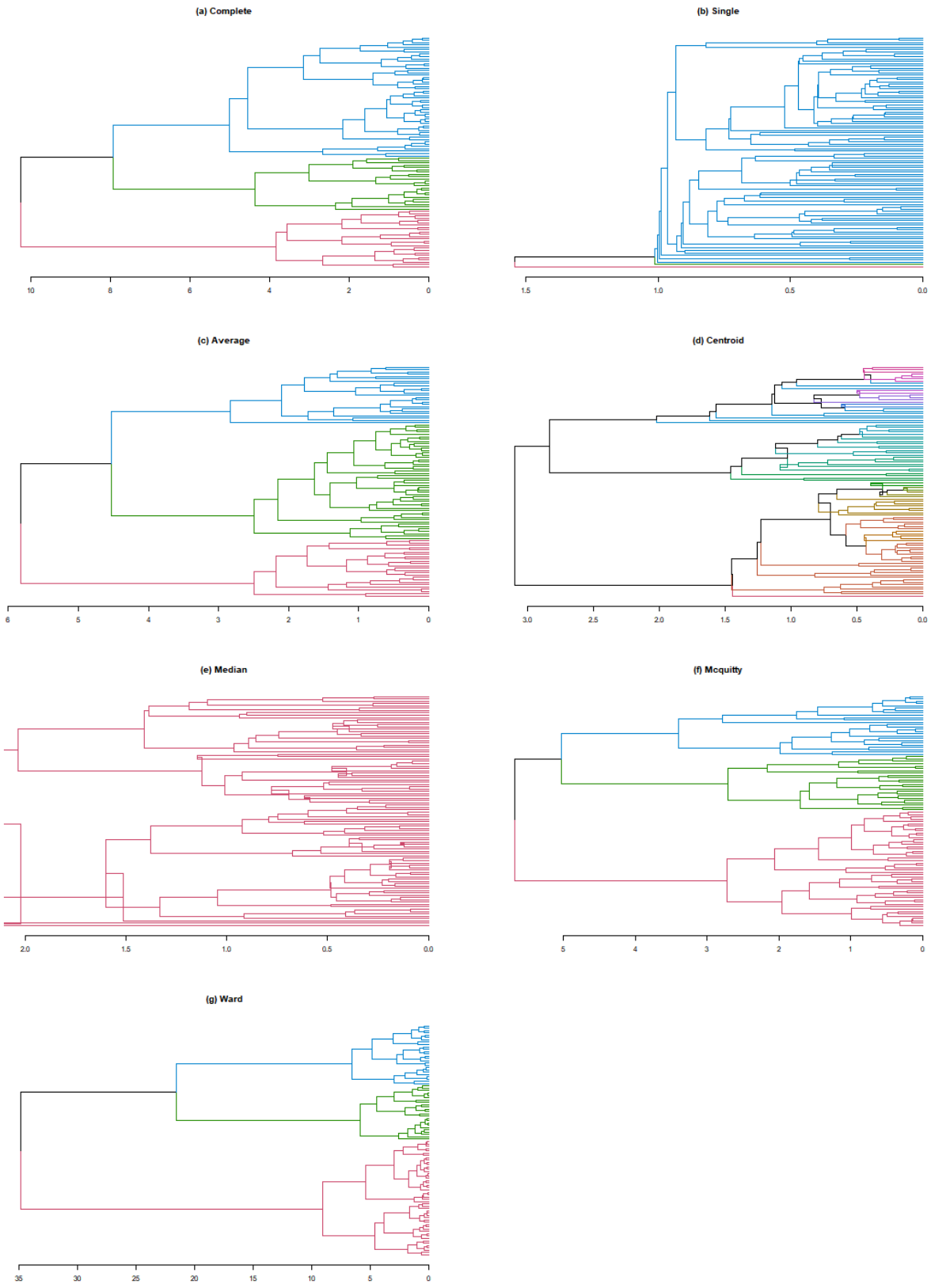


Figure 3: Different forms of linkage for $K = 3$ Toy Data

Table 3: Linkage for Agglomerative Hierarchical Clustering

Linkage Name	**Formula**	**Description**
Complete linkage	$d_{\text{CL}}(G,H) = \max_{\{i\in G, i'\in H\}} d_{ii'}$	Identifies the maximum dissimilarity between all pairs of observations in cluster G and cluster H
Single linkage	$d_{\text{SL}}(G,H) = \min_{\{i\in G, i'\in H\}} d_{ii'}$	Identifies the minimum dissimilarity between all pairs of observations in cluster G and cluster H
Average linkage	$d_{\text{AL}}(G,H) = \frac{1}{N_G \times N_H} \sum_{i\in G} \sum_{i'\in H} d_{ii'}$	Calculates the unweighted average dissimilarity between all pairs of observations in cluster G and cluster H
Centroid linkage	$d_{\text{CentL}}(G,H) = d(\bar{\boldsymbol{x}}_G, \bar{\boldsymbol{x}}_H)$	Calculates the dissimilarity between the mean vectors of cluster G and cluster H
Median linkage	$d_{\text{MedL}}(G,H) = d(\text{median}_G, \text{median}_H)$	Calculates the dissimilarity between the median vectors of cluster G and cluster H
McQuitty linkage	$d_{\text{McL}}(G,H) = \frac{1}{2} \sum_{i\in G} \sum_{i'\in H} d_{ii'}$	Calculates the weighted average dissimilarity between all pairs of observations in cluster G and cluster H
Ward linkage	$d_{\text{WardL}}(G,H) = \frac{N_G \times N_H}{N_G + N_H} \|\bar{\boldsymbol{x}}_G - \bar{\boldsymbol{x}}_H\|^2$	Identifies the within-cluster variance when cluster G and cluster H are joined together, using squared Euclidean distance

In Figure 3 we see the tree structure of the hierarchical clustering dendrogram, with all the trees being displayed at a 90° anti-clockwise rotation. At the top, or in this case on the left, of the dendrogram the root node can be seen and this represents all the observations in one cluster. At the bottom of the dendrogram, in this case on the right, all the singletons can be seen, also called leaves. The height of the dendrogram reflects the value of dissimilarity at which two clusters joined together. The dendrogram can be cut at various heights to result in different numbers of clusters, K (James *et al.*, 2013:391-394; Xu & Wunsch II, 2009:31).

Let us discuss the different forms of linkage from Table 3 in more detail. Complete linkage can also be called furthest-neighbour linkage. It is known for returning well balanced clusters and this can be seen in the toy data example in Figure 3(a). Complete linkage is useful when the goal is to find compact clusters that lie relatively close together. It also has an effect called space dilation, which refers to the fact that similar observations may only be joined together very low down on the dendrogram. Only the observations that are furthest from each other in clusters G and H come into play in this form of linkage and the observations that lie between them, that might be quite similar, are ignored (Hastie *et al.*, 2009:523-524; James *et al.*, 2013:394-395; Kaufman & Rousseeuw, 1990:48,227). It is a good candidate to be used in the real world data example to follow in a later chapter and will also be investigated in the simulation study chapter.

Single linkage can also be called nearest-neighbour linkage. Some problems are foreseen for single linkage as it can cause a dendrogram structure where singletons are added one at a time to one big growing cluster. This phenomenon is referred to as chaining and returns extremely unbalanced clusters. This does not give much insight into the true underlying structure in the data. There exists the possibility to take advantage of this chaining effect which arises when the goal is to find elongated clusters. Single linkage also has an effect called space contraction, the opposite of complete linkage. This refers to the fact that dissimilar observations may be joined together very high on the dendrogram. Only the observations that are closest to each other in clusters G and H come into play in this form of linkage and the observations that lie outside of them, that might be quite dissimilar, are ignored (Hastie *et al.*, 2009:523-524; James *et al.*, 2013:394-395; Kaufman & Rousseeuw, 1990:48,227). Chaining can clearly be seen in Figure 3(b) for the toy data example.

Average linkage is a compromise between complete linkage and single linkage. It can also be called group average (GA) or unweighted pair-group method using the average approach (UPGMA). It is called unweighted because we explicitly account for the cluster sizes. It aims to produce compact clusters, as far apart as possible. The compactness of a cluster can be measured by its diameter, which is the largest dissimilarity among the member observations of a certain cluster G, i.e.

$$D_G = \max_{\{i \in G, i' \in G\}} d_{ii'}.$$

Average linkage also has an effect called space conservation, which refers to the fact that all observations in cluster G and H contribute to the linkage and none of them are ignored (Everitt *et al.*, 2011:76,79; Hastie *et al.*, 2009:523-525; James *et al.*, 2013:394-395; Kaufman & Rousseeuw, 1990:227). Average linkage can be found in Figure 3(c) for the toy data example and will be investigated further as well.

Centroid and median linkage can be found in Figure 3(d) and (e), respectively. They also have the alternative names of unweighted pair-group method using the centroid approach (UPGMC) and weighted pair-group method using the centroid approach (WPGMC), respectively. It is clear that the dendrograms returned by these linkage methods are uninterpretable. This phenomenon of clusters joining together at a lower height than the individual clusters are called reversals or inversions (Everitt *et al.*, 2011:76-77,79; James *et al.*, 2013:394-395). Again, even though there might be scenarios where these forms of linkage are useful, they will not be explored further.

McQuitty linkage can be seen in Figure 3(f) and is also called weighted average linkage or weighted pair-group method using the average approach (WPGMA). This linkage form is similar to average linkage, except that it does not account for the cluster sizes (Everitt *et al.*, 2011:78-79). It may sound counter-intuitive to call this a weighted average and not specify weights explicitly, but by not accounting for the cluster sizes they essentially act as implicit weights. Not explicitly accounting for cluster size in the linkage formula may cause smaller clusters to be assigned larger implicit weights. This may alleviate the bias against small clusters found in the unweighted average linkage. Mcquitty linkage will be investigated further.

Lastly we can see Ward linkage in Figure 3(g) and it returns very well balanced clusters. This form of linkage is similar to centroid linkage, but it uses an explicit weighting of the centroids. It can be seen as a minimisation of the increase in the total within-cluster variance as new clusters are formed (Everitt *et al.*, 2011:77-79; Murtagh & Legendre, 2014:281-285). We will investigate this form of linkage in the simulation study as well.

2.4.2 Divisive Hierarchical Clustering

Divisive hierarchical clustering, also called Top-down clustering, starts from the top of a dendrogram with one big cluster and moves its way downwards by splitting clusters until N singletons are formed at the bottom. This clustering method has been documented less rigorously than agglomerative hierarchical clustering. This is because the number of possibilities for the first combination of singletons for agglomerative hierarchical clustering is $\binom{N}{2}$, whereas for divisive hierarchical the number of possibilities for the first split into two clusters is $2^{N-1} - 1$. It is a useful method if only a few large clusters are desired from a big dataset. There is the option to repeatedly use K-means clustering with $K = 2$ until the data is divided into N singletons. This process would be unfeasible and hard to duplicate as there would a random initialisation at each step (Hastie *et al.*, 2009:526; Kaufman & Rousseeuw, 1990:253). A more replicable algorithm for divisive hierarchical clustering proposed by Macnaughton-Smith, Williams, Dale and Mockett in 1964 can be found below.

Algorithm for Divisive Hierarchical Clustering:

1. All N observations are placed in a single cluster.
2. Iterate until there are N singletons:
   a. Identify the cluster with the largest diameter, call this cluster G.
   b. Identify the observation $\boldsymbol{x}_i$ with the largest average dissimilarity to all the other observations in G by $\frac{1}{N_G-1}\sum_{i' \in G \neq i} d_{ii'}$.
   c. Create a new cluster, called H, containing this observation as the first member.
   d. For cluster G iterate until all Diff_i-values become negative:
      i. Calculate

$$\text{Diff}_i = \frac{1}{N_G \times (N_G - 1)} \sum_{i \in G} \sum_{i' \neq i \in G} d_{ii'} - \frac{1}{N_G \times N_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'}$$

      for every observation i in G.

      ii. Identify the observation i with the largest positive value for Diff_i and place it in cluster H.

For the very first iteration of the algorithm the cluster identified in Step 2(a) will be the original cluster containing all the observations. At every iteration of Step 2 the cluster H is called the splinter group, because it is the cluster that splinters off from the original, G. The goal is to find all the observations in G that lie closer to observations in the splinter group, H, than to the other observations in G and then to move them over until the algorithm stabilises. The Diff_i-values measure the average dissimilarity in G compared to the average dissimilarity between G and H. If $\text{Diff}_i > 0$ it means that there are still observations in G that are more similar to H than to G. As soon as $\text{Diff}_i \leq 0, \forall i$ the two clusters have stabilised (Hastie *et al.*, 2009:526,528; Kaufman & Rousseeuw, 1990:271).

2.5 Practical Issues

2.5.1 Missing Values

Missing values can be treated in several ways. If one of a pair of observation values being compared is missing, then the specific entry can be omitted from the dissimilarity matrix D. This will lead to a dissimilarity between two rows of X defined as

$$D(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \frac{p}{p - \sum_{j=1}^{p} \delta_{ii'j}} \sum_{j, \delta_{ii'j}=0} d_j(x_{ij}, x_{i'j}),$$

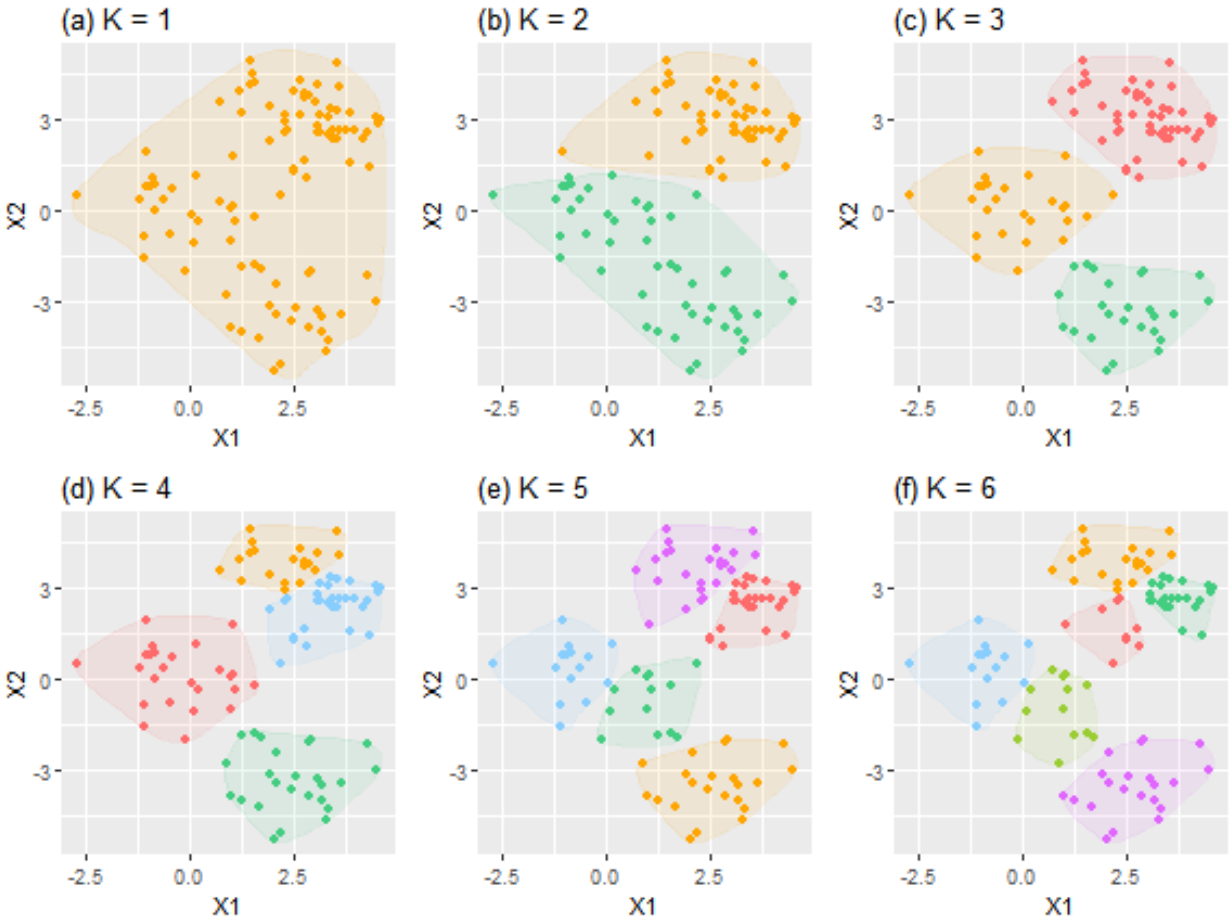


Figure 4: Figures showing the evolution of K-means clustering for increasing values of K

where $\delta_{ii'j} = I(x_{ij}$ or $x_{i'j}$ is missing).

Another option is to impute the missing values, but this can become a problem if imputed values start to play key roles in the clustering algorithm. For example, if an observation with imputed missing values are chosen as a medoid in K-medoids clustering it would make the clustering results less trustworthy. If there are missing values present in categorical values, then a separate category can be created labelled "missing" (Hastie *et al.*, 2009:507; Xu & Wunsch II, 2009:20).

We are not worried about missing values in this study as the dataset from Bank C is guaranteed to be free of missing values. This is because of the process by which the data is collected. When a client performs a transaction with Bank C the information surrounding the transaction is automatically recorded.

2.6 How to choose K

Care has to be taken when specifying the parameter K. If the parameter is specified too large, unnecessary splitting of groups in the data that belong together may occur. If the parameter is specified too small, then some groups returned by the cluster algorithm will be heterogeneous and this may lead to wrong conclusions. For example, in K-means clustering we are using the minimum within-cluster dissimilarity as the target and there is a risk of choosing K so large that the within-cluster dissimilarity almost reaches zero. This could be seen as overfitting in the same way as growing a decision tree to the point where every terminal node contains only one observation.

Figure 4 shows the different scenarios, for K-means clustering, where K is specified too small in (a) and (b), correctly in (c) and too large in (d) to (f). In this toy dataset, the true number of classes is 3. K-means clustering is used as an example here, but the principle holds for all clustering methods. The danger of any clustering algorithm is that it will always return a set of cluster allocations, no matter how wrong the specification of K may be (Hastie *et al.*, 2009:518-519).

Regarding the two scenarios of choosing K too large or too small, the latter may cause more damage to the final conclusions drawn from the analysis. This is because we are more comfortable with too many homogeneous clusters that can be combined, than with too few clusters that are still heterogeneous and would need further splitting.

The notation for the within-cluster variation vector, defined as the set $\boldsymbol{W} = \{W_1, W_2, ..., W_{K'}\}$ before, will be used again and K' as the specified number of clusters. This K' may or may not be different from the underlying true number of clusters K. We now discuss specific methods of choosing K' (Hastie *et al.*, 2009:518-519). To clarify, K refers to the true underlying number of clusters in the data and will not be known if the data is truly unsupervised. K' refers to the specified number of cluster for the dataset by the analyst. The notation k is used while different possibilities for K' is being investigated and the final value for K' has not yet been specified.

Firstly we discuss the **elbow method** which is a heuristic approach to choosing K'. We increase k from a chosen minimum, in steps of size one, to a chosen maximum and record the values of W_k at each k. The value of W_k will decrease as k increases. We can draw the plot of these decreasing W_k-values and look for a "kink" or an "elbow" in the plot. The "kink" can be found where the rate of descent in the within-cluster variation suddenly becomes less steep. An elbow graph for our toy data example can be seen in Figure 5. This figure has a clear turning point at $k = 3$ and we should specify $K' = 3$, which is the correct value for K. The bend in the plot can, however, be much more subtle, which may cause difficulties in identifying a corresponding K-value (Hastie *et al.*, 2009:519).

Another method for choosing K' is through the **gap statistic**. This is illustrated in Figure 6(b) for our toy dataset. This statistic was suggested in 2001 by Tibshirani, Guenther and Hastie. At a number of clusters k the gap statistic is the difference between the expected value under a distribution that has no natural clustering present, called a null distribution,

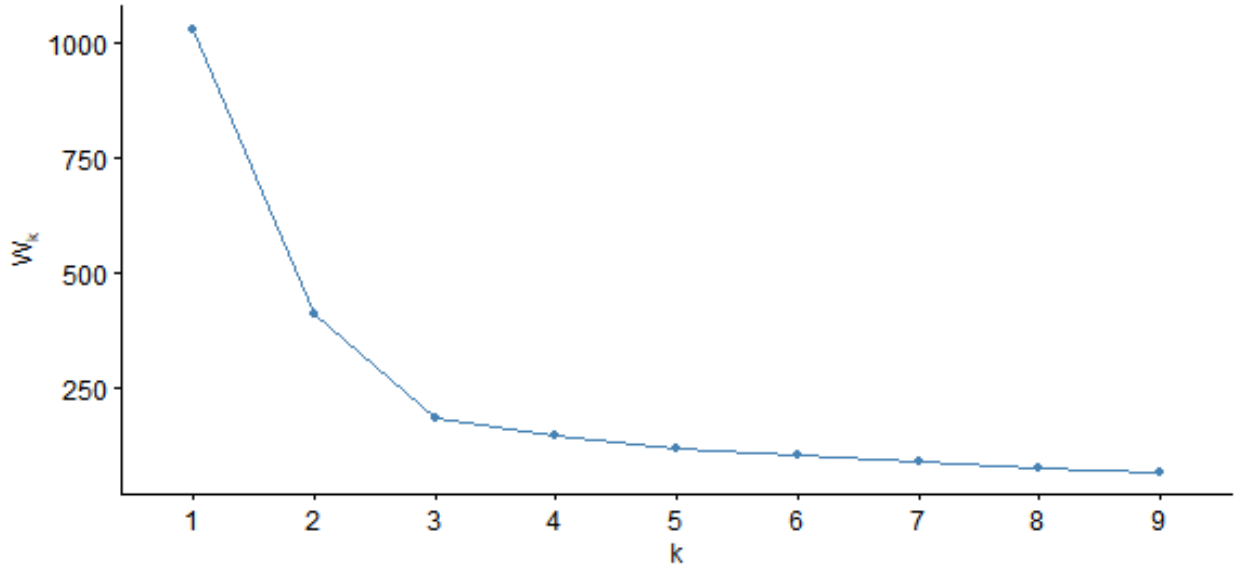


Figure 5: The elbow method for choosing K'

and $\log W_k$. More formally we define the gap statistic by

$$\text{Gap}_N(k) = \mathbb{E}_N^*\{\log W_k\} - \log W_k$$

where $\mathbb{E}_N^*$ is the expectation under the null distribution for the sample of size N. This expectation is not known, but can be estimated using the Bootstrap method. Another popular alternative is choosing the null distribution to be the uniform distribution, such that the gap statistic formula above simplifies to

$$\text{Gap}_N(k) = \frac{1}{B}\sum_{b=1}^{B}\{\log W_{kb}^*\} - \log W_k,$$

where W_{kb}^* represents the b^{th} Monte Carlo replicate of the within-cluster variation under the uniform distribution for the current number of clusters k. In Figure 6(a) the red line represents $\frac{1}{B}\sum_{b=1}^{B}\{\log W_{kb}^*\}$ at different values of k and the green line is the $\log W_k$-values at different values of k. The largest difference between these two lines indicates the appropriate value of K'. In Figure 6(a) we can already see that the largest difference lies at $k = 3$, as we expected.

We can find the standard deviation of $\log W_{kb}^*$ by letting $\bar{l} = \frac{1}{B}\sum_{b=1}^{B}\{\log W_{kb}^*\}$ and computing

$$\text{sd}_k = \sqrt{\frac{1}{B}\sum_{b=1}^{B}\{\log W_{kb}^* - \bar{l}\}^2}.$$

We define $s_k = \text{sd}_k\sqrt{1+\frac{1}{B}}$. The number of chosen clusters K' is found at the smallest value of k where $\text{Gap}_N(k) \geq \text{Gap}_N(k+1) - s_{k+1}$. In Figure 6 we can see that $k = 3$ has the largest

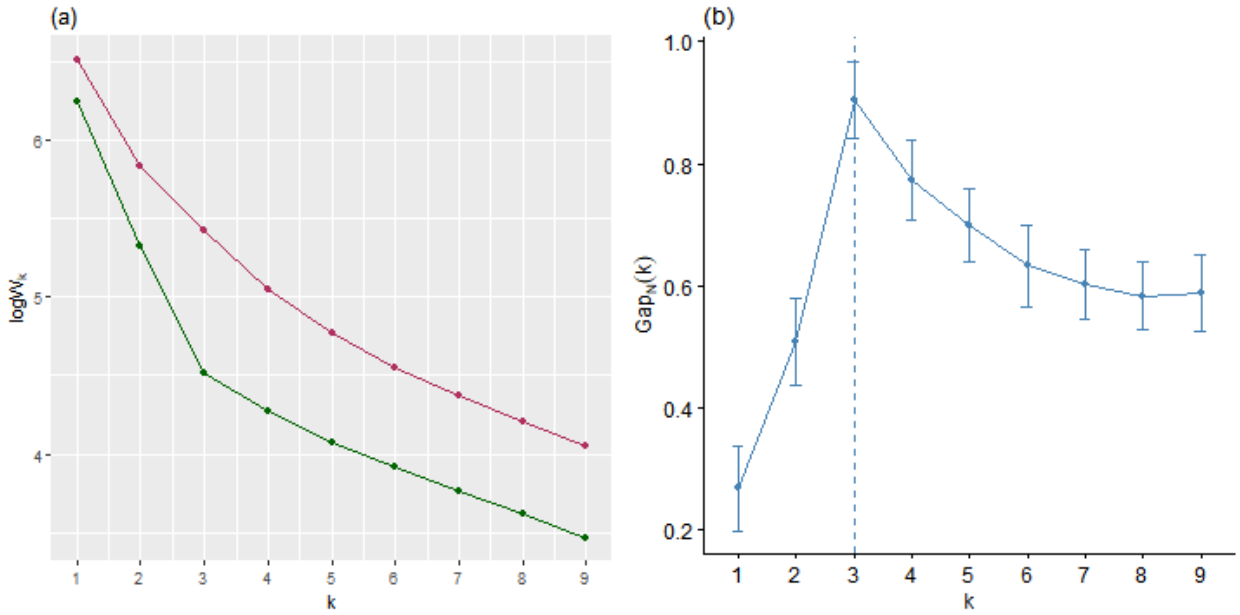


Figure 6: The gap statistic for choosing K'

gap and it lies within one standard error of the value for $k = 4$. Thus we choose $K' = 3$. It is important to note that the gap statistic can also identify if there is no clustering structure in the data as the optimal number of clusters returned will be one (Hastie *et al.*, 2009:519; Tibshirani, Guenther & Hastie, 2001:411-415).

A third method of finding the optimal value of K is called the **average silhouette** method. This method evaluates the tightness of the clusters formed, for a given value of k, and how well they are separated from each other. Consider observation i and assume it was assigned to cluster A. We define $a(i)$ as the average dissimilarity of i to all the other member observations of cluster A. Now we identify the closest neighbour cluster of i. This is done by finding the other cluster with the lowest average dissimilarity to i. We call this dissimilarity $b(i)$. We define the silhouette of observation i as:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}.$$

A large silhouette value means that observation i is nested well inside its own cluster and has a high dissimilarity from its nearest neighbour cluster. The silhouette values are averaged over all N observations for every value of k we possibly want to choose. K' is chosen where the average silhouette is a maximum ($K' = 3$), as can be seen in Figure 7 for the toy dataset (Rousseeuw, 1987:55-57).

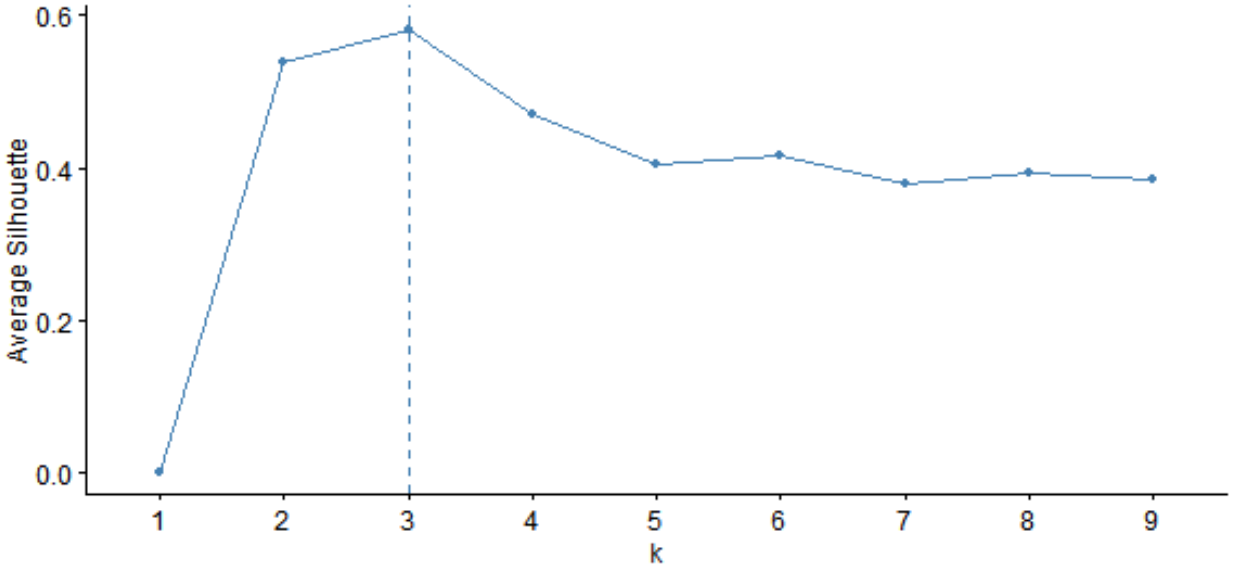


Figure 7: The average silhouette Method for choosing K'

2.6.1 Standardisation of Variables

Standardisation is necessary when variables are measured on vastly different scales. Variables with very large measuring units will have an unduly large influence on the clustering algorithm. Standardisation also inherently means that all the variables are given equal influence, which will also probably not be the case otherwise. One form of standardisation that is popular is defined for entry x_{ij} in matrix X by

$$\frac{x_{ij} - \bar{x}_j}{s_j},$$

where $\bar{x}_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij}$ and $s_j = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_{ij} - \bar{x}_j)^2}$. This leaves the values of every variable with a zero mean and a variance of one. Another possibility is to scale variables according to the minimum and maximum value per variable. For entry x_{ij} in matrix X this is implemented by computing

$$\frac{x_{ij} - \min(\boldsymbol{x}_{.j})}{\max(\boldsymbol{x}_{.j}) - \min(\boldsymbol{x}_{.j})},$$

where $\boldsymbol{x}_{.j} = \{x_{1j}, x_{2j}, ..., x_{Nj}\}$ and would adjust the range of matrix X to $[0, 1]$. The second method of standardisation generally outperforms the first (Everitt *et al.*, 2011:67; Xu & Wunsch II, 2009:22-23).

An illustration of how standardisation affects the outcome of K-means clustering can be seen in Figure 8. Figure 8(a) shows a toy dataset with $p = 2$ and $K = 2$, generated from two Gaussian distributions, each $\mathrm{N}(1, 2)$. The value 3 was added to the X_1-values of the first cluster to translate it to the right. A K-means clustering was performed on this data and the outcome can be seen in Figure 8(b). The toy dataset was then standardised using the mean and the standard deviation, and this can be seen in Figure 8(c). A second K-means

clustering was done on the standardised data and the results are shown in Figure 8(d). The standardisation of the data caused the K-means algorithm to misinterpret the natural groupings in the data (Hastie *et al.*, 2009:506).

The example here is shown for K-means clustering, but other methods of clustering can have a similar experience when variables are standardised. The decision for standardisation should be made with as much *a priori* knowledge about the measuring units of the variables as possible. Clustering can also be performed on the dataset with and without standardisation. The results can then be compared to determine the influence standardisation of the variables has on the outcome.

2.6.2 Weighting of Variables

We can also define a measure for the dissimilarity between two objects to contain a weighting per variable. The weighted dissimilarity across all variables would then be $D(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) = \sum_{j=1}^{p} w_j d_j(x_{ij}, x_{i'j})$. The weights attached to the variables are denoted by w_j and $\sum_{j=1}^{p} w_j = 1$.

If the intention is to give every variable the same influence on the dissimilarity measure, it would not be adequate to set all the weights to the same value, for example $\frac{1}{p}$. Each pair of observations will have a relative contribution to the total average dissimilarity $\overline{D}$. The influence that a specific X_j has on $D(\boldsymbol{x}_i, \boldsymbol{x}_{i'})$ depends on this relative contribution. The average dissimilarity is found by calculating

$$\begin{aligned}\overline{D} &= \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N} D(\boldsymbol{x}_i, \boldsymbol{x}_{i'}) \\ &= \sum_{j=1}^{p} w_j \frac{1}{N^2}\sum_{i=1}^{N}\sum_{i'=1}^{N} d_j(x_{ij}, x_{i'j}) \\ &= \sum_{j=1}^{p} w_j \overline{d}_j,\end{aligned}$$

where $\overline{d}_j$ can be seen as the average dissimilarity amongst the data cases with respect to variable X_j. The relative influence of X_j on $\overline{D}$ is $w_j\overline{d}_j$. So if the goal is equal influence for each variable to the total average dissimilarity, the weight would have to be set at $\frac{1}{\overline{d}_j}$.

When squared Euclidean distance is used as the dissimilarity measure, a weight can be incorporated by transforming the entries of the X matrix. The input of this new weighted algorithm would be the matrix Z, which is of the same size as X. The entries of Z are defined as:

$$z_{ij} = x_{ij}\sqrt{\frac{w_\ell}{\sum_{\ell=1}^{p} w_\ell}},$$

where w_ℓ are the weights attached to the variables. Any clustering algorithm can now be used with the input matrix Z (Hastie *et al.*, 2009:505-506; Weatherwax & Epstein, 2018:112-113). The proof of the validity of this reweighting can be found in Appendix B.

Figure 8: Illustration of scaled variables negatively affecting the outcome of K-Means clustering

Variable selection is also a form of variable weighting as variables are awarded a weight of zero or equal weights summing to one. Even though clustering does not have a specific outcome variable to predict, variable selection may still be beneficial. It can assist in the visualisation of the data in the form of dimension reduction. This will also ease the pressure on clustering algorithms when applied to cases of large p and can improve the quality of the clusters by not clustering the noise (Everitt *et al.*, 2011:63-67).

2.7 Cluster Validity

Clustering is inherently an unsupervised problem. There is no clear and agreed upon way to determine the success of the clustering. Any clustering method will find groups in the data, whether they truly exist or not. Clustering needs to be understood as an investigation into the data and structures that may exist. It can also be used as a form of dimension reduction. However, if we find clusters in the data we would like to know to what extent we can trust the results and use them in practical applications.

There are different kinds of measures for the validity of a clustering. Firstly, we have external measures that can be used to compare a clustering to a given set of cluster labels. These labels can come from *a priori* knowledge about the data or because the data were generated in a specific way with cluster labels. Secondly, we can test the internal measures of the clustering. This mostly relates to hierarchical clustering and determines whether hierarchical clustering structure is present in the data. Thirdly, we need relative measures to compare different clustering outcomes. These outcomes can come from different algorithms applied to the same dataset or from the same algorithm with different parameter initialisations. The intra- and inter-cluster dissimilarity, which can also be referred to as the separation (isolation) and the compactness (cohesiveness) of the clusters, are mostly used in these measures (Everitt *et al.*, 2011:267-268; Hastie *et al.*, 2009:486-487; Xu & Wunsch II, 2009:263-265).

2.7.1 External Criteria

We will define the Rand Index, the Jaccard Index and the Fowlkes and Mallows index for determining the similarity between two clustering outcomes, also referred to as partitions. Let us call the first suggested partition P and the second suggested partition P'. There are a few scenarios that can occur regarding the observations in P and P' and we will describe these in terms of the observations indexed by i and j below.

SS: number of cases where $C(i) = C(j)$ ($\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ reside in the same cluster) for P and P'.
SD: number of cases where $C(i) = C(j)$ for P and $C(i) \neq C(j)$ ($\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ do not reside in the same cluster) for P'.
DS: number of cases where $C(i) \neq C(j)$ for P and $C(i) = C(j)$ for P'.
DD: number of cases where $C(i) \neq C(j)$ for P and $C(i) \neq C(j)$ for P'.

We also define $M = \binom{N}{2} = \frac{N(N-1)}{2} = SS + SD + DS + DD$ as the total number of pairs (i, j)

of observations. Firstly, we define the **Rand Index** by

$$R = \frac{SS + DD}{M}.$$

The Rand Index was proposed by William M. Rand in 1971. The larger the value of R the more similar P and P' are to each other. Also, $R \in [0, 1]$.

Secondly, we define the **Jaccard Coefficient** by

$$J = \frac{SS}{SS + SD + DS}.$$

This coefficient was suggested by Paul Jaccard as early as 1901. Here the value of DD is completely ignored. The larger the value of J the more similar P and P' are to each other and $J \in [0, 1]$.

Thirdly, we define the **Fowlkes and Mallows Index** by

$$FM = \sqrt{\frac{SS}{SS + SD} \times \frac{SS}{SS + DS}}.$$

This coefficient was suggested by E.B.Fowlkes and C.L.Mallows in 1983 and $FM \in [0, 1]$. Again the value of DD is completely ignored. The larger the value of FM the more similar P and P' are to each other. The Fowlkes and Mallows Index progresses toward 1 much more slowly than the Rand Index and might be better at identifying unrelated cluster structures (Everitt *et al.*, 2011:264-265; Xu & Wunsch II, 2009:265-266).

2.7.2 Relative Criteria

The relative criteria can all be used to choose the optimal value for K. But once it has been chosen then these criteria can also be used to compare different clustering algorithms on the same dataset for the same value of K. For this purpose we can use the average silhouette method discussed before. Another popular statistic we can use to measure the validity of a clustering is called the **Dunn Index** suggested by J.C. Dunn in 1974. This index is defined by

$$\mathrm{Du}(K) = \min_{i=1,\ldots,K} \left(\min_{j=i+1,\ldots,K} \left(\frac{D(C_i, C_j)}{\max_{\ell=1,\ldots,K} \mathrm{diam}(C_\ell)} \right) \right),$$

where $D(C_i, C_j) = \min_{\boldsymbol{x} \in C_i, \boldsymbol{y} \in C_j} D(\boldsymbol{x}, \boldsymbol{y})$, the distance between two clusters, and $\mathrm{diam}(C_i)$ is the diameter of a cluster as defined before (largest dissimilarity among the member observations of a cluster). Different definitions of the distance between two clusters and a cluster diameter can be explored, but it will not be investigated in this project. The larger the value of $\mathrm{Du}(K)$ the more compact and well separated the clusters are.

Another index that we will consider is the **Davies-Bouldin Index** suggested by David L. Davies and Donald W. Bouldin in 1979. It is defined by

$$DB(K) = \frac{1}{K} \sum_{i=1}^{K} \left\{ \max_{j \neq i} \left(\frac{e_i + e_j}{D_{ij}} \right) \right\}$$

where $e_i = \frac{1}{N_i}\sum_{j=1}^{N_i}(\boldsymbol{x}_j - \bar{\boldsymbol{x}}_i)^2$ is the average distance between points in cluster i and their centroid, and $D_{ij} = \|\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}}_j\|$ is the distance between the centroids of cluster i and cluster j. A smaller value of $DB(K)$ indicates a more compact and well separated clustering (Xu & Wunsch II, 2009:266-271).

2.8 Summary

In this chapter we investigated the literature available on the traditional clustering methods of K-means, K-medoids and hierarchical clustering. We also looked at the different possible ways of specifying the appropriate number of clusters K. Some practical issues were also discussed, such as missing values, weighting of variables and standardisation. Lastly, the different criteria available for cluster validation were investigated.

In the second part of the literature review we will discuss clustering methods that are appropriate for large and high-dimensional datasets. Specifically, we will investigate CLARA and SOMs.

CHAPTER 3: LITERATURE REVIEW (PART 2)

3 Clustering Large and High-Dimensional Datasets

3.1 Introduction

Some of the methods described in the previous chapter start to fail when N or p becomes large. The reason for this mostly stems from clustering methods taking the dissimilarity matrix D as input. If N is very large and D is of size $N \times N$, it might be impossible to store or even calculate a matrix of that size. There are also issues when p becomes large as the Curse of Dimensionality applies. Most clustering methods rely on distance measures, for example Euclidean distance, to determine proximity between the observations. The idea of dissimilarity might become vague as all points will be lying far away from each other in high dimensional space.

In this chapter we will discuss Clustering Large Applications (CLARA), that was developed to be useful for large datasets, but still struggles to handle high dimensions. We will also discuss Self-Organising Maps (SOMs) that provide a method to map high-dimensional data onto a lower dimensional grid to aid in visualisation and understanding of the data. This grid can then be fed into more classical clustering methods, since the dimensions of the data have been reduced. Lastly we will discuss an adaption to SOMs, called Growing Self-Organising Maps (GSOMs). This method determines the size of the lower dimensional grid as part of the algorithm and does not require it to be specified beforehand.

3.2 Clustering Large Applications (CLARA)

The CLARA algorithm was proposed in 1986 by Kaufman and Rousseeuw and adds a sampling step to the PAM algorithm, to be able to apply it to large datasets. CLARA takes the original dataset as input, because calculating and storing the dissimilarities between the large N observations would be infeasible. The PAM algorithm is fitted to several random samples from the original observations and the most successful clustering is returned. The goals of the two algorithms are the same, to cluster N observations into K groups by minimising the average distances between the observations and their closest medoids. The CLARA algorithm is given below.

Algorithm for CLARA:

1. Specify the value K.
2. Take a random sample of minimum size $40 + 2K$ from the N observations.
3. Fit the PAM algorithm to the sample to determine the K medoids.

4. Assign the N observations to the K clusters according to their closest medoids, using Euclidean distance.
5. Calculate the average distance from all observations to their closest medoids as a summary statistic for the clustering.
6. Repeat Step 2 to 5 for a minimum of 5 times. After the first iteration the K medoids found in the previous iterations are automatically included in the sample.
7. Choose the clustering linked to the minimum average distance as the final clustering.

The CLARA algorithm finds the same kinds of clusters as the PAM algorithm, *i.e.* spherical cluster. This makes sense because the clustering itself is performed by PAM, CLARA is just a wrapper for PAM with a sampling step. The method is also robust to outliers, as that is the nature of using medoids instead of mean centroids. The sample sizes need to be chosen large enough to ensure that all the clusters have some representative observations in the sample. The random sample is taken with replacement, but discarding any duplicate observations. What can be considered as a large N has changed over the years, currently PAM can handle thousands of inputs with a hard upper limit of $N = 65536$ (Kaufman & Rousseeuw 1990:41,126,144-145).

3.3 Self-Organising Maps (SOMs)

In the following sections a detailed discussion of SOMs will be presented. This is a clustering technique with close connections to Neural Networks which is a very popular supervised learning technique. The SOM was introduced in 1982 by Teuvo Kohonen. It is a form of Competitive Learning and a brief explanation of Competitive Learning follows in the next section. In the sections following thereafter we discuss the SOM algorithm, the initialisation of the SOM, the batch SOM, combining SOMs with other clustering methods and useful adaptions to the original SOM.

3.3.1 Competitive Learning

In this type of learning we have a network structure with three layers, an input layer, a competitive layer and an output layer. The competitive layer is similar to the hidden layers, of which several can be present, in a supervised Neural Network. In the Competitive Learning process the input layer accepts one p-dimensional observation at a time, $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, ..., x_{ip}\}$, and each of the input nodes represents one of the p-dimensions, $x_{i\ell}$. The input nodes are fully connected to the competitive layer nodes and each of the competitive layer nodes represents an activation function of the input layer nodes, $s(\boldsymbol{x}_i, \boldsymbol{w}_j)$. Here $\boldsymbol{w}_j$ represents a weight vector that will be applied to the input vector, so that $s(\boldsymbol{x}_i, \boldsymbol{w}_j) = \boldsymbol{w}_j^T \boldsymbol{x}_i = \sum_{\ell=1}^{p} w_{j\ell} x_{i\ell}$; these weight vectors are also called prototypes or neurons (these terms will be used interchangeably).

The number of neurons, say M, will have to be specified beforehand in most of the Competitive Learning algorithms, except when a growing phase is present, like in GSOMs. The prototypes are then labelled $\{\boldsymbol{w}_1, \boldsymbol{w}_2, ..., \boldsymbol{w}_M\}$ and M can be specified to be greater than or equal to the number of desired clusters K, so that $M \geq K$. If we specify the number of neurons to be more than K, a second clustering algorithm will be applied to the final set of neurons to return the exact number of clusters required (Xu & Wunsch II, 2009:111-112).

Only the neuron with the largest value for $s(\boldsymbol{x}_i, \boldsymbol{w}_j)$ is activated and pushed through to the output layer, so that $J = \underset{j}{\text{argmax}}\{s(\boldsymbol{x}_i, \boldsymbol{w}_j)\}$ indexes this neuron. It is also called the Best Matching Unit (BMU) to the input observation, or the winner neuron. This neuron, $\boldsymbol{w}_J$, now receives the opportunity to be adjusted towards the input vector, $\boldsymbol{x}_i$, to resemble it more closely. The equation for this adjustment has the form:

$$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) + h_{Jj}(t)(\boldsymbol{x}_i - \boldsymbol{w}_j(t)),$$

and here we need to introduce some notation and definitions.

Firstly, $h_{Jj}(t)$ is a neighbourhood function and $\Omega_J(t)$ is the set of indices of the neurons that form a neighbourhood around $\boldsymbol{w}_J$, *i.e.* that lie close to $\boldsymbol{w}_J$. As pointed out later, the neighbourhood will decrease in size as the iterations of the different algorithms continue. This means the number of points seen as close to $\boldsymbol{w}_J$ will decrease.

A simple possibility for specifying $h_{Jj}(t)$ would be:

$$h_{Jj}(t) = \begin{cases} \eta(t), & \text{if } j \in \Omega_J(t) \\ 0, & \text{if } j \notin \Omega_J(t). \end{cases}$$

Here $\eta(t)$ is the learning rate function which ensures that the algorithm benefits from slow learning and does not converge too quickly. The learning rate in the above equation is time-dependent, but it can also be specified as constant, $\eta(t) = \eta$. Now if we choose to specify $\Omega_J(t) = \{J\}$, this means that a neighbourhood around $\boldsymbol{w}_J$ only includes $\boldsymbol{w}_J$ itself and the above equation simplifies to:

$$h_{Jj}(t) = \begin{cases} \eta(t), & \text{if } j = J \\ 0, & \text{if } j \neq J. \end{cases}$$

This will cause the updating equation to simplify to:

$$\boldsymbol{w}_J(t+1) = \boldsymbol{w}_J(t) + \eta(t)(\boldsymbol{x}_i - \boldsymbol{w}_J(t)).$$

This is called Hard Competitive Learning and can also be referred to as Winner Takes All (WTA). If a neighbourhood of neurons around $\boldsymbol{w}_J$ is adjusted towards $\boldsymbol{x}_i$ it is called a Winner Takes Most (WTM) method or Soft Competitive Learning (Hastie *et al.*, 2009:528-529; Xu & Wunsch II, 2009:113,139).

We refrain from using the update equation $\boldsymbol{w}_J(t+1) = \boldsymbol{w}_J(t) + \eta(t)\boldsymbol{x}_i$, which could lead to exponential growth of the neurons, without an upper limit. With Hard Competitive Learning we run the risk of leaving certain neurons untouched as they might never have

the largest activation value. With Soft Competitive Learning and the introduction of the neighbourhood function, $h_{Jj}(t)$, we alleviate this problem by updating a whole group of neurons and, hopefully, improving all the neurons in the neighbourhood's probability of being the next BMU (Xu & Wunsch II, 2009:113).

3.3.2 Details about SOMs

The SOM was suggested as an unsupervised method for dimension reduction and clustering. It was also originally suggested as an online method for illustrative purposes. An online learning method is presented the training data observations in a sequential order and the outcome is updated for each observation. Batch learning methods are presented the full training dataset in one iteration of the learning algorithm. We will first discuss the online SOM and the batch SOM follows in a later section.

If class labels are known beforehand a supervised or semi-supervised SOM can be fitted, although this will not be discussed further in this study. The SOM provides a way for a high dimensional dataset to be mapped onto a lower dimensional grid of prototypes. The most popular choice for the dimensionality of the SOM grid is two dimensions, even though just one dimension or dimensions larger than two are also possible. We will only discuss two-dimensional SOMs in this study and the two-dimensional SOM grid is illustrated in Figure 10.

The grid of prototypes captures as much of the information in the full dataset as possible and tries to preserve the topology of the input space. This means that observations that lie close together in the input space should lie close together in the prototype grid. For this purpose the input space may only be stretched and bent, it is not allowed to cut or tear the space to fit the input observations onto this grid. This provides a way of visualising and understanding the proximities between the observations of the high-dimensional dataset in a two dimensional plane. For lower dimensional and smaller datasets the SOM might not be the best option and one of the methods designed for small sets should rather be used (Hastie *et al.*, 2009:528; Kohonen, 2001:105-106; Kohonen, 2013:53,55-56; Xu & Wunsch II, 2009:138).

To illustrate the various aspects of the SOM a toy dataset generated from a Multivariate Gaussian distribution is considered. In this dataset there are $p = 4$ variables and three distinct classes of observations, $K = 3$. The dataset is illustrated in Figure 9, plotted on the first two PCs which explained 87.1% of the variance in the data.

Each prototype can be labelled according to its position in the map. We can see the two dimensional map as an axis system and each prototype can receive a coordinate pair that references its location in the map. There will be M coordinate pairs with the horizontal axis called q_1 and the vertical axis called q_2. We count the rows from the bottom upwards and the columns from left to right (Hastie *et al.*, 2009:528). For example, the first prototype in the left bottom corner of the grid will be labelled $\boldsymbol{r}_1 = (q_{1,1}, q_{2,1})$. The position label $\boldsymbol{r}_1$ refers to the coordinate pair of the position of the prototype $\boldsymbol{w}_1$.

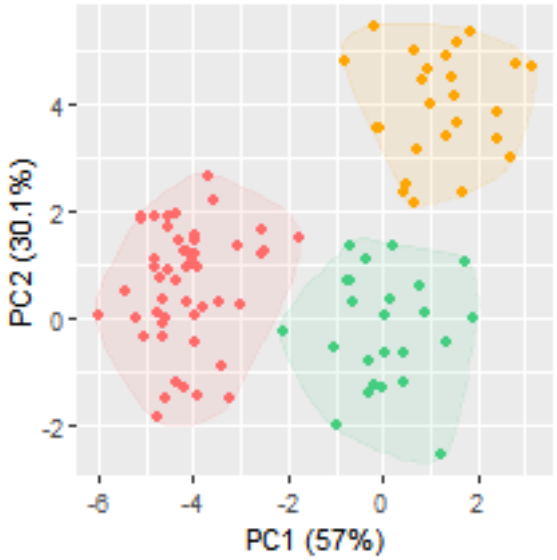


Figure 9: Multivariate Gaussian data, $p = 4$ and $K = 3$

The concepts of a neighbourhood and "closeness" can be defined within this coordinate system and Euclidean distance can be used for this purpose. A neighbourhood around the prototype $\boldsymbol{w}_j$ will include the prototype itself and the prototypes that lie close to it in the two dimensional coordinate plane, similarly all the $\boldsymbol{r}_i$ that lie close to $\boldsymbol{r}_j$ will be included in the neighbourhood. It is important to note that the neighbourhood in a SOM is determined in the two dimensional grid and not in the original high-dimensional input space and that distance is calculated in $\mathbb{R}^2$, not in $\mathbb{R}^p$ (Hastie *et al.*, 2009:528-529; Kohonen, 2001:111).

The prototypes can be represented on the SOM grid in different formats. The grid can be rectangular with the prototypes lining up horizontally and vertically. Another possibility is to have the prototypes present in a hexagonal way and the prototype grid will look like a honey comb. Kohonen (2013:55) recommends the use of the hexagonal map as this visualisation may be closer to the truth of how the prototypes align in the original input space. Observations and prototypes are unlikely to perfectly align to a rectangular grid in the input space. The algorithm may benefit from the extra flexibility to order the prototypes, because in a rectangular grid each prototype has four immediate neighbours and in a hexagonal grid that increases to six. There is also the possibility to organise the prototypes in an irregular fashion, for example to fit the shape of a geographical map, which can be useful in spatial and geographical clustering (Kohonen, 2001:110; Mayer, Merkl & Rauber, 2005).

Self Organising Map (SOM) Online Algorithm:

1. Initialise the prototype grid by deciding the grid dimensions and initialising the prototypes $\boldsymbol{w}_j \in \mathbb{R}^p$, $j = 1, 2, ..., M$.
2. Determine the learning rate function $\eta(t)$ and the neighbourhood function $h_{Jj}(t)$.
3. Choose an input observation $\boldsymbol{x}_i$ randomly.

4. Find the closest prototype $\boldsymbol{w}_J(t)$ to $\boldsymbol{x}_i$ in the input space $\mathbb{R}^p$ using Euclidean distance, $J = \underset{j}{\text{argmax}}\{\|\boldsymbol{x}_i - \boldsymbol{w}_j(t)\|\}$.

5. Establish the neighbourhood of prototypes $\Omega_J(t)$ around $\boldsymbol{w}_J(t)$ with $h_{Jj}(t)$ and calculate the current learning rate $\eta(t)$. The size of this neighbourhood (*i.e.* the number of prototypes it contains) decreases to 1 with t.

6. Update all $\boldsymbol{w}_j(t)$ in the neighbourhood of $\boldsymbol{w}_J(t)$ to move closer to input point $\boldsymbol{x}_i$ with

$$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) + h_{Jj}(t)(\boldsymbol{x}_i - \boldsymbol{w}_j(t)).$$

7. Iterate Step 3 to 6 until convergence, *i.e.* no significant change in the positions of the prototypes.

After the prototype grid has converged, new datasets can also be mapped onto the grid. Each prototype in the grid might represent a cluster by itself or might be grouped with other prototypes to form a larger cluster. The new observations then belong to the same clusters as their best matching prototypes; these best matching prototypes are determined using Euclidean distance (Hastie *et al.*, 2009:528-529; Kohonen, 2013:54,56; Xu & Wunsch II, 2009:139-141).

A requirement for the neighbourhood function and the learning rate is that they should both be monotonically decreasing with time. The learning rate update function can be defined as $\eta(t+1) = \alpha\eta(t)$. Here the value for $0 < \alpha < 1$ will depend on the speed of learning required, but should be chosen close to 1 for slow learning. We would like $\eta(t) \to 0$ as $t \to \infty$. This will be achieved by choosing $\eta(0)$ very large and having many steps in the algorithm (> 1000) (Alahakoon *et al.*, 2000:606).

The neighbourhood function should start very wide and grow smaller until the algorithm turns into an online K-Means algorithm and only updates one prototype at a time. This is a progression from Soft Competitive Learning to Hard Competitive Learning in slow and small steps. A good choice for the neighbourhood function is to multiply the learning rate with a radial basis function kernel that has the form:

$$h_{Jj}(t) = \eta(t) \exp\left(\frac{-\|\boldsymbol{r}_J - \boldsymbol{r}_j\|^2}{2\sigma^2(t)}\right), \text{ for all } j.$$

The function $\sigma(t)$ has to be monotonically decreasing with time as we would like the neighbourhood to shrink with time (Kohonen, 2001:111-112).

A suggested function for $\sigma(t)$ by Xu & Wunsch II (2009:140) is

$$\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau}\right),$$

where σ_0 is a large initial value and τ is a constant. Kohonen (2001:111; 2013:56-57) believes that specifying the form of the function $\sigma(t)$ is not too important to the SOM algorithm.

The requirement for the function is that it starts out very large (half of the diameter of the whole map) and reduces to a very small size (one or two prototypes) as t progresses. The importance of carefully specifying the parameters increases with the prototype grid size, if the grid is relatively small (for example, less than 100 nodes) then it would be sufficient to choose the parameters in a very simple fashion. The size of the initial neighbourhood would depend on the initialisation method for the SOM grid. If the prototypes are relatively ordered, as with linear initialisation discussed below, then the algorithm can go straight to the convergence phase. If the prototypes are not ordered at initialisation, as with random initialisation also discussed below, and the starting neighbourhood is chosen too small the map might not reach convergence. Convergence represents the point where the prototype vectors no longer change considerably from one iteration to the next.

3.3.3 Initialisation of the SOM

The first part of initialising the prototype grid is deciding on the number of prototypes and how they will be organised. The number of prototypes in a two dimensional grid can range from four to thousands. It will depend on the specific data problem and may also include some prior knowledge about the data or the number of clusters expected. It is recommended, as with most statistical learning models, that a few different grid sizes are tested on the data and an appropriate size is identified.

The height and width of the SOM can be determined by using principal component analysis (PCA). We can find the first two PCs and the eigenvalues linked to them and then create a ratio of the largest eigenvalue of the data to the second largest eigenvalue. Using this ratio as a rule of thumb should help define appropriate dimensions for the prototype grid by determining the height over width ratio of the SOM in the same way. For the toy dataset this ratio was 1.89, which means we can initialise the prototype grid on the ratio $2:1$. For example, if we want the width to be 3, we have to make the height 6. An example of this, together with the two types of SOM topologies, can be seen in Figure 10. The pie charts inside the prototype vectors represent the weight that is given to each variable in that prototype. These value come from a quick SOM fit with a Gaussian neighbourhood and random initialisation, for illustrative purposes.

SOMs, unfortunately, suffer from border effects as data are more sparse around the edges of the map and the spacing of the prototypes might be less reliable. This is a phenomenon that is witnessed in almost all statistical methods as it is harder to extrapolate a model around the edges of the data than interpolate in the interior of the dataset. In SOMs it has been suggested that the border effect can be alleviated by using spherical maps. Even though this sounds promising it will not be investigated further in this study. Growing SOMs, discussed in detail later, circumvent the problem of deciding on the size of the prototype grid as the method starts with a 2×2 grid and grows additional nodes as they are needed. This is a major improvement over the trial-and-error method that would have to be used for deciding the size of the prototype grid of the original SOM (Alahakoon, Halgamuge & Srinivasan, 2000:602; Kohonen, 2001:142; Kohonen, 2013:55-56).

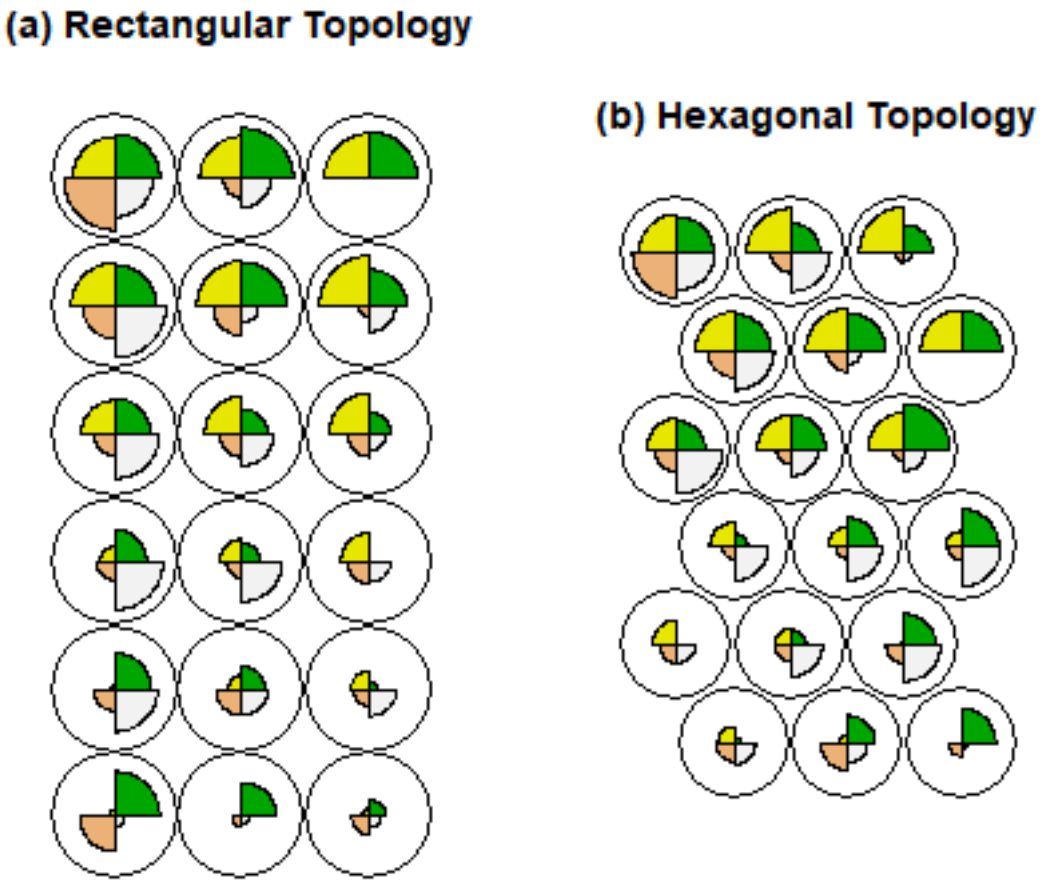


Figure 10: Examples of the two types of SOM topologies, (a) Rectangular and (b) Hexagonal, with dimensions according to the ratio of the first two eigenvalues and the pie charts representing the weights of the prototype vectors

The second part of initialising the prototype grid focuses on determining the initial weight vectors themselves. A first option, called random initialisation, is to assign observations randomly to the prototype vectors. This might cause the algorithm to take longer to stabilise as there is no pattern in the starting weight vectors. Initialising the SOM at random was initially done for illustrative purposes and simplicity when the SOM was developed.

It is also possible to initialise the map in such a way that there is already some order amongst the prototypes. This means that the algorithm does not have to use valuable time to create a sensible order in the prototypes. A way to do this is by using the first two principal components (PCs) of the input dataset, this is called linear initialisation. Akinduko, Mirkes and Gorban in 2016:220-221 found that there is merit in both types of initialisation. They found that the optimal initialisation method is dependent on the linearity of the data presented to the algorithm. For linear and semi-linear datasets it is more beneficial to use linear initialisation. Nonlinear data benefits from random initialisation, as many different starting values can be tested.

We now look at linear initialisation in more detail. A SOM algorithm should give the same results for the same PC initialisation, which means that it is completely reproducible. Random initialisation will only be reproducible if a seed has been set. The prototype vectors are then set as linear combinations of the first two PCs of the input data. According to the toy data example, the first two PCs are PC_1 = (-0.58, -0.81, -0.04, -0.02) and PC_2 = (0.81, -0.59, 0.05, 0). If we would like to initialise a grid of size 6×3 then we could create two weight vectors, which are called $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ for the PC-values, ranging between $[-1, 1]$. As an example of how $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ can be specified we set $\boldsymbol{a}_1$ = (-1, -0.6, -0.2, 0.2, 0.6, 1) and $\boldsymbol{a}_2$ = (-1, 0, 1). To initialise the prototype vector with location $\boldsymbol{r}_1$ and with coordinates $(1, 1)$ the values would be $\boldsymbol{w}_1(0) = \boldsymbol{a}_{1,1} \times \text{PC}_1 + \boldsymbol{a}_{2,1} \times \text{PC}_2 = (-1) \times \text{PC}_1 + (-1) \times \text{PC}_2$ = (-0.23, 1.4, -0.01, 0.02). A second example is the prototype vector with location $\boldsymbol{r}_5$ and with coordinates $(2, 2)$ the values would be $\boldsymbol{w}_6(0) = (-0.6) \times \text{PC}_1 + 0 \times \text{PC}_2$ = (0.35, 0.49, 0.02, 0.01).

According to Kohonen in 2001:142-143, when linear initialisation is used, the prototypes are already ordered and the first part of the SOM algorithm can be bypassed. The first 100 to 1000 iterations of the SOM algorithm are seen as the ordering phase where the learning rate is chosen initially large, as well as the neighbourhood. When linear initialisation is used the convergence phase can be started with a relatively small learning rate and neighbourhood, meaning that the ordering of the prototypes is basically done and fine-tuning can begin. Linear initialisation has some downfalls as there might be some empty prototypes at the end of the algorithm. This happens because the distribution of the input data over the input space is not taken into account in the SOM algorithm, meaning that the density of the points in the input space might be misrepresented in the final map (Akinduko *et al.*, 2016:216, 220-221; Kohonen, 2001:142; Kohonen, 2013:57).

3.3.4 The Batch SOM

SOMs were initially designed as an online method and was theoretically proven as such. However, Teuvo Kohonen in his 2013 paper suggests that the batch SOM is much more

appropriate for practical use. There are fewer parameters to specify, for example the learning rate falls away. The algorithm also converges faster than the online algorithm (Kohonen, 2013:53,57). It is interesting that in other sources the batch SOM is not explored in such detail and not always recommended above the online SOM. The algorithm for the batch SOM can be found below.

Self Organising Map (SOM) Batch Algorithm:

1. Initialise the prototype grid by deciding the grid dimensions and initialising the prototypes $\boldsymbol{w}_j \in \mathbb{R}^p$, $j = 1, 2, ..., M$.
2. Determine the neighbourhood function $h_{Jj}(t)$.
3. Supply each prototype vector with an empty sublist, C_j.
4. Choose an input observation $\boldsymbol{x}_i$.
5. Find the closest prototype $\boldsymbol{w}_J(t)$ to $\boldsymbol{x}_i$ in the input space $\mathbb{R}^p$ using Euclidean distance, $J = \underset{j}{\text{argmax}}\{\|\boldsymbol{x}_i - \boldsymbol{w}_j(t)\|\}$.
6. Add the input observation $\boldsymbol{x}_i$ to $\boldsymbol{w}_J(t)$'s sublist, C_J.
7. Repeat Step 4 to 6 until all inputs have been presented to the prototype grid, this represents one epoch.
8. Establish the neighbourhood of prototypes $\Omega_J(t)$ around each $\boldsymbol{w}_J(t)$ with $h_{Jj}(t)$.
9. Update each $\boldsymbol{w}_J(t)$ with the weighted mean vector of input observations in C_J:

$$\boldsymbol{w}_J(t+1) = \frac{\sum_{m=1}^{M} n_m(t) h_{Jm}(t) \bar{\boldsymbol{x}}_m}{\sum_{m=1}^{M} n_m(t) h_{Jm}(t)},$$

   where $n_m(t)$ is the number of observations in the sublist C_m for prototype m.
10. Iterate Step 3 to 9 until convergence.

In Step 9 $\bar{\boldsymbol{x}}_m = \frac{1}{n_m(t)} \sum_{i \in C_m(t)} \boldsymbol{x}_i$, which is the mean of all the input vectors in the sublist of prototype m. If the neighbourhood function would simply award a weight of 1 to prototypes in the neighbourhood around each $\boldsymbol{w}_J$ and 0 to the others, the update function in Step 9 would simplify to $\boldsymbol{w}_J(t+1) = \frac{\sum_{i \in C_{\Omega_J}(t)} n_i(t) \bar{\boldsymbol{x}}_i}{\sum_{i \in C_{\Omega_J}(t)} n_i(t)}$. Here $C_{\Omega_J}(t) = \{C_j\}$ and $j \in \Omega_J(t)$ is a union of the sublists from the prototypes in the neighbourhood. Convergence again represents the point where the prototype vectors no longer change considerably from one iteration to the next (Akinduko *et al.*, 2016:214-215; Kohonen, 2001:138-140; Kohonen, 2013:57-58).

The batch SOM is motivated by the fact that the online SOM converges to a stable state after enough iterations. This implies that

$$\lim_{t \to \infty} \mathbb{E}\{\boldsymbol{w}_j(t+1)\} = \lim_{t \to \infty} \mathbb{E}\{\boldsymbol{w}_j(t)\}$$

and

$$\mathbb{E}\{h_{Jj}(t)(\boldsymbol{x}_i - \boldsymbol{w}_j(t))\} = 0, \ \forall j.$$

These expected values are taken with respect to the underlying distribution generating the data. This means that in the limit we expect the values of the prototype vectors to stop changing from one iteration to the next. Let us define $\boldsymbol{x}(t)$ as the observation chosen at iteration t. We can now let $t \to \infty$ and approximate the expected value by the mean of all the time states to return:

$$\begin{aligned}
&\frac{1}{t}\sum_{t=1}^{\infty}\{h_{Jj}(t)(\boldsymbol{x}(t) - \boldsymbol{w}_j^*)\} = 0 \\
&\Rightarrow \sum_{t=1}^{\infty}\{h_{Jj}(t)(\boldsymbol{x}(t) - \boldsymbol{w}_j^*)\} = 0 \\
&\Rightarrow \sum_{t=1}^{\infty} h_{Jj}(t)\boldsymbol{x}(t) - \boldsymbol{w}_j^* \sum_{t=1}^{\infty} h_{Jj}(t) = 0 \\
&\Rightarrow \sum_{t=1}^{\infty} h_{Jj}(t)\boldsymbol{x}(t) = \boldsymbol{w}_j^* \sum_{t=1}^{\infty} h_{Jj}(t) \\
&\Rightarrow \boldsymbol{w}_j^* = \frac{\sum_{t=1}^{\infty} h_{Jj}(t)\boldsymbol{x}(t)}{\sum_{t=1}^{\infty} h_{Jj}(t)},
\end{aligned}$$

where $\boldsymbol{w}_j^*$ represents a stabilised prototype vector that does not change with increments of time. Similarly to the online SOM, the batch SOM reduces to a K-Means algorithm when the neighbourhood function has become small enough to only include the BMU (Kohonen, 2001:138-140; Kohonen, 2013:57). Both the batch SOM and the online SOM will be investigated in the Simulation Study that will follow.

3.3.5 Combining SOMs and other clustering methods

SOMs can in itself be used as a clustering method, but it can also be used as a dimension reduction technique before a second round of clustering. Dimension reduction referred to in this discussion is based on the fact that clustering the prototypes is based on the two-dimensional structure of the SOM. Instead of having to cluster points in p dimensions, the problem reduces to clustering prototypes in two dimensions.

This would entail specifying the prototype grid fairly large, such that $M >> K$. The prototype vectors are then used as the input vectors to a second clustering. Each observation in the input data is then assigned to the cluster its prototype vector belongs to. Now that the dimensionality of the data has been reduced to only two dimensions, any of the classic clustering methods (that break down in high dimensions or with large datasets) can be used. Classic clustering methods refer to hierarchical clustering and K-means, for example, or any of the clustering methods discussed.

The grid of prototypes also contain much less noise than the original dataset; each prototype is a mean-like summary of the data points assigned to it and thus leads to a reduction in variation in the data. When using the prototypes as input to a clustering algorithm it is

recommended to exclude any prototypes that did not get the opportunity to be the BMU (Vesanto, & Alhoniemi, 2000:586,588,592; Xu & Wunsch II, 2009:138). This two-tiered data clustering approach will be investigated in the Simulation Study and also the clustering methods that fit optimally on top of a SOM.

3.3.6 Growing Self-organizing Maps (GSOMs) and Growing Hierarchical Self-organizing Maps (GHSOMs)

As mentioned before, GSOMs are SOMs with an initial prototype grid of size 2×2 and it grows additional nodes as they are needed. This means that the map can expand into certain areas of the data where more prototypes are needed and sparse areas of the data can be served by only one or two prototypes. This also alleviates the problem that is found in SOMs, that the data structure has to be anticipated in the initialisation of the prototype grid and also in deciding on an appropriate number of iterations for the algorithm to converge. There is also the possibility to impose a hierarchical structure on the GSOM. This would mean that the GSOM is grown in different phases. The first phase grows the initial SOM and the subsequent phases, in essence, 'zoom in' on certain parts of the map. At each phase a certain part of the GSOM map is fed into a second GSOM algorithm. The advantage here is to learn more about the regions in the data that contain more information than others. Regions that provide little information may indicate noise or outliers (Alahakoon *et al.*, 2000:602-603; Kohonen, 2013:56).

The algorithm for GSOM is given below. It is important to note that we keep a counter at each of the prototypes, called the total error, TE_i. Each time the specific prototype is the BMU, the difference between the input node and the BMU (before updating) is added to the total error of the prototype. If this TE_i value exceeds the growth threshold, GT, then the opportunity exists to grow new nodes.

Growing Self Organising Map (GSOM) Algorithm:

1. Initialise the prototypes $\boldsymbol{w}_j \in \mathbb{R}^p$ and set $\text{TE}_j = 0$, for $j = 1, 2, 3, 4$.
2. Calculate the initial growth threshold (GT), determine the learning rate function $\eta(t)$ and the neighbourhood function $h_{Jj}(t)$.
3. Choose an input observation $\boldsymbol{x}_i$ randomly, without replacement. This is repeated until one epoch is complete, then all of the observations are put back in the sample.
4. Find the closest prototype $\boldsymbol{w}_J(t)$ to $\boldsymbol{x}_i$ in the input space $\mathbb{R}^p$ using Euclidean distance, $J = \underset{j}{\text{argmax}}\{\|\boldsymbol{x}_i - \boldsymbol{w}_j(t)\|\}$.
5. Establish the neighbourhood of prototypes $\Omega_J(t)$ around $\boldsymbol{w}_J(t)$ with $h_{Jj}(t)$ and calculate the current learning rate $\eta(t)$.
6. Update all $\boldsymbol{w}_j(t)$ in the neighbourhood of $\boldsymbol{w}_J(t)$ to move closer to input point $\boldsymbol{x}_i$ with
$$\boldsymbol{w}_j(t+1) = \boldsymbol{w}_j(t) + h_{Jj}(t)(\boldsymbol{x}_i - \boldsymbol{w}_j(t))$$

7. Calculate the error $\text{err}_J(t) = \|\boldsymbol{x}_i - \boldsymbol{w}_J(t)\|^2$.
8. Update the total error for prototype $\boldsymbol{w}_J(t)$: $\text{TE}_J = \text{E}_J(t+1) = \text{E}_J(t) + \text{err}_J(t)$.
9. If $\text{TE}_J > \text{GT}$ and
   a) $\boldsymbol{w}_J(t)$ is a boundary node, then grow new nodes and initialise them as discussed below.
   b) $\boldsymbol{w}_J(t)$ is an interior node, then distribute the error to neighbouring prototypes - see later for full description.
10. Reset the learning rate function $\eta(t)$ to the starting value.
11. Iterate Step 3 to 10 so that at least one epoch has been completed and node growth is at a minimum (convergence has occurred).
12. Set a small starting value for the learning rate function $\eta(t)$ and a small starting neighbourhood width for $h_{Jj}(t)$.
13. Continue only running the SOM algorithm until the algorithm has converged.

The chosen method of initialisation for the prototype grid in Step 1 is random initialisation and the grid shape is specified as rectangular. The error calculated in Step 7 can also be calculated with different distance measures, but squared Euclidean distance was chosen here (Alahakoon *et al.*, 2000:603-604).

Extra nodes can only be grown from boundary nodes and they are added in all possible growing directions. This means that a corner node would grow two extra nodes, one vertical and one horizontal; a boundary node that is not a corner node will only grow one extra node to the open side. Choosing the initial shape of the prototype grid as 2×2 allows the algorithm many options as to which direction to expand into. Growing nodes in all of the directions available to a border node may cause some unnecessary nodes that will never be the BMU. These unnecessary nodes can again be removed in a 'clean up' phase at the end of the algorithm or can be used to clearly indicated cluster borders in the prototype grid. When the prototype grid is too small there will be too many input observations assigned to one of the prototypes and it will drive up that specific node's TE_J. This indicates that that specific area of the input space needs more attention and node growth is necessary in that direction (Alahakoon *et al.*, 2000:604-605).

In GSOMs it makes sense to have the neighbourhood function start much smaller than for the original SOM, as the grid itself is very small. Mostly the neighbourhood function has the form

$$h_{Jj}(t) = \begin{cases} \eta(t), & \text{if } j \in \Omega_J(t) \\ 0, & \text{if } j \notin \Omega_J(t). \end{cases}$$

The learning rate will be dependent on the number of nodes in the current iteration of the GSOM. It makes sense to start with a small value for $\eta(t)$, because of the small grid, but as the grid grows the learning rate might have to inflate before it can decrease again. This logic allows us to introduce the update function for the learning rate as $\eta(t+1) = \alpha \times \psi(m(t)) \times \eta(t)$.

Here $0 < \alpha < 1$ is the parameter that controls the size of the learning rate reduction from one iteration of the algorithm to the next. The function $\psi(m(t))$ is dependent on the current number of nodes in the prototype grid, $m(t)$, and a good choice for this function is $\psi(m(t)) = (1 - \frac{R}{m(t)})$. Here R is an arbitrary constant but can be something similar to the initial number of prototypes in the grid, for example 4 (Alahakoon *et al.*, 2000:603,606).

When new nodes are added to the GSOM it is important to pay attention to the weight initialisation. When a new node is added between two existing nodes, it is acceptable to take the average of the two neighbouring nodes as the initialisation value. Mostly, newly grown nodes will only have one neighbour from the existing nodes. The node where the growth originates from will have neighbours of its own, and these will have to be used to help initialise the new node.

Let us call the node where growth originates from $\boldsymbol{w}_A$ and its neighbour $\boldsymbol{w}_B$. If $\|\boldsymbol{w}_B\| > \|\boldsymbol{w}_A\|$ then the new weight vector will be $\boldsymbol{w}_A - (\boldsymbol{w}_B - \boldsymbol{w}_A)$ and if $\|\boldsymbol{w}_B\| < \|\boldsymbol{w}_A\|$ then the new weight vector will be $\boldsymbol{w}_A + (\boldsymbol{w}_A - \boldsymbol{w}_B)$, where $\|.\|$ is the Euclidean norm of a vector. It is always preferable to choose $\boldsymbol{w}_B$ to form a straight line between $\boldsymbol{w}_A$ and the new node, but if that is not available any of the neighbours will be useful for the new weight initialisation. The formula stays the same, regardless of where the neighbour is attached. There is also the possibility, due to trimming of the prototype grid, that a new node may be grown from a prototype with no other neighbours. Initialisation in this case is just the average between the largest and smallest prototype in the grid. Here largest and smallest refers to the weight vectors with the largest and smallest Euclidean norms, respectively. The newly assigned weight values might be weighted down or up to make sure that they fit in with the organisation of the existing grid (Alahakoon *et al.*, 2000:605).

We also have to discuss the distribution of the error to other nodes in the prototype map if an interior node's TE was larger than the growth threshold, GT. Because the node where the growth is needed is in the interior of the grid the error of nodes around it must be increased. As the algorithm evolves, this increasing of the error of the neighbours of the winning node will eventually reach a border node. This process allows interior BMUs to also have an influence on the size of the final grid. The error of the BMU itself will be updated by $\mathrm{E}_J(t+1) = \frac{\mathrm{GT}}{2}$. All the bordering nodes to $\boldsymbol{w}_J(t)$ will have their errors updated by $\mathrm{E}_j(t+1) = \mathrm{E}_j(t) + \gamma \times \mathrm{E}_j(t)$, where $0 < \gamma < 1$ is called the factor of distribution. This can be decided in line with how fast the map is desired to grow (Alahakoon *et al.*, 2000:607).

The growth threshold, GT, also needs to be discussed. The growth threshold controls the spread of the prototype grid and its size can be dictated by the purpose of the GSOM. If the goal is for the map to be large and very detailed then a small GT value will be required. If a smaller map with less detail is required as a brief overview of the data, or as a starting point for further GSOMs on smaller parts of the map, then a large value of GT should be chosen. The number of variables in the input dataset should also be considered because the TE value at each node will become higher with higher dimensionality of the input data. This is because the Curse of Dimensionality starts moving observations further apart.

The spread factor, SF, is introduced here as a universal parameter that controls the spread of the GSOM and allows the GSOMs from different datasets with different dimensionalities

to be comparable. We define the equation $\text{GT} = -p \times \ln(\text{SF})$, where $0 \leq \text{SF} \leq 1$ and the derivation will not be discussed here but can be found in Alahakoon *et al.*, 2000:608-609. This SF has full control over the growth of a GSOM. It is recommended to use $0 \leq \text{SF} \leq 0.3$ in the exploratory phase of data analysis which should provide a good overview of the input dataset. Now a new GSOM can be grown with an $\text{SF} > 0.3$ if it is required or a certain area of the data can be chosen to 'zoom in' on (Alahakoon *et al.*, 2000:604,607-609).

An advantage of the GSOM is that the shape of the map itself already reveals the different possible clusters in the data. It will branch out into the different areas of interest and will be relatively easy to pick out and investigate further with larger SF values. The original GSOM can be grown with a certain SF, and once a subsection of the data is identified, a second GSOM can be grown with a different SF. This process can continue until a range of different maps have been grown with a range of different SF values. As long as each map was grown on a subsection of the previous map, a hierarchical structure of maps (GHSOM) will be created. This can reveal all the possible clusters in the data (Alahakoon *et al.*, 2000:610,613).

3.4 Summary

In this part of the literature review we have discussed methods to be used for large and high dimensional datasets. The methods discussed were CLARA, SOMs, GSOMs and GHSOMs. Only CLARA and SOMs can be tested and investigated in the following simulation study because R software for the other methods have not been developed.

CHAPTER 4: SIMULATION STUDY

4 Data Simulation Study

4.1 Introduction

The purpose of the simulation study is to illustrate the application of the different clustering methods discussed in the literature review. These include the more traditional methods of K-means, K-medoids and hierarchical clustering, where data samples are assumed to be small to medium sized and low-dimensional. We will also investigate methods that are more geared towards larger datasets such as CLARA, as well as SOM that can handle high dimensions.

By generating the datasets according to certain specifications, the environment is controlled. The data will be generated from a multivariate Gaussian distribution. All the groups will have the same covariance matrix, but the mean vectors will be varied to set the groups apart. The differences amongst the mean vectors will be varied in a systematic way. By this we mean that we will move the mean vectors of the groups closer together to ascertain how difficult the methods find it to separate the data into distinct clusters.

We will also use larger datasets with more clusters to test CLARA and SOMs. Datasets that were generated to have specific cluster shapes, *i.e.* not spherical, will be used to determine how the different methods handle different cluster shapes that will probably occur in real-world data. The methods investigated in this simulation study are inherently unsupervised learning methods. Investigating them in a supervised fashion, by specifying clusters in the data beforehand, is a way of determining their effectiveness in different scenarios.

In the simulation study the simulated data will first be specified and visualised. Next the various clustering methods will be discussed, along with the approriate R packages and functions to use for each method. The relevant R code and output will form part of the text to ensure that the study is reproducible and understandable. The clustering methods will also be compared using certain criteria discussed in the literature review.

4.2 Design of the Datasets

We first load all the packages that will be needed. **MASS** is used when generating multivariate Gaussian data (Venables & Ripley, 2002), while **ggplot2** is used for visualisations (Wickham, 2016), along with **grid**, **gridBase** and **gridExtra** to arrange more than one plot in a grid (R Core Team, 2018; Murrell, 2014; Auguie, 2017). The package **xtable** and the function with the same name are very useful for creating Latex code for tables from R output (Dahl, 2016). The R package **knitr** is used for importing images into the final pdf (Xie, 2018). The necessary further explanations will be provided as the packages are being called.

```
library(MASS)
library(xtable)
```

```
library(ggplot2)
library(ggalt)
library(grid)
library(gridBase)
library(gridExtra)
library(knitr)
library(cluster)
library(factoextra)
library(dendextend)
library(clv)
library(kohonen)
library(SOMbrero)
```

In Table 4 a summary of the data scenarios that were chosen for this simulation study can be seen. The details that are available per scenario are the number of variables, the number of clusters, the number of observations, the shapes of the clusters (spherical, elongated or mixed) and the cluster size balance. The scenarios will be discussed in detail throughout the study.

The variables have a multivariate Gaussian distribution with varying mean vectors for different cluster labels. We specify the covariance matrix as being the same for all the clusters. A very small correlation between the variables is added. Below is the function to create the covariance matrix and the desired output from the function. This covariance matrix will be used for all data generation.

```
CovMatFunc <- function(p, corx, sdx){
  #p: number of variables
  #corx: specified pairwise correlation value (same for all pairs)
  #sdx: specified standard deviation per variable (same for all variables)

  CovMat <- matrix((sdx^2)*corx, nrow = p, ncol = p)
  diag(CovMat) <- (sdx)^2
  return(CovMat)
}
```

$$\Sigma = \begin{bmatrix} 1 & 0.15 & 0.15 & \dots & 0.15 \\ 0.15 & 1 & 0.15 & \dots & 0.15 \\ 0.15 & 0.15 & 1 & & 0.15 \\ \vdots & \vdots & & \ddots & 0.15 \\ 0.15 & 0.15 & 0.15 & \dots & 1 \end{bmatrix}.$$

The function that assembles all of the datasets can be seen in Appendix C. This function, called *GenXDat*, can generate spherical and elongated clusters. The μ-vectors will determine how well separated the clusters are.

Table 4: Scenario A and B overview

Scenario	**A**	**B**
$\boldsymbol{p}$	10	10
$\boldsymbol{K}$	3	11
$\boldsymbol{N}$	600	15000
Shapes	Spherical	Mix
Balance	Balanced	Imbalanced

4.2.1 Datasets for Scenarios A and B

Scenario A is purposefully designed to be simple and to be used for illustration of all the traditional clustering techniques. These include K-means, K-medoids and hierarchical clustering. This scenario has compact and spherical clusters lying fairly close together, but with minimal overlap. The idea is that all the traditional methods should perform well on this dataset. Scenario B is specifically designed to have a large number of observations and to have more clusters than the previous scenario. The aim of this is to play to the strengths of a method such as SOMs, that can handle high-dimensional data and large datasets. It is designed to be more complex with elongated clusters. This is to test the effectiveness of SOMs and also to investigate how well the traditional clustering methods fit on top of SOMs.

For simplicity, the number of clusters is chosen as three, $K = 3$, for Scenario A. Initially we will also keep the clusters of the same size, $N_1 = N_2 = N_3 = 200$ and $N_A = 600$ for Scenario A. Scenario B will have eleven clusters so that $K = 11$. Also, $N_1 = N_2 = N_{11} = 2000$, $N_3 = N_4 = N_9 = N_{10} = 1500$, $N_6 = N_8 = 1000$, $N_5 = N_7 = 500$ and $N_B = 15000$. We would like to investigate these scenarios in 10-dimensional space, *i.e.* $p = 10$. The mean vectors for Scenarios A and B will be varied to observe how the results change as the generated clusters move further away from each other, *i.e.* distances among mean vectors increase. "Further away" in this case refers to distance in Euclidean space. Scenario B will test how the clustering methods react to elongated clusters. For Scenario B three elongated clusters will be included.

The first μ-vector for Cluster 1 will be kept fixed at the origin, for both scenarios, as a reference point:

$$\boldsymbol{\mu}_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

The two μ-vectors compared to the origin for Scenario A are:

$$\boldsymbol{\mu}_2^{\mathrm{A}} = \begin{bmatrix} 6 & 6 & 6 & 6 & 6 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$\boldsymbol{\mu}_3^{\mathrm{A}} = \begin{bmatrix} 3 & 3 & 3 & 3 & 3 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

We hope that specifying the cluster means in this way will result in three clusters that are simple to separate. These three clusters serve an illustrative purpose and should be the easier of the two scenarios.

The ten μ-vectors compared to the origin for Scenario B are:

$$\begin{aligned}
\boldsymbol{\mu}_2^{\mathrm{B}} &= \begin{bmatrix} 12 & 12 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\boldsymbol{\mu}_3^{\mathrm{B}} &= \begin{bmatrix} 11 & 11 & 11 & 11 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\boldsymbol{\mu}_4^{\mathrm{B}} &= \begin{bmatrix} 10 & 10 & 10 & 10 & 10 & 10 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\boldsymbol{\mu}_5^{\mathrm{B}} &= \begin{bmatrix} 9 & 9 & 9 & 9 & 9 & 9 & 9 & 9 & 0 & 0 \end{bmatrix} \\
\boldsymbol{\mu}_6^{\mathrm{B}} &= \begin{bmatrix} 8 & 8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\boldsymbol{\mu}_7^{\mathrm{B}} &= \begin{bmatrix} 7 & 7 & 7 & 7 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\boldsymbol{\mu}_8^{\mathrm{B}} &= \begin{bmatrix} 6 & 6 & 6 & 6 & 6 & 6 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\boldsymbol{\mu}_9^{\mathrm{B}} &= \begin{bmatrix} 5 & 5 & 5 & 5 & 5 & 5 & 5 & 5 & 0 & 0 \end{bmatrix} \\
\boldsymbol{\mu}_{10}^{\mathrm{B}} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 4 & 4 & 4 & 4 & 4 \end{bmatrix} \\
\boldsymbol{\mu}_{11}^{\mathrm{B}} &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 3 & 3 & 3 & 3 & 3 \end{bmatrix}.
\end{aligned}$$

The mean vectors for Scenario B results in clusters of varying levels of separation. This means that we expect some of the clusters to be more difficult to separate than others. We expect the elongated clusters and the clusters with a high level of overlap to pose the largest difficulty to the clustering methods.

4.2.2 Visualising 10-dimensional data for Scenarios A and B

The generated data are more than three-dimensional and are therefore impossible to visualise without some kind of simplification or projection. Principal Component Analysis (PCA) will be used to find the first two principal components (PCs) of the data. The clusters can then be visualised as projections onto these two PCs. The function *prcomp* in the R **stats** package allows us to perform this PCA (R Core Team, 2018). The R code for determining the PCs for Scenario A and determining the percentage of variance explained by each PC can be found below. In the table below we also find the output of the percentage variance explained per PC for Scenarios A and B.

```
pca_A <- prcomp(XDatSet_A[,-11], center = FALSE, scale. = FALSE)
perc_explained_A <- (pca_A$sdev^2)/sum(pca_A$sdev^2)
```

```
xtable(perc_explained_AB)
```

% Var Explained	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
Scenario A	90.5	1.7	1.1	1.1	1.1	1.0	1.0	0.9	0.9	0.8
Scenario B	76.5	9.9	7.0	3.4	1.6	0.4	0.3	0.3	0.3	0.3

We can see in the table above that for Scenario A only the first one or possibly two PCs are significant. A clear, large drop in percentage variance explained can be seen after the

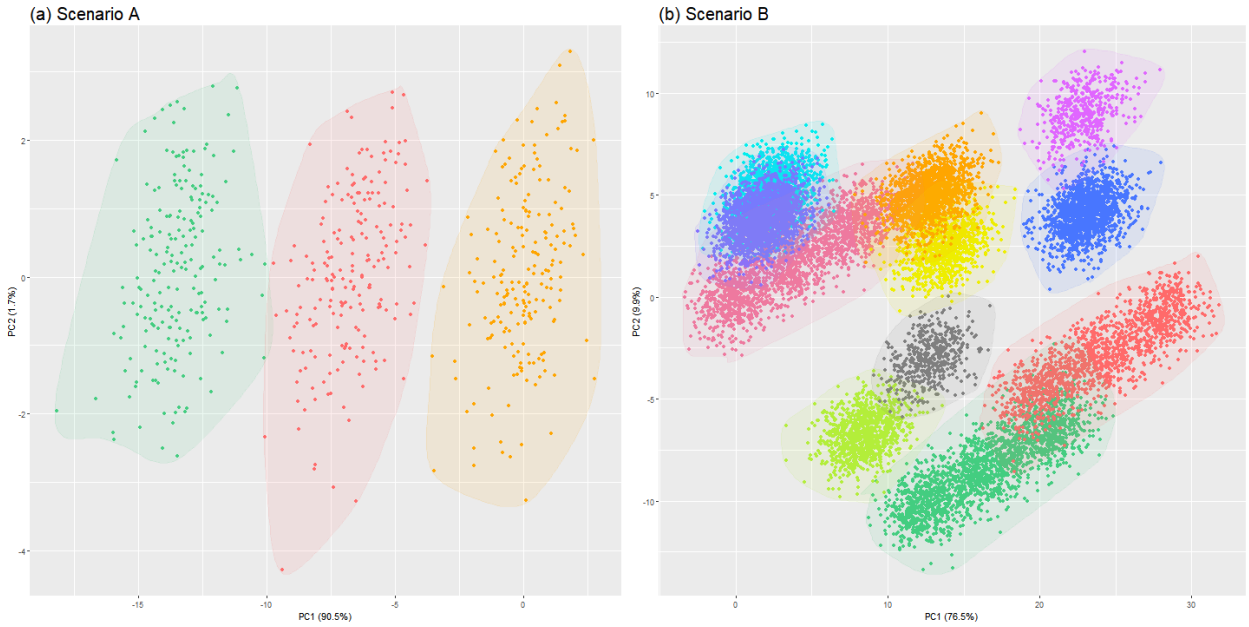


Figure 11: Data Scenarios A and B plotted in terms of their first two PCs with given class labels

1st PC and a much smaller drop after the 2nd PC. For Scenario B the first 5 PCs seem to be significant. A large drop in percentage variance explained can be seen after the 1st PC and much smaller drops after the next 4 PCs. Also, because we have specified the covariance matrix and we know that all the variances will be the same, we do not need to normalise the data. For a real-world dataset normalisation will be required so as to not favour variables with larger measurement units.

As we can see in Figure 11(a) the three classes are separable in Scenario A, and there is practically no overlap between the classes. Here the first two PCs explained 92.2% of the variance in the data. Figure 11(b) shows Scenario B where the first two PCs explained 86.4% of the variance in the data. In Figure 11(b) we can see the larger number of clusters of Scenario B. There is some overlap between some of the clusters, while others are well separated. The three elongated clusters can also be seen clearly.

The shaded areas in Figure 11 are created by the function *geom_encircle* from the package **ggalt** (Rudis, Bolker & Schulz, 2017). The R code to create Figure 11(a) is given below and the code is similar for the rest of the colour-coded cluster plots. The function *ggplot* only takes dataframes as input, so a dataframe needs to be created if the data is not already in the correct format.

```
dat_for_plot_A <- cbind(as.data.frame(pca_A$x[, 1:2]),
                        labels = as.factor(XDatSet_A[,11]))

plot_A <- ggplot(dat_for_plot_A, aes(x = PC1, y = PC2, colour = labels))+
  geom_point(show.legend = F)+
  ggtitle("(a) Scenario A")+
  theme(plot.title = element_text(size = 15))+
  xlab(paste("PC1 (", perc_explained_A[1], "%)", sep = ""))+
  ylab(paste("PC2 (", perc_explained_A[2], "%)", sep = ""))+
  scale_color_manual(breaks = c("1", "2", "3"),
                     values = c("orange", "seagreen3","indianred1"))+
  geom_encircle(data = subset(dat_for_plot_A, labels == 1),
                aes(x=PC1, y=PC2), fill = "orange",  colour = "orange",
                alpha = 0.1, s_shape = 0.5, expand= 0.005)+
  geom_encircle(data = subset(dat_for_plot_A, labels == 2),
                aes(x=PC1, y=PC2), fill = "seagreen3", colour = "seagreen3",
                alpha = 0.1, s_shape = 0.5, expand= 0.005)+
  geom_encircle(data = subset(dat_for_plot_A, labels == 3),
                aes(x=PC1, y=PC2), fill = "indianred1",
                colour = "indianred1", alpha = 0.1,
                s_shape = 0.5, expand= 0.005)
```

4.3 Cluster Validation

After we have applied the different clustering methods we would like to get an indication of the appropriateness of the clusters found and also their isolation and compactness. These concepts were discussed in the literature review. There is an R package called **clv** for this purpose. It was designed by Lukasz Nieweglowski and was last updated on 19 February 2015. This package contains calculations for some external criteria, for example the Rand Index, Jaccard Coefficient and Fowlkes and Mallows Index we discussed before. We also use this package for calculating relative criteria, such as the Dunn and Davies-Bouldin indices (Nieweglowski, 2015). We will also use the average silhouette value per clustering to compare the different clustering algorithms. This is done at the end of each of the scenario discussions.

Comments about performance can only be made because we have the luxury of knowing the true underlying partitioning of the data. The cluster labels were not used in the training of the clustering algorithms and the cluster labels will also not be available in a real-world dataset. There is no measure of accuracy available for clustering in an unsupervised environment. We can simply use the clusters to gain more insight into the kind of subgroups that might exist in our data.

All the conclusions made about the different clustering methods, and the R software for them, are very specific to the data that were generated here. If a different distribution is used or different parameters are chosen, then the conclusions might have been different. This also

holds for the conclusions about the methods for choosing K.

4.4 Scenario A

We will use Scenario A to explain all the R software needed for fitting different clustering methods. We will also show all visualisations available and how to create them. For the remaining scenarios only the interesting findings or outcomes will be shown.

4.4.1 *K*-means Clustering

The most popular R software associated with K-means comes from the **stats** package that is automatically loaded with the R environment (R Core Team, 2018). The function is called *kmeans*. Input parameters for this function include:

x: the data matrix X (in the format of a vector, matrix or dataframe)
centers: the number of clusters K
iter.max: the maximum number of iterations of the algorithm
nstart: number of random initialisations of the mean vectors
algorithm: can choose between Hartigan-Wong, Lloyd, Forgy or MacQueen (we will use the default, Hartigan-Wong, as it is almost certain to converge)

The most important outputs include:

cluster: the vector of cluster allocation indices, C
centers: the matrix of cluster centers, $\{\boldsymbol{m}_1, \boldsymbol{m}_2, ...\boldsymbol{m}_K\}$
totss: the total sum of within- and between-cluster variation, $T(C) = W(C) + B(C)$
withinss: a vector of the within-cluster variations
tot.withinss: total within-cluster variation, $W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k} d_{ii'}$
betweenss: total between-cluster variation, $B(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')\neq k} d_{ii'}$
size: number of points in each cluster
iter: number of iterations actually performed

We need to specify a value for K as the first step in the K-means algorithm. The data were generated, so we know it should be $K = 3$ for Scenario A. We would like to confirm that the gap statistic, elbow method and average silhouette method, explained in the literature review, find this value as well.

4.4.1.1 Choosing the value of *K*

We use the function *clusGap* from the package **cluster** to determine the values to calculate the gap statistic. Input parameters for this function include:

x: the data matrix X (in the format of a matrix or dataframe)
FUNcluster: the clustering algorithm to use
K.max: number of clusters to test, must be larger than one

B: number of Bootstrap samples to use for the null distribution
d.power: power applied to the Euclidean distance

The most important output is:

Tab: the output matrix with a row for every **K.max** value and columns logW, E.logW, gap, and SE.sim; gap = E.logW - logW; SE.sim = $\text{sd}_k\sqrt{1+\frac{1}{B}}$; sd_k is the standard error of the gap

This package was last updated on 9 April 2018 and is largely based on the methods described in the book *Finding Groups in Data* by Kaufman and Rousseeuw (Maechler, Rousseeuw, Struyf, Hubert & Hornik, 2018). We use this package several times in this study, for PAM, CLARA and divisive hierarchical clustering.

We also use the function *fviz_gap_stat* from the package **factoextra** for the visualisation of the gap statistic. Input parameters for this function include:

gap_stat: an object returned from the function *clusGap*
maxSE: a list of two parameters, *method* and *SE.factor*; *method*: we use Tibs2001SEmax as this was proposed in the paper by Tibshirani *et al.* in 2001; other options are globalmax, firstmax, firstSEmax; *SE.factor*: number of standard errors to consider

The output is a *ggplot* object. This package was last updated on 22 August 2017 (Kassambara & Mundt, 2017). The R code used to create the gap statistic and its plots can be seen below for Scenario A.

```
set.seed(159874)
gap_out_A <- clusGap(x = XDatSet_A[,-11], FUNcluster = kmeans, B = 500,
                     K.max = 10, nstart = 25, d.power = 2)
ClusFrame_A <- data.frame(cbind(gap_out_A$Tab[,1:2], K = c(1:10)))

Gap_1A <- ggplot(ClusFrame_A, aes(x = K, y = logW))+
  geom_point(color = "darkgreen")+ geom_line(color = "darkgreen")+
  geom_point(aes(x = K, y = E.logW), color = "maroon")+
  geom_line(aes(x = K, y = E.logW), color = "maroon")+
  scale_x_continuous(breaks = 1:10, labels = c("1", "2", "3", "4", "5", "6",
                                               "7", "8", "9", "10"))+
  ggtitle("(a)") + xlab("k") + ylab(expression(logW[k]))

Gap_2A <- fviz_gap_stat(gap_out_A, maxSE = list(method = "Tibs2001SEmax",
                                                SE.factor = 1))+
  ggtitle("(b)") + xlab("k") + ylab(expression(Gap[N](k)))
```

The function *fviz_nbclust* from the package **factoextra** is used for the visualisations of the elbow method and the average silhouette method. Input parameters for this function include:

x: the data matrix X (in the format of a matrix or dataframe)
FUNcluster: the clustering algorithm to use

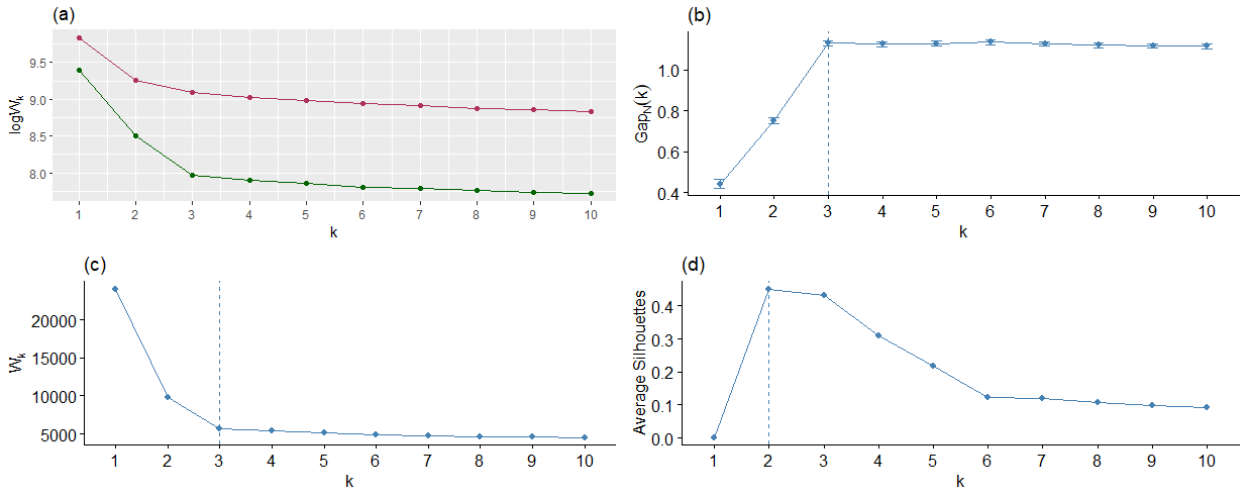


Figure 12: The gap statistic (a) and (b), the elbow method (c) and the silhouette method (d) for choosing K for Scenario A using K-means clustering

method: the optimal number of clusters selection method; can be "wss", "silhouette" or "gap_stat"
k.max: number of clusters to test, must be larger than one
nboot: number of Bootstrap samples to use for the null distribution; only needed when **method** = "gap_stat"
nstart: the number of random initialisations if **FUNcluster** = "kmeans"

This function also returns a *ggplot* object and it can be manipulated as such. The code for plotting the elbow method and the average silhouette method graphs can be seen below for Scenario A.

```
set.seed(159874)
Elbow_A <- fviz_nbclust(x = XDatSet_A[,-11], FUNcluster = kmeans,
                        method = "wss", k.max = 10, nstart = 25)+
  ggtitle("(c)") + xlab("k") + ylab(expression(W[k]))+
  geom_vline(xintercept = 3, linetype = 2, color = "steelblue")

set.seed(159874)
Silhouette_A <- fviz_nbclust(x = XDatSet_A[,-11], FUNcluster = kmeans,
                        method = "silhouette", k.max = 10, nstart = 25)+
  ggtitle("(d)") + ylab("Average Silhouettes") + xlab("k")

grid.arrange(Gap_1A, Gap_2A, Elbow_A, Silhouette_A, nrow = 2, ncol = 2)
```

Figure 12 shows the results when applying the gap statistic, the elbow method and the average silhouette method to Scenario A for K-means clustering. Figures 12(a) and (b) illustrate that the gap statistic chose the appropriate number of clusters for Scenario A, $K = 3$. The elbow method, that can be seen in Figure 12(c), chose the number of appropriate clusters to be 3, as there seems to be a "kink" in the plot at $K = 3$. A warning for the elbow

method is that the "kink" in the plot might be very subtle and other methods for the choice of the optimal number of clusters will have to be relied on. In Figure 12(d) we can see the average silhouette method for Scenario A. Here the number of clusters was chosen as $K = 2$. This is incorrect according to how the data were generated, but it hints at a problem that will be experienced in real-world data.

In real-world data we will not know the true underlying clustering structure in the data. If clusters are overlapping or superimposed, the analyst will not know this and has to trust the output from methods such as the gap statistic or the elbow method to suggest an appropriate number of clusters. The conclusion regarding the choice of the value K is to apply all three methods, the gap statistic, the elbow method and the average silhouette method. If they agree on the value for K, then the decision is made. If the three methods do not agree, a majority vote has to be taken or an educated guess has to be made.

There might be some prior knowledge about the underlying clusters in the data that can be incorporated here. A range of values for K can also be fitted and the most appropriate model can be chosen from this range. There also exists an R package called **NbClust** that can calculate 30 different statistics for determining the value of K. It can be included in the analysis if the three methods discussed above completely disagree. Otherwise it might be unnecessary duplication of work (Charrad, Ghazzali, Boiteau, & Niknafs, 2014).

4.4.1.2 Fitting K-means

We can now fit a K-means clustering to Scenario A of which the R code can be seen below. The known value $K = 3$ will be used. This value is only known because the data were generated from a known structure. With a real-world dataset the value of K will have to be estimated.

```
set.seed(98745)
kmeans_A <- kmeans(x = XDatSet_A[,-11], centers = 3, iter.max = 20,
                   nstart = 25) #algorithm = "Lloyd")
#Hartigan-Wong algorithm used as default
centers_Akm <- kmeans_A$centers[,-11] %*% pca_A$rotation[, c(1,2)]
```

We will create a plot of the K-means clustering with the data observations colour-coded in terms of their cluster labels. The R code used for this purpose similar to the R code for Figure 11 and will not be given again. We will also create a silhouette plot by using the function *silhouette* from the package **cluster** (Maechler *et al.*, 2018). We defined the concept of a silhouette of an observation in the literature review. A larger silhouette value means that an observation is nested well inside its own cluster and has a high dissimilarity from its nearest neighbour cluster. This plot returns the silhouette value for every observation in the cluster and plots these values as a very dense bar chart. We also receive more information per cluster, such as the number of observations per cluster and the average silhouette value per cluster. We are also given the overall average silhouette value.

As the two plotting outcomes are of different types, the command *par(mfrow = c())* or the function *grid.arrange* cannot be used. We have to create an empty plotting grid and then

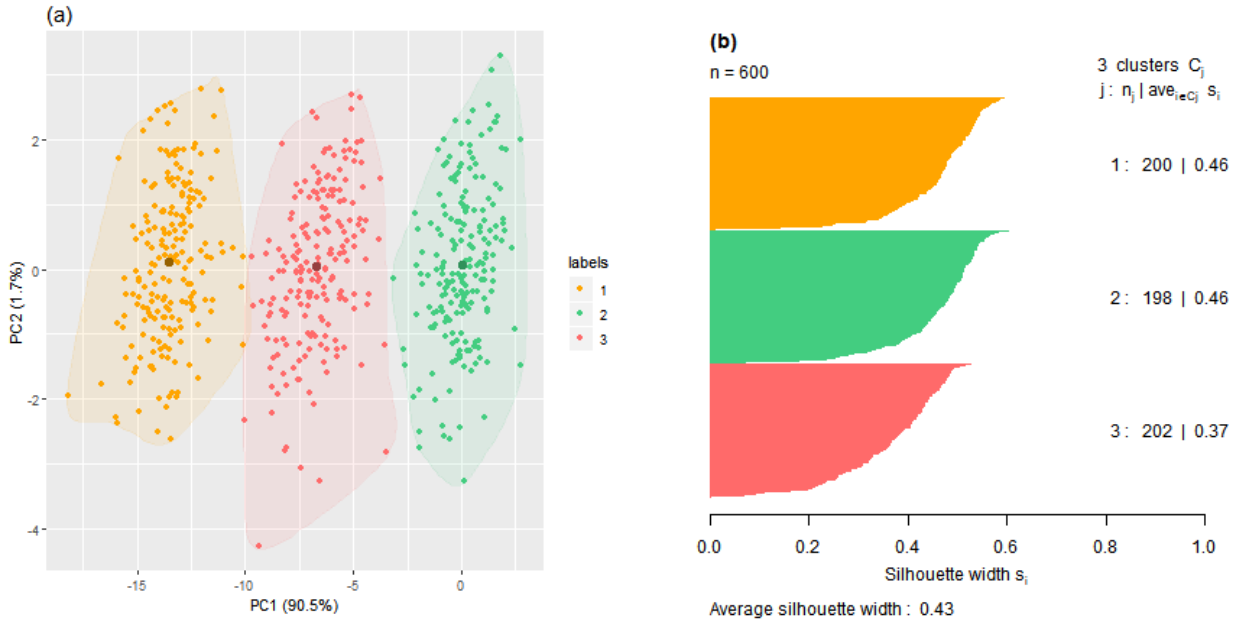


Figure 13: Scenario A plotted in terms of the first two PCs with class labels from K-means clustering (a) and the silhouette plot (b)

specify the specific viewports we would like to place our next plot into. We use a combination of *pushViewport* and *popViewport*, from the package **grid**, to plot a *ggplot* object and a base R plot in the same grid (R Core Team, 2018). The R code for this can be seen below.

```
D_mat_euclid_A <- dist(XDatSet_A[,-11], method = "euclidean")
Sil_Akm <- silhouette(x = kmeans_A$cluster, dist = D_mat_euclid_A)

plot.new()
pushViewport(viewport(layout = grid.layout(1, 2)))

pushViewport(viewport(layout.pos.row = 1, layout.pos.col = 1))
print(plot_Akm, newpage = FALSE)
popViewport()

pushViewport(viewport(layout.pos.row = 1, layout.pos.col = 2))
par(fig = gridFIG(), new = TRUE)
plot(Sil_Akm, main = "(b)", cex.main = 0.75, font.main = 1,
                        col = c("orange", "seagreen3", "indianred1"),
                        border = NA)
popViewport()
```

In Figure 13(a) we can see the K-means clusters for data Scenario A. Here the colour-coded clusters are plotted with the first two PCs as the axes. The center for each cluster has also been indicated with a larger plotting character. If we compare this figure to Figure 11(a) we can see that the colours for the clusters have been re-ordered. The K-means algorithm,

together with any of the clustering algorithms, has no sense of the ordering of the clusters and labels them in a random way. As long as the observations that belong together are placed in the same cluster, the actual cluster labels are not important. K-means clustering performed well in separating the three clusters for Scenario A.

Figure 13(b) shows the silhouette plot for K-means clustering of Scenario A using Euclidean distance as the distance measure. A value close to one for the silhouette index indicates a "good" clustering in terms of compact and isolated clusters. In Figure 13(b) we see that the average silhouette value is 0.43, so the K-means clusters can be seen as an "average" clustering. This may be attributed to the fact that the clusters lie close together and are not very isolated. The cluster sizes are almost perfectly balanced and cluster 1 and 2 have larger silhouette values, indicating that the points inside the clusters lie close together and far from the second nearest cluster. We can see from Figure 13(a) why cluster 3 has a lower average silhouette value as there are more outliers present in this cluster.

4.4.2 *K*-medoids Clustering

We would like to fit a K-medoids or a Partitioning Around Medoids (PAM) clustering to data Scenario A. For this we will use the function *pam* from the package **cluster** (Maechler *et al.*, 2018). The important inputs here are:

x: the input matrix X or the dissimilarity matrix D
k: the number of clusters K
diss: a logical parameter where TRUE indicates that **x** is a dissimilarity matrix, FALSE indicates that **x** is the original input matrix.
metric: the distance measure for dissimilarity, currently only supports Euclidean and Manhattan distance; only used if **diss** is FALSE; default is Euclidean
medoids: the vector of indices specifying initial medoids if desired; default is NULL
stand: a logical parameter where TRUE indicates that **x** will be standardised before the dissimilarities are calculated; only used if **diss** is FALSE; default is FALSE

The most important outputs include:

medoids: the $K \times p$ matrix of cluster medoids, $\{\boldsymbol{m}_1, \boldsymbol{m}_2, ...\boldsymbol{m}_K\}$
id.med: the vector of the indices of the medoids in the original dataset
clustering: a vector of length N of the cluster assignment for each observation

The function *pam* only supports datasets with $N \leq 65536$. When datasets larger than this are considered, the function *clara*, which will be discussed later and also comes from the package **cluster**, should be used. The clusters found by the PAM algorithm are plotted in the same way as the K-means clusters and the code will not be repeated. We tested the gap statistic, the elbow method and the average silhouette method for finding K and the respective values found were again 3, 3 and 2. The value for K will be taken as the true value, which is 3. The code that fits the PAM algorithm to Scenario A and returns the cluster centers can be found below.

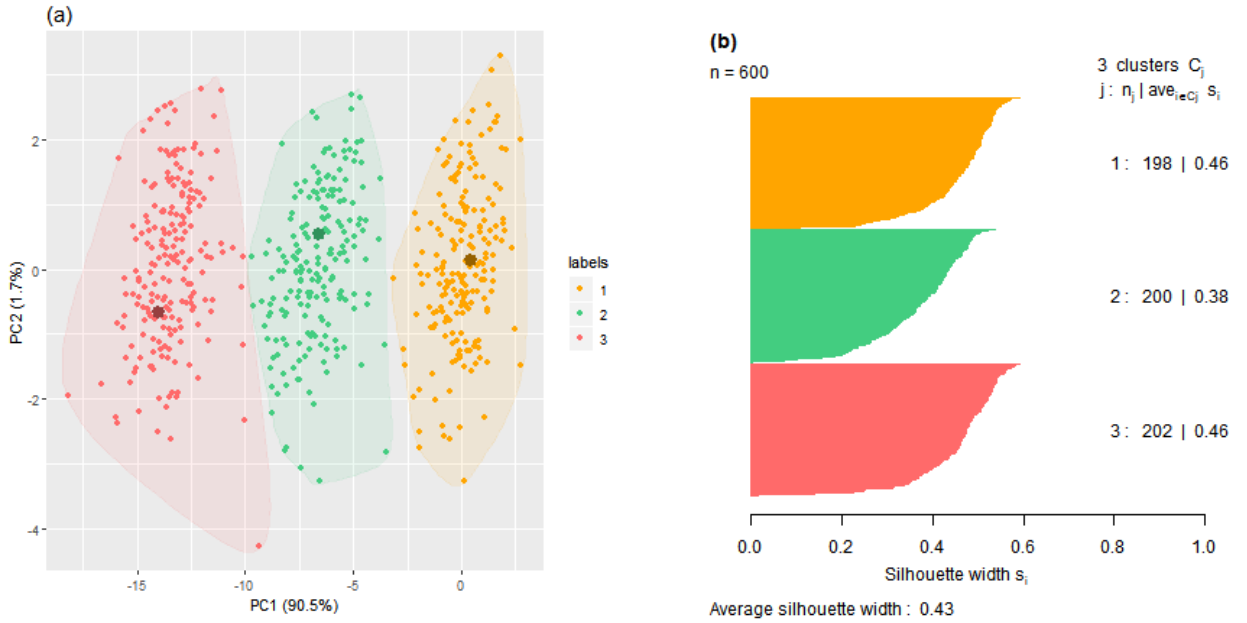


Figure 14: Scenario A plotted in terms of the first two PCs with class labels from K-medoids clustering (a) and the silhouette plot (b)

```
set.seed(98745)
pam_A <- pam(x = D_mat_euclid_A, k = 3, diss = TRUE, stand = FALSE)

centers_Apam <- XDatSet_A[pam_A$medoids,-11] %*% pca_A$rotation[, c(1,2)]
```

In Figure 14(a) and (b) we can see the outcome of the PAM algorithm being fit to the data of Scenario A. Figure 14(a) again plots the colour-coded clusters with the first two PCs as the axes. The medoid for each cluster has been indicated with a larger plotting character. In Figure 13(a) we can see that the cluster centers lie perfectly in the middle of the clusters. Comparing this to Figure 14(a) we can see that the medoids lie slightly off-center in the clusters and this is because the medoids have to be actual observations in the dataset. From Figure 14(a) it can be seen that K-medoids clustering did a relatively good job at separating the three classes with only minor misclassifications. The outliers from cluster 2 were in this case assigned to cluster 3, whereas K-means assigned them correctly.

Figure 14(b) shows the silhouette plot for K-medoids clustering of Scenario A using Euclidean distance as the distance measure. We see the average silhouette value is again 0.43. The cluster sizes are almost perfectly balanced and cluster 1 and 3 have larger silhouette values. Intuitively cluster 3 should have the lowest silhouette value as it has the most outliers, but the positioning of the medoid has alleviated this problem, as can be seen in Figure 14(b). This might also emphasise the affect of outliers on the PAM algorithm and that there might be value in removing outliers before running the algorithm.

4.4.3 Clustering Large Applications (CLARA)

We would like to fit the CLARA clustering algorithm to data Scenario A. Even though it is geared towards large datasets it will be used here for illustrative purposes. When samples are small to medium sized it might be more appropriate to use the PAM algorithm. PAM has a hard limit of $N = 65536$, so any datasets larger than that would necessarily have to be handled by CLARA or another algorithm for large datasets. For the CLARA algorithm we will use the function *clara* from the package **cluster** (Maechler *et al.*, 2018). The important inputs here are:

x: the input matrix X
k: the number of clusters K
metric: the distance measure for dissimilarity, currently only supports Euclidean, Manhattan and Jaccard distance; default is Euclidean
stand: a logical parameter where TRUE indicates that **x** will be standardised before the dissimilarities are calculated; default is FALSE
samples: the number of samples to be drawn from the large dataset; default is 5; recommended to be set larger
sampsize: the size of each sample; default is $40 + 2 \times K$; recommended to be set larger
medoids.x: a logical indicator where TRUE indicates that the medoids themselves will be kept, if FALSE then the indices of the medoids will be returned and memory is saved
keep.data: a logical indicator where TRUE indicates that the data will be kept, if FALSE memory and time will be saved
rngR: a logical indicator where if TRUE an R random number generator is used, if FALSE the old random number generator from the original FORTRAN code is used
pamLike: a logical indicator where TRUE indicates that the SWAP phase from the PAM algorithm should also take place, if FALSE the SWAP phase will be skipped

The most important outputs are:

sample: the labels of the observations from the "best" sample, the sample that was chosen as the best clustering by the CLARA algorithm
medoids: a $K \times p$ matrix of cluster medoids, $\{\boldsymbol{m}_1, \boldsymbol{m}_2, ...\boldsymbol{m}_K\}$; NULL if **medoids.x** = FALSE
id.med: the vector of the indices of the medoids in the original dataset
clustering: the vector of length N of the cluster assignment for each observation

We will use $K = 3$, the true number of clusters. The gap statistic, elbow method and average silhouette method returned 3, 3 and 2 as the values for K, respectively. The R code that fits the CLARA algorithm to Scenario A and returns the cluster centers can be found below.

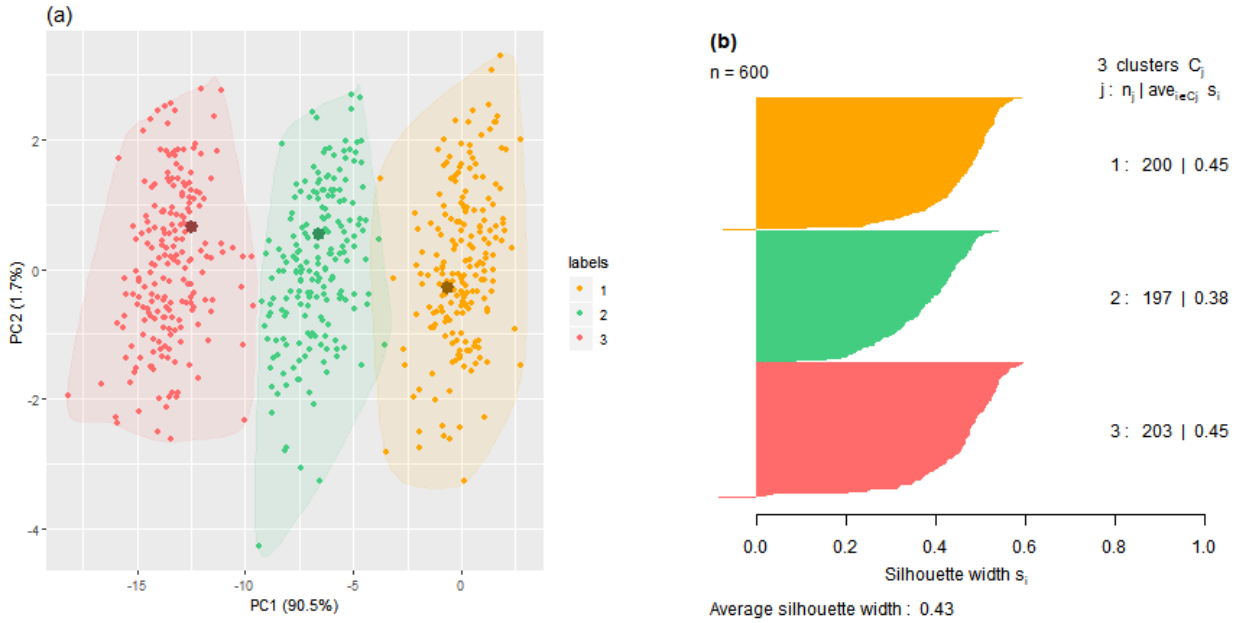


Figure 15: Scenario A plotted in terms of the first two PCs with class labels from the CLARA algorithm (a) and the silhouette plot for Scenario A (b)

```
set.seed(1014789)
clara_A <- clara(x = XDatSet_A[,-11], k = 3, metric = "euclidean",
                 samples = 10, sampsize = 100, medoids.x = FALSE,
                 keep.data = FALSE, rngR = TRUE, pamLike = TRUE)

centers_Aclara <- XDatSet_A[clara_A$i.med,-11] %*% pca_A$rotation[, c(1,2)]
```

In Figure 15(a) and (b) we can see the outcome of the CLARA algorithm fit to Scenario A. Figure 15(a) plots the colour-coded clusters with the first two PCs as the axes. The medoid for each cluster is indicated with a larger plotting character. Again the medoids lie slightly off-center in the clusters and this is because the medoids have to be actual observations in the dataset and because the medoids were only determined from samples of the data. From Figure 15(a) it can be seen that CLARA clustering did a relatively good job at separating the three clusters with only minor misclassifications.

Figure 15(b) shows the silhouette plot for the CLARA clustering of Scenario A using Euclidean distance as the distance measure. We see the average silhouette value is again 0.43. The cluster sizes are fairly balanced and cluster 1 and 3 have larger silhouette values. The negative silhouette values come from the misclassified observations.

4.4.4 Agglomerative Hierarchical Clustering

We would like to apply agglomerative hierarchical clustering to the data of Scenario A. We would also like to test the different forms of linkage for agglomerative hierarchical clustering.

In the **stats** package there exists a function *hclust* used for agglomerative hierarchical clustering (R Core Team, 2018). The important inputs here are:

d: the dissimilarity matrix D
method: the method of linkage; the options are Ward(two different types), Single, Complete, Average, Mcquitty, Median and Centroid linkage

The most important outputs include:

merge: an $(N-1) \times 2$ matrix; the rows represent the steps of the clustering algorithm; the two elements of the rows represent the two clusters that were merged; if a value in the row is negative it represents a singleton.
height: a vector of length $N-1$ containing the values of dissimilarity at which the clusters were merged; will go on the vertical axis of a dendrogram.
order: a vector of length N returning a permutation of the original observations to ensure that branches do not cross in the plotting of a dendrogram
labels: the cluster label for each observation

There is also a function in the **cluster** package, called *agnes*, that performs agglomerative hierarchical clustering in a similar way to *hclust* (Maechler *et al.*, 2018). This function takes the original dataset X as an input, and a metric for distance, for example Euclidean, needs to be specified. The linkage methods for this function includes single, complete, average, Ward (which corresponds to *hclust*'s second Ward type), weighted (which corresponds to *hclust*'s Mcquitty) and a few others which will not be discussed here. The functions *hclust* and *agnes* can return agglomerative coefficients (ACs) that measure the level of hierarchical clustering structure in the underlying data. Both *hclust* and *agnes* provide output that can be plotted on a dendrogram. Since *hclust* has more forms of linkage available and produces a similar output to *agnes*, we feel comfortable only using *hclust* for fitting agglomerative hierarchical clustering.

We will investigate the different forms of linkage that can be used for Scenario A. Because the clusters in Scenario A are fairly compact but close together we believe that complete linkage will perform well. We have already looked at all the different forms of linkage in the literature review, so only the linkage forms with sensible dendrograms will be discussed here. We will leave out single, centroid and median linkage. Below the code for fitting agglomerative hierarchical clustering to Scenario A can be seen for complete linkage. We will also use the package **dendextend** for making adjustments to the visualisation of the dendrograms (Galili, 2015).

```
hclust_complete_A <- hclust(d = D_mat_euclid_A, method = "complete")
hclust_complete_dend_A <- as.dendrogram(hclust_complete_A)
hclust_complete_dend2_A <- color_branches(hclust_complete_dend_A, k=3,
                            col = c("orange", "seagreen3", "indianred1"))
```

It is important to note whether an algorithm needs Euclidean distance, or squared Euclidean distance, as input. In the *hclust* function we find *ward.D* and *ward.D2* linkage methods. The difference between the two is that *ward.D* uses squared Euclidean distance as input and *ward.D2* only needs Euclidean distance. It is also recommended that the heights of the

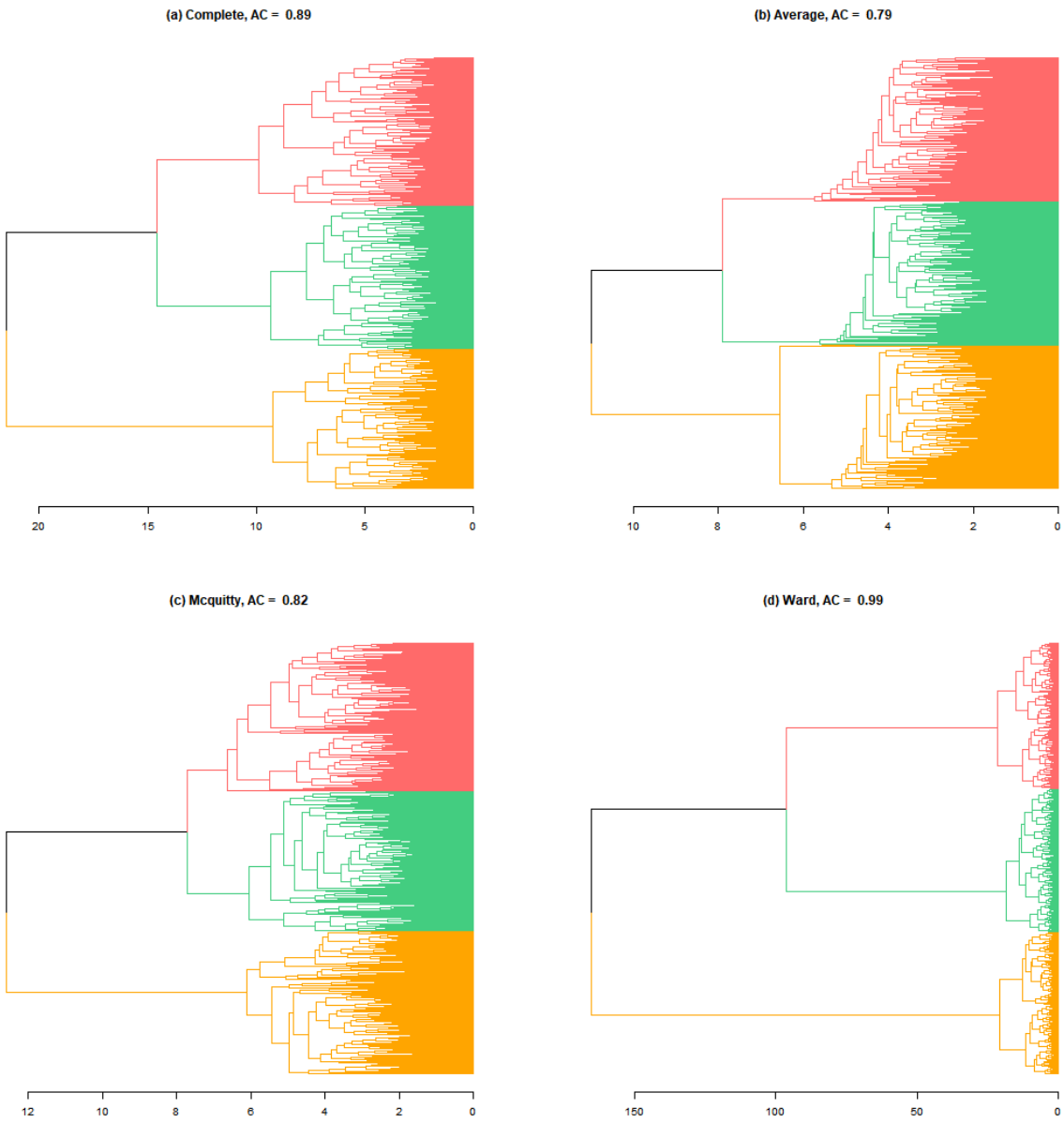


Figure 16: Dendrograms for different forms of linkage for Scenario A

dendrogram be squared before plotting for *ward.D*. We will assume equivalence of the two methods and not investigate it further, thus only *ward.D2* linkage will be used (Murtagh & Legendre, 2014).

The value for K will be taken as the true value, 3, for each of the linkage types. We did confirm this value for each linkage method with the gap statistic, elbow method and average silhouette method. For all complete, average, McQuitty and Ward linkage the values for K were 3 as given by the gap statistic, 3 as given by the elbow method and 2 as given by the average silhouette method.

In Figure 16(a) to (d) we can see the dendrograms for (a) complete, (b) average, (c) McQuitty and (d) Ward linkage clustering in Scenario A. Looking at the dendrograms, all the cluster methods seem to have returned balanced clusters. In the heading of each dendrogram the AC can be found. The closer to 1 the AC-value lies, the more hierarchical clustering structure was found in the data. In Figure 16(d) we can see that Ward linkage clustering found the highest level of clustering structure in the data.

In Figure 17(a) to (h) we can see the outcome of agglomerative hierarchical clustering, with different forms of linkage, being fit to the data in Scenario A. The plots in the first column are colour-coded clusters with the first two PCs as the axes. The center for each cluster is not shown here as it is not part of the clustering algorithm. In the second column of Figure 17 we can see the silhouette plots for the different linkage methods. From Figure 17(a), (c) and (e) it can be seen that complete, average and McQuitty linkage made some clear misclassifications. There are some points lying very clearly in a neighbouring cluster and this also accounts for the negative values that can be seen in the corresponding silhouette plots in 17(b), (d) and (f), respectively. Ward linkage, seen in Figure 17(g) and (h), did the best job at separating the clusters and no obvious misclassifications were made. No negative silhouette values are present here.

We see the average silhouette values for all the clusterings are again 0.43. All of the suggested clusters are fairly balanced, but again Ward linkage produced the most balanced clustering suggestion. For all the silhouette plots in the right column of Figure 17 we can see that the average silhouette values for cluster 1 and 3 were higher than for cluster 2. The reasoning for this is again the clear outliers present in cluster 2.

4.4.5 Divisive Hierarchical Clustering

We would like to apply divisive hierarchical clustering to Scenario A. The implementation and interpretation are very similar to agglomerative hierarchical clustering and a dendrogram is also returned. Divisive hierarchical clustering does not implement different forms of linkage, only the maximum diameter of the clusters are used. In the **cluster** package there exists a function *diana* that will be used for divisive hierarchical clustering (Maechler *et al.*, 2018). The important inputs here are:

x: the input matrix X or the dissimilarity matrix D
diss: a logical parameter where TRUE indicates that **x** is a dissimilarity matrix, FALSE

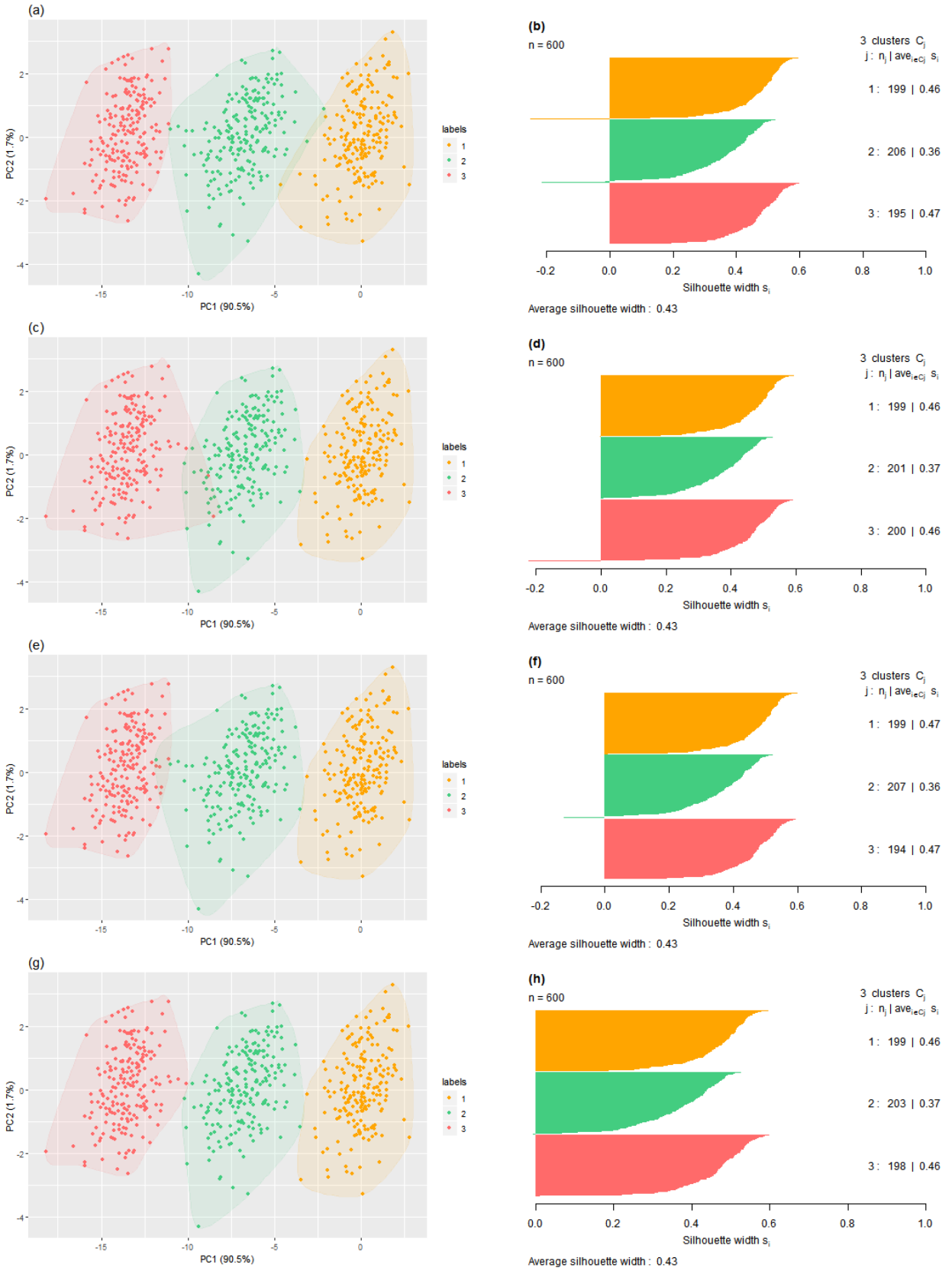


Figure 17: Agglomerative hierarchical clustering data plots and silhouette plots with (a)-(b) complete, (c)-(d) average, (e)-(f) McQuitty and (g)-(h) Ward linkage in Scenario A

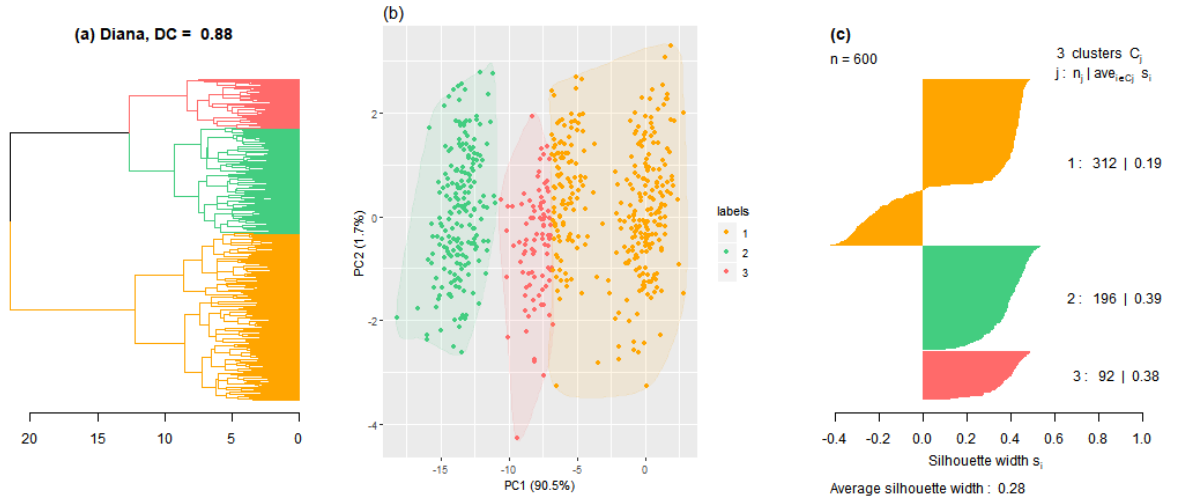


Figure 18: Divisive hierarchical clustering (a) dendrogram, (b) cluster plot and (c) silhouette plot for Scenario A

indicates that **x** is the original input matrix
metric: the distance measure for dissimilarity, currently only supports Euclidean and Manhattan distance; only used if **diss** is FALSE; default is Euclidean
stand: a logical parameter where TRUE indicates that **x** will be standardised before the dissimilarities are calculated; only used if **diss** is FALSE; default is FALSE

The most important outputs include:

order: a vector of length N returning a permutation of the original observations to ensure that branches do not cross in the plotting of a dendrogram
height: a vector with diameters of clusters before splitting occurred
dc: the divisive coefficient, giving an indication of hierarchical clustering structure in the data
merge: a matrix of size $(N-1) \times 2$; the rows represent the steps of the clustering algorithm; the two elements of the rows represent the two clusters that were formed at each split; if a value in the row is negative it represents a singleton that was split off

Below we can see the R code for applying the divisive hierarchical clustering algorithm to Scenario A. We will use the true number of clusters, $K = 3$. The gap statistic, elbow method and average silhouette method returned values for K of 4, 4 and 2, respectively. This is slightly disheartening because we would have liked our methods of choosing K with the DIANA algorithm to recognise the "correct" cluster structure in the data.

```
diana_A <- diana(x = D_mat_euclid_A, diss = TRUE, stand = FALSE)
diana_dend_A <- as.dendrogram(diana_A)
diana_dend2_A <- color_branches(diana_dend_A, k=3,
                      col = c("orange", "seagreen3", "indianred1"))
```

In Figure 18(a) to (c) we can see the plots for the divisive hierarchical clustering algorithm, DIANA, in Scenario A. Figure 18(a) shows the dendrogram for this clustering and here we

can already see that the first split in the algorithm basically splits the data in half. Since the DIANA algorithm is error non-correcting, there is no way for the observations of cluster 2 that were assigned to cluster 1 to be put in their correct cluster.

The divisive coefficient (DC) was quite high, meaning that the DIANA algorithm found evidence of a hierarchical clustering structure in the data. In Figure 18(b), the colour-coded clusters with the first two PCs as the axes can be seen. The center for each cluster is not shown here as it is not part of the clustering algorithm. Here we can clearly see that cluster 1 is much larger than it should be. Figure 18(c) contains the silhouette plots for this clustering. The cluster sizes are extremely unbalanced and the overall average silhouette value of 0.28 is quite low. The very low average silhouette value and the negative silhouette values for cluster 1 can be explained by the fact that 1 contains many observations that belong to cluster 3.

4.4.6 Self-Organising Maps (SOMs)

Now we fit a few variations of SOMs to Scenario A. The main purpose is to unpack the R software available to use for SOMs and to find informative ways to visualise our results. The package we will use is called **kohonen** and it was last updated on 17 August 2018 (Wehrens & Buydens, 2007; Wehrens & Kruisselbrink, 2018). The main uses of this package are to fit unsupervised SOMs and also SuperSOMs; these SOMs have the ability to accept more than one layer of input data. We will not investigate them further here. Another part of the package that we will not investigate is supervised SOMs. This package can also be used to map new data observations onto an already trained SOM. Firstly, the function *somgrid* is used to specify the dimensions and the form of the prototype grid. The important inputs here are:

xdim: the number of columns of the prototype grid
ydim: the number of rows of the prototype grid
topo: the topology of the grid; options are "hexagonal" or "rectangular"
neighbourhood.fct: the neighbourhood kernel function; options are "bubble" (refers to an indicator weight function) or "gaussian"
toroidal: a logical value where TRUE indicates that the prototype grid has to be toroidal, *i.e.* the bottom and the top of the grid are connected and the left and the right sides of the grid are connected

This function outputs the map that has to be fed into the SOM training function called *som*. The important inputs here are:

X: the input matrix X in matrix format (does not accept dataframes)
grid: output from the function *somgrid*
rlen: the number of data epochs
alpha: the learning rate, specified as the initialisation value and the end value; the algorithm will linearly decrease from the one value to the other over the iterations of the data; ignored if batch SOM is chosen; default is 0.05 to 0.01
radius: radius of the neighbourhood, specified (as with alpha) as the initialisation value and the size of the final neighbourhood; if only one value is provided, the algorithm will

decrease to zero over the iterations; default is $\frac{2}{3}$ of total distances between all prototypes;
radius ≤ 1 only updates prototype itself
mode: the algorithm type; options are "online", "batch" and "pbatch"; we will only use "online" and "batch"
init: the initialisation vectors for the prototypes in terms of a matrix with the rows as the prototype vectors; if **init** = NULL, then chosen randomly (without replacement) from X

The most important outputs include:

unit.classif: the BMU for each observation
distances: the distances from the observations to their BMUs
code: the list of the prototype vectors
changes: the mean distance of all the observations to their prototypes

We now fit a SOM onto the data of Scenario A using a hexagonal map topology. To initialise the SOM we need to decide on the dimensions of the map. The ratio of the first eigenvalue linked to the largest PC to the second eigenvalue linked to the second largest PC is 52.99. This means that the SOM map should be initialised in the ratio $53:1$. This ratio is quite high, so for convenience the map will have a ratio of $10:3$. We will also use a Gaussian neighbourhood function and will test the batch SOM and the online SOM algorithms. We will also test random initialisation and linear initialisation.

As the **kohonen** package does not calculate linear initialisation, it must be calculated manually and provided to the function. The first two PCs of Scenario A are PC_1 = (-0.45, -0.45, -0.45, -0.45, -0.44, -0.01, 0, 0, -0.01, -0.01) and PC_2 = (0.11, -0.13, 0.01, 0.02, 0.01, -0.41, -0.53, -0.46, -0.32, -0.45). We also define two vectors $\boldsymbol{a}_1$ = (-1, -0.78, -0.56, -0.33, -0.11, 0.11, 0.33, 0.56, 0.78, 1) and $\boldsymbol{a}_2$ = (-1, 0, 1). The vectors $\boldsymbol{a}_1$ and $\boldsymbol{a}_2$ are specified to be equally distributed between -1 and 1 along both dimensions of the SOM. We can use these vectors to initialise the weight vectors for the SOM algorithm. Below the code for linear initialisation of the SOM grid can be seen.

```
a_1 <- seq(from =-1, to= 1, length.out = 10)
a_2 <- seq(from =-1, to= 1, length.out = 3)
PC_1 <- pca_A$rotation[,1]
PC_2 <- pca_A$rotation[,2]
a_1_PC_1 <- a_1 %*% t(PC_1)
a_2_PC_2 <- a_2 %*% t(PC_2)
mat_1 <- rbind(a_1_PC_1, a_1_PC_1, a_1_PC_1)

mat2_index <- c(rep(1,10), rep(2,10), rep(3, 10))
mat_2 <- a_2_PC_2[mat2_index,]

proto_vec <- mat_1 + mat_2
```

We can now test four versions of the SOM algorithm, by using random initialisation with the batch SOM and the online SOM and also linear initialisation with the batch SOM and the

online SOM. We also plot the U-matrices for these map outcomes. A U-matrix is a plot of the sum of the distances to all the immediate neighbours of each prototype. For example, if a hexagonal prototype grid topology is used, each prototype has 6 other prototypes surrounding it. The entry in the U-matrix will be the sum of the distances from the prototype in question to each of these 6 immediately adjacent prototypes.

This means that a colour-coded map can be created that reveals the natural clustering in the data. The R code for fitting the SOMs and plotting the U-matrices can be seen below. As a side note, the SOM algorithm can start overfitting when too many data epochs are chosen, specifically in the online algorithm. For example, using 5000 data epochs is not necessary, the method normally stabilises in less than 1000 epochs.

```
SOMgrid_A <- kohonen::somgrid(xdim = 3, ydim = 10, topo = "hexagonal",
                      neighbourhood.fct = "gaussian", toroidal = FALSE)

SOM_Arandbatch <- som(X = XDatSet_A[,-11], grid = SOMgrid_A, rlen = 1000,
                      mode = "batch")
SOM_Arandonline <- som(X = XDatSet_A[,-11], grid = SOMgrid_A, rlen = 1000,
                      alpha = c(0.5, 0.01), mode = "online")
SOM_Alinbatch <- som(X = XDatSet_A[,-11], grid = SOMgrid_A, rlen = 1000,
                      mode = "batch", init = proto_vec)
SOM_Alinonline <- som(X = XDatSet_A[,-11], grid = SOMgrid_A, rlen = 1000,
                      alpha = c(0.5, 0.01), mode = "online", init = proto_vec)

par(mfrow = c(2,2))
plot(SOM_Arandbatch, type = "dist.neighbours", main = "(a)")
plot(SOM_Arandonline, type = "dist.neighbours", main = "(b)")
plot(SOM_Alinbatch, type = "dist.neighbours", main = "(c)")
plot(SOM_Alinonline, type = "dist.neighbours", main = "(d)")
```

In Figure 19(a) to (d) we can see the U-matrices for (a) the batch SOM with random initialisation, (b) the online SOM with random initialisation, (c) the batch SOM with linear initialisation and (d) the online SOM with linear initialisation for Scenario A. In all of the U-matrices we can clearly see three clusters forming in the data. The darker the colour of the prototypes, the closer the prototypes are lying together. That means that the lighter coloured prototypes form natural boundaries for the three clusters.

We can now fit Ward linkage agglomerative hierarchical clustering on top of the SOM. To illustrate this, we will use the batch SOM with linear initialisation. In Figure 20(a) to (d) the outcome of Ward linkage on top of the batch SOM can be seen for Scenario A. In Figure 20(a) the prototype grid can be seen with the weight vectors represented as pie charts. The clusters found by Ward linkage has also been indicated. Figure 20(b) shows the dendrograms for this clustering; the AC is 0.93 and this indicates a very good natural hierarchical clustering structure in the data.

In Figure 20(c) the colour-coded clusters formed by the Ward linkage and SOM combination can be seen. Visually the clustering looks very good with a few minor misclassifications.

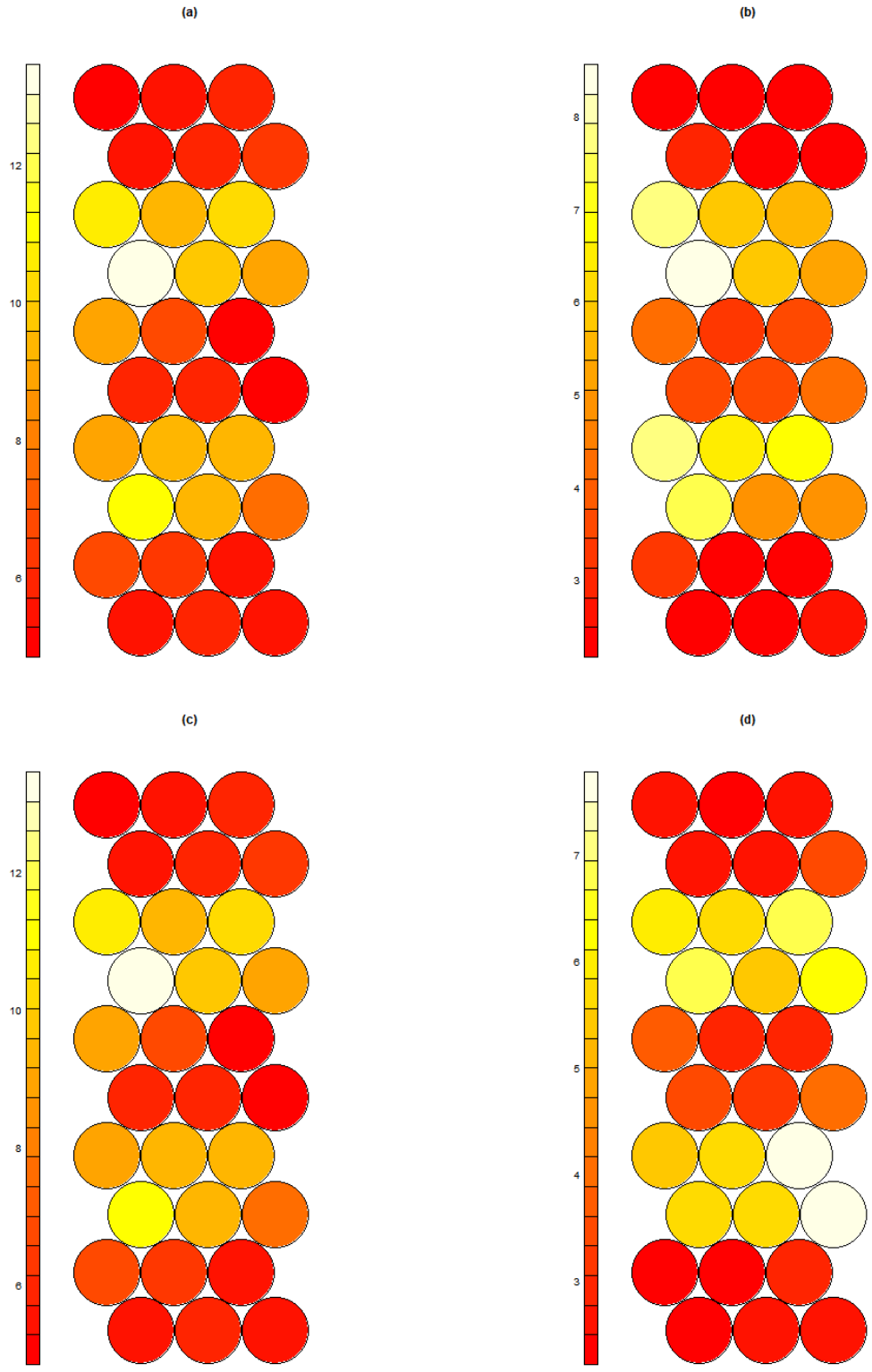


Figure 19: SOM U-matrices for (a) the batch SOM with random initialisation, (b) the online SOM with random initilisation, (c) the batch SOM with linear initialisation and (d) the online SOM with linear initialisation for Scenario A

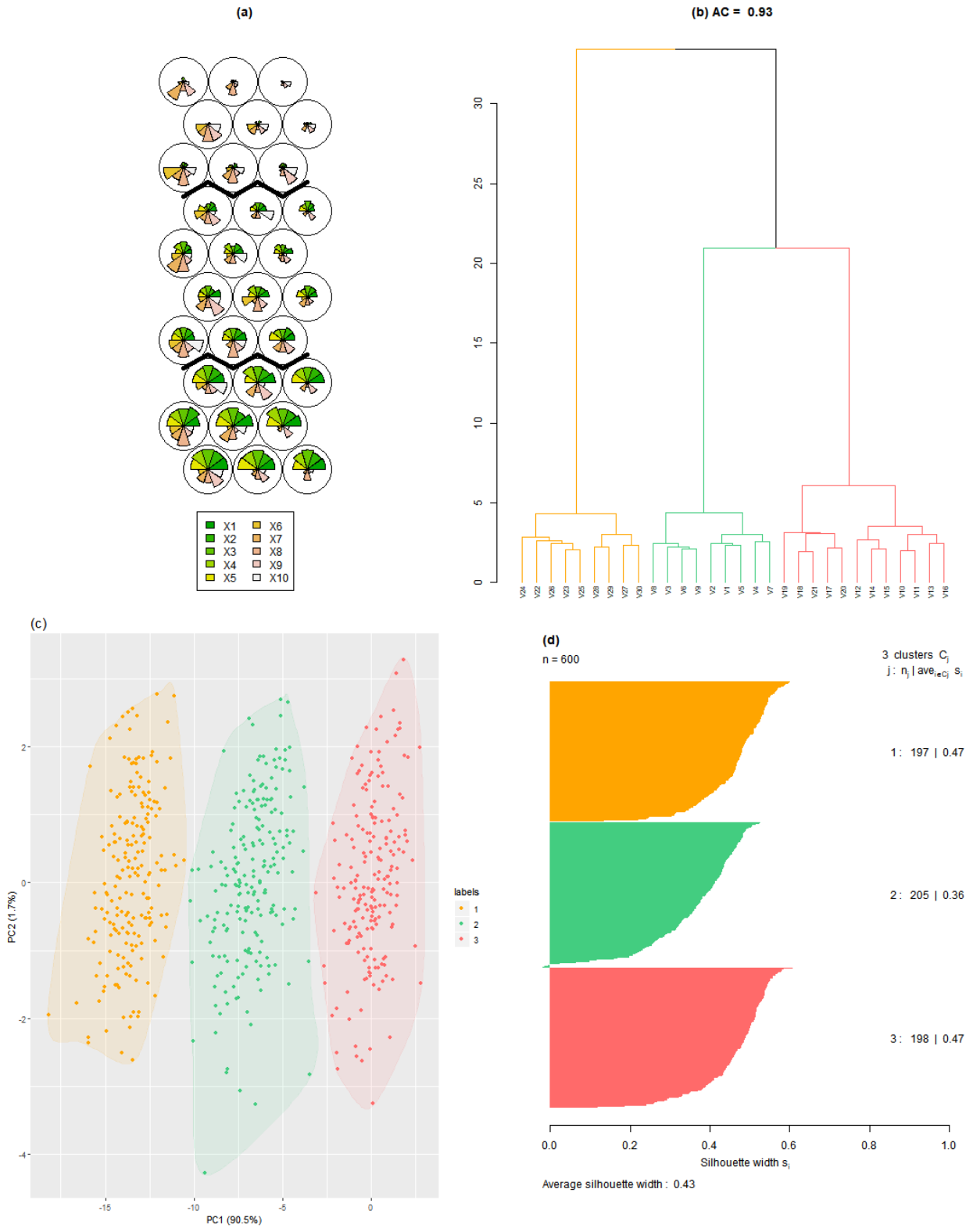


Figure 20: Ward linkage clustering on top of the batch SOM with linear initialisation for Scenario A, (a) prototype grid with pie charts representing prototype weights, (b) dendrogram of Ward linkage clustering on top of the batch SOM, (c) colour-coded cluster outcome and (d) silhouette plot

Figure 20(d), the silhouette plot, also supports this with the average silhouette value being similar to all the clustering methods we have looked at in the chapter. The returned clusters are fairly balanced and very few negative silhouette values are present. This method of clustering the SOM seems to have worked very well and will be tested on more complex data later. Although not reported here it might have been insightful to calculate the values of K suggested by the gap statistic, elbow method and the average silhouette method as well.

Another package we could use is called **SOMbrero** and it was last updated on 5 February 2018 (Villa-Vialaneix, Bendhaiba & Olteanu , 2018). The main uses of this package are to fit unsupervised SOMs on different data types; the options available are numeric data, contingency table data and also relational table data (dissimilarity matrices). For this package only the online SOM algorithm can be used, the batch SOM is not available. There is also a Graphical User Interface (GUI) available so that the functionality is available to users who are not fully R proficient (Boelaert, Bendhaiba, Olteanu & Villa-Vialaneix, 2014).

In this package there is a function called *trainSOM* used for training the SOMs. The important inputs here are:

x.data: the input matrix X, as a matrix or a dataframe
dimension: the dimensions of the prototype grid provided as a vector of length 2; default is (5,5)
topo: the only current topology available is "square"
radius.type: the neighbourhood kernel function; options are "gaussian" or "letremy" (the latter refers to an indicator weight function)
dist.type: the distance function for determining the neighbourhood; if **radius.type**="letremy", then the default **dist.type** is "letremy", if **radius.type**="gaussian", then **dist.type** is "euclidean"; other options are "maximum", "manhattan", "canberra", "binary" and "minkowski"
type: the input type for the SOM; options are "numeric", "korresp" and "relational"; default is "numeric"
mode: the SOM algorithm type; currently only "online" is available
maxit: the maximum number of iterations allowed; default value is 500, but more is recommended
nb.save: the number of snapshots (back-ups) that are taken during the training process; these are useful when the energy of the algorithm wants to be displayed visually; default is 0
verbose: a logical indicator that when TRUE reports the process of the algorithm as it evolves, if FALSE then feedback is only given after the algorithm stops; default is FALSE
proto0: the values to initialise the prototype vectors, if they are desired to be given; default value is NULL
init.proto: the method for initialising the prototype vectors; options are "random" (completely random values), "obs" (random observations chosen) and "pca" (first two PCs are used, as explained in the literature review)
scaling: the type of centering and scaling that needs to be done on the data before the algorithm starts; options for **type** = "numeric" are "unitvar" (centered and scaled),

"none" (no centering or scaling) and "center" (only centering); default is "unitvar"
eps0: the learning rate according to the formula: $\frac{0.3\times\textbf{eps0}}{1+0.2\times\frac{t}{\text{dim}}}$, where dim is the prototype grid number of columns multiplied by the number of rows

The most important outputs include:

clustering: the vector of cluster allocation indices, C
prototypes: the final prototype vectors
backup: depending on the input **nb.save**, this will return the number of back-ups made throughout the algorithm's progress

This package would deliver very similar results to the **kohonen** package for numerical data and will not be tested further. However, when other data types are present **SOMbrero** might be a very good option.

4.4.7 Discussing the validity of the clusters for Scenario A

4.4.7.1 External Criteria Validity

In Table 5 we can see the Rand, Jaccard and Folkes and Mallows indices calculated for Scenario A against the true clustering of the datasets. We have to keep in mind that the closer these three indices are to 1, the closer the clustering algorithm came to the true clustering. We can clearly see that divisive hierarchical clustering did very poorly in Scenario A; this was apparent in Figure 18 as well. The other methods all did fairly well, achieving index values above 0.9 for the Rand, Jaccard and Folkes and Mallows indices. According to all three indices, K-means did the best in Scenario A with Ward linkage clustering as a close second. After divisive hierarchical clustering, complete linkage clustering did the second worst. As the Ward linkage on the SOM was for illustrative purposes only, it will not be compared to the other methods here.

4.4.7.2 Relative Criteria Validity

In Table 5 we can also see the average silhouette, Dunn and Davies-Bouldin indices calculated for Scenario A. These indices all indicate a level of compactness of clusters and how well they are separated. We would like the average silhouette and the Dunn indices to be as large as possible and the Davies-Bouldin index to be as small as possible. It must be noted

Table 5: Scenario A Comparison of Clustering Algorithms

Index	*K*-Means	PAM	CLARA	Complete	Average	McQuitty	Ward	Divisive
Rand	0.996	0.991	0.989	0.982	0.989	0.985	0.993	0.814
Jaccard	0.987	0.974	0.967	0.948	0.967	0.955	0.98	0.595
FolkesMallows	0.993	0.987	0.983	0.973	0.983	0.977	0.99	0.749
Ave Silhouette	0.432	0.431	0.429	0.428	0.429	0.43	0.432	0.282
Dunn	0.127	0.127	0.238	0.224	0.236	0.222	0.258	0.118
DaviesBouldin	0.894	0.895	0.899	0.895	0.895	0.893	0.892	1.111

that all the average silhouette and the Dunn index values are smaller than we would have liked, similarly the Davies-Bouldin index is much larger than we would have liked. This can possibly be attributed to the fact that the clusters in Scenario A lie fairly close to each other. According to these indices, divisive hierarchical clustering still did very poorly in Scenario A. All the other methods returned fairly similar values for the different indices. The average silhouette awarded the same values to K-means clustering and Ward linkage clustering and they jointly had the best clustering in Scenario A. PAM had the second highest average silhouette value. The Dunn index gave the highest value to Ward linkage clustering and the second highest value to Average linkage clustering. The Davies-Bouldin index awarded the lowest value to Ward linkage and the second lowest to McQuitty linkage clustering.

In conclusion, summarising the values found in Table 5, Ward linkage clustering performed the best in Scenario A, returning the correct cluster labels and choosing compact and isolated clusters. As the Ward linkage on the SOM was for illustrative purposes only, it will not be compared to the other methods here.

4.5 Scenario B

For Scenario B we would like to fit the different traditional clustering algorithms, discussed before, on top of a SOM. We would also like to fit the traditional clustering algorithms without the dimension reduction step of the SOM, for comparison. This proved difficult as the dissimilarity matrix of Scenario B is big and this slowed down the clustering methods considerably. The traditional methods that were used were K-means, K-medoids and Ward linkage clustering. CLARA was also used to determine how it compares to PAM on a large dataset. The traditional clustering methods that were fit on top of the SOM were K-means, K-medoids, Ward linkage and divisive hierarchical clustering.

For the SOM the hexagonal map topology, used before, was chosen again. A Gaussian neighbourhood function was used again. Scenario B was tested on the batch SOM and the online SOM with both random and linear initialisation. The test involved fitting all four of these models to the data of Scenario B and then fitting Ward linkage clustering on top to determine how well the prototypes clustered, compared to the true clustering. This test is not perfect, as it may capture the capabilities of the Ward linkage clustering in the process as well. Different numbers of epochs were also tried, but 100 epochs seemed sufficient. For the online algorithm the learning rate was decreased from 0.1 to 0.01 over the data epochs. The grid was chosen of size 80×10 as explained below. In Table 6, the summary of the validity statistics for these tests can be seen.

Of these options the batch SOM with random initialisation was the best performing algorithm and will be used going forward. This is in line with Kohonen's (2013:53,57) recommendation that the batch SOM is more appropriate to use for practical applications. It is also in line with the recommendation of Akinduko *et al.* 2016:220-221 that random initialisation is more appropriate when the data is extremely nonlinear and complex. This means that it would be impossible to fit a straight line or hyperplane to the data. The ratio of the first eigenvalue to the second is 7.72. This means that the SOM map should be initialised in the ratio $8:1$, so

Table 6: Scenario B Comparison SOM Algorithms and Initialisations

Index	Batch Random	Batch Linear	Online Random	Online Linear
Rand	0.963	0.951	0.948	0.94
Jaccard	0.718	0.649	0.632	0.572
FolkesMallows	0.838	0.79	0.779	0.729
Ave Silhouette	0.41	0.41	0.4	0.4
DaviesBouldin	0.922	0.92	0.913	0.94

Table 7: Scenario B Comparison of Clustering Algorithms

Index	***K*-MeansS**	**PAMS**	**WardS**	**DIANAS**	***K*-Means**	**PAM**	**CLARA**	**Ward**
Rand	0.949	0.96	0.964	0.946	0.935	0.95	0.951	0.957
Jaccard	0.628	0.697	0.725	0.622	0.558	0.628	0.642	0.681
FolkesMallows	0.773	0.824	0.843	0.771	0.719	0.773	0.784	0.812
Ave Silhouette	0.428	0.405	0.407	0.37	0.393	0.412	0.43	0.408
DaviesBouldin	0.834	0.924	0.916	1.231	0.928	0.935	0.835	0.871

the maps are chosen to be of size 80×10.

In Figure 21 we can see the U-matrix of the batch SOM with random initialisation fitted to Scenario B. Here we can see from the lighter regions in the U-matrix that 5 or 6 natural clusters are forming in the data. We know that the true number of clusters is 11, but some of these clusters lie very close together and are overlapping, so it makes sense that the SOM finds it difficult to recognise all these clusters.

In Table 7 we can see the Rand, Jaccard and Folkes and Mallows indices used to compare the tested clustering methods to the underlying clustering in the data. The first four columns contain the methods fit on top of the SOM and the last four columns are the methods fit to the original dataset. We can see that Ward linkage on top of the batch SOM with random initialisation performed the best in Scenario B. The second best method seems to have been K-medoids (PAM) clustering combined with the batch SOM. K-means fit on the original dataset seems to have performed the worst in this case, but worst is very relative, as all the methods had very similar results.

In Table 7 we can also see the average silhouette and Davies-Bouldin indices calculated for Scenario B. The Dunn index could not be calculated on such a large dataset. For the average silhouette the best method was CLARA as it has the highest value. The second best was K-means combined with the batch SOM and the worst was DIANA combined with the batch SOM. DIANA also had the highest value for the Davies-Bouldin index which is undesirable. The lowest Davies-Bouldin value was found by K-means combined with the batch SOM and the second lowest was found by CLARA. This indicates that CLARA and K-means combined with the batch SOM produced the most compact and well separated clusters.

It is interesting to note how PAM and CLARA performed on the dataset of Scenario B. The PAM algorithm took very long to run on the dataset and the CLARA algorithm was almost instant, proving the usefulness of CLARA on large datasets. CLARA also performed better than PAM in Scenario B. It had higher Rand, Jaccard, Fowlkes and Mallows and average

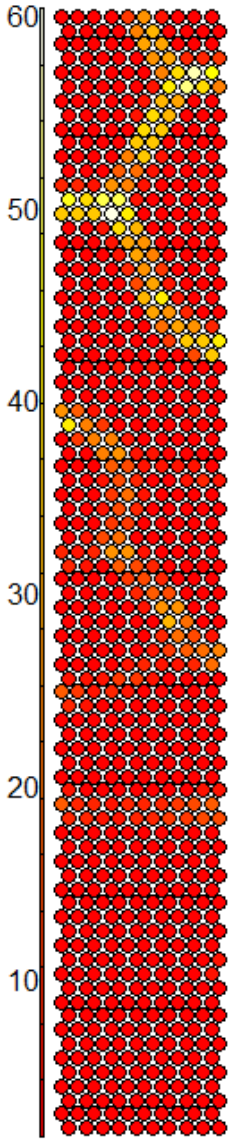


Figure 21: The U-matrix for a batch SOM with random initialisation on Scenario B

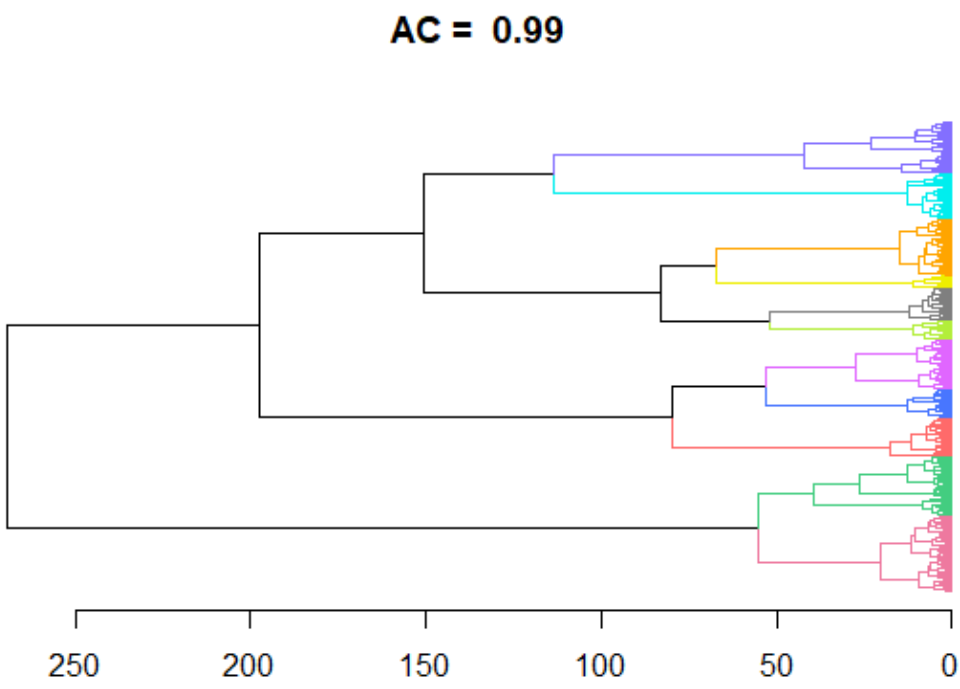


Figure 22: Dendrogram of Ward linkage on a batch SOM in Scenario B

silhouette indices and a lower value for the Davies-Bouldin index.

In Figure 22 we can see the dendrogram of Ward linkage clustering on the batch SOM. The AC in this case was 0.99, meaning that Ward linkage found overwhelming evidence for hierarchical clustering structure in the data made up of the SOM prototypes. The clusters here are also clearly not balanced, but this is to be expected as the clusters were purposefully designed to be of different sizes.

In Figure 23(a) we can see the cluster plot with colour-coded cluster labels. Here it is apparent that some misclassifications have been made. Only two of the three elongated clusters were identified. Cluster 1 is supposed to be two different clusters, but they seem to have been too overlapping to tell them apart. Figure 23(b) shows the silhouette plot of Ward linkage on the batch SOM and there clearly are some negative silhouette values present. Clusters 3, 4, 5, 6, 9, 10 and 11 seem to have been identified mostly correctly. The cluster sizes have been identified very well and the unbalanced nature of the cluster sizes was captured.

In Figure 24(a) we can see the cluster plot with colour-coded cluster labels for CLARA applied in Scenario B. The representative observations have also been indicated with larger plotting characters. Some misclassifications have clearly been made and all three of the elongated clusters were incorrectly identified. Cluster 4 is supposed to be two different clusters, but as mentioned before, they lie very close together and overlap considerably. Figure 24(b) shows

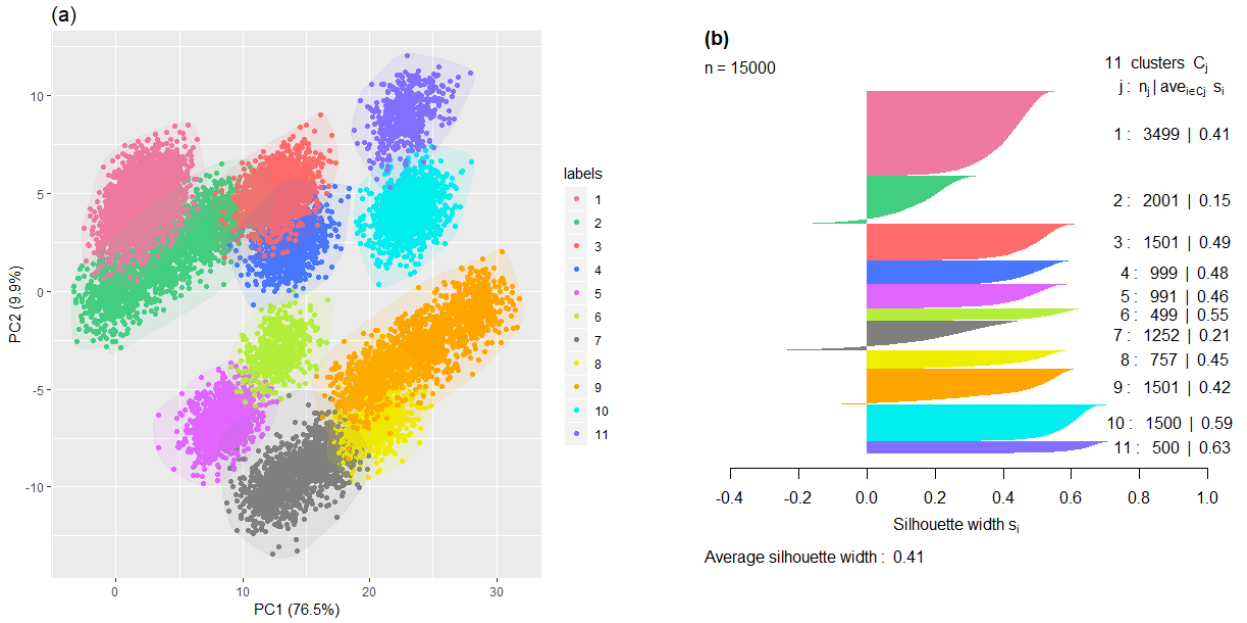


Figure 23: Ward linkage on top of the batch SOM (a) cluster plot and (b) silhouette plot for Scenario B

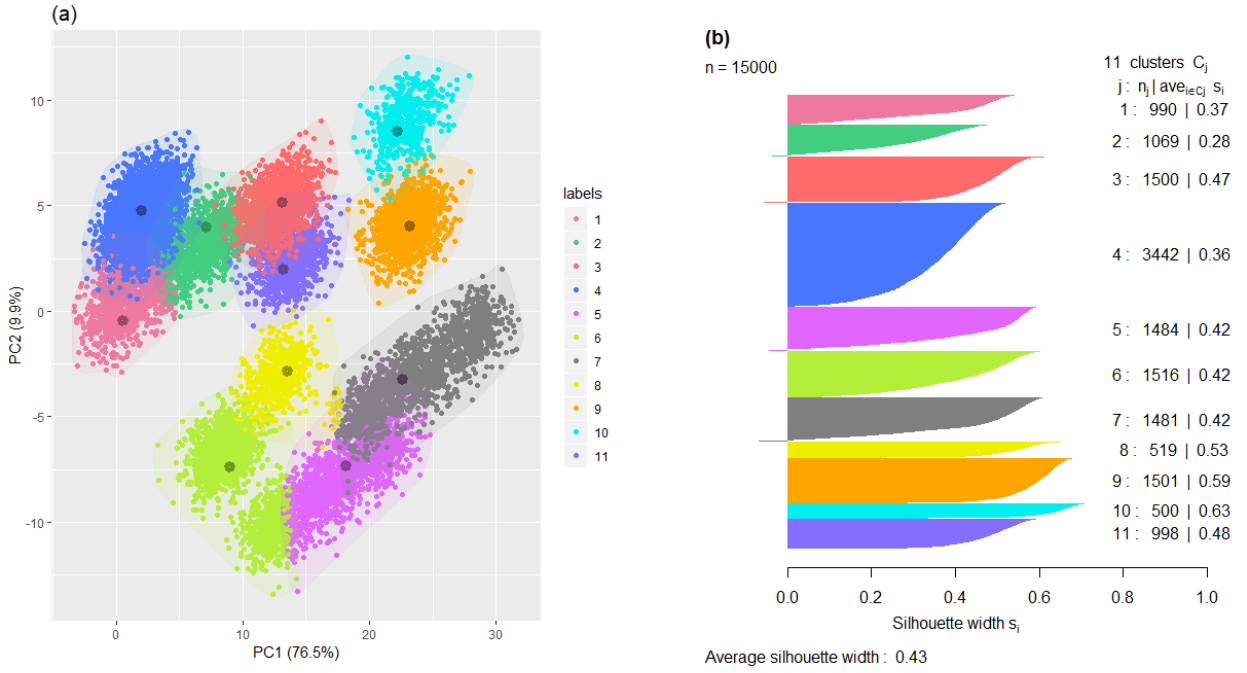


Figure 24: CLARA clustering for (a) cluster plot and (b) silhouette plot for Scenario B

the silhouette plot of the CLARA clustering and there are some negative silhouette values present, but less than for Ward linkage on the batch SOM. CLARA had the highest average silhouette value of all the clustering methods tested in Scenario B, meaning that it produced the most compact and isolated clusters. Here, clusters 3, 7, 8, 9, 10 and 11 seem to have been identified mostly correctly. The cluster sizes have been identified very well and the unbalanced nature of the cluster sizes was captured.

CLARA (and SOMs) perform on par with traditional clustering methods, but they have the added advantage to be useful in the presence of many input variables and large datasets as well. This means that we can use these methods in the real-world data example with the confidence that we will do at least as well as traditional clustering methods.

4.6 Summary

In this chapter different clustering methods were studied in a simple data scenario, Scenario A. The methods were also tested on a more complex dataset, Scenario B, and SOMs were also implemented as a dimension reduction for this scenario. Ward linkage clustering performed the best in Scenario A and also in Scenario B, combined with the batch SOM. The batch SOM as a dimension reduction technique was very successful and increased the accuracy of the classification of the data points in Scenario B. In the next chapter we will investigate real-world data in the form of transactional data per client from Bank C. We will use the batch SOM as a dimension reduction tool and apply Ward linkage hierarchical clustering to find clusters in the data.

CHAPTER 5: DATA EXTRACTION AND CLEANING

5 Data Extraction and Cleaning

5.1 Introduction

In the literature review we investigated the traditional clustering methods of K-means, K-medoids and hierarchical clustering. We also investigated SOMs and some of its variants such as GSOMs and GHSOMs. In the simulation study we looked at these methods in a controlled environment and determined which methods performed the best on simulated data. The traditional clustering methods were also fit on top of the SOMs to determine if there are some methods that are more appropriate for this task. The conclusion from this study was that Ward linkage hierarchical clustering performs well on its own and also on top of the SOM. CLARA also performed better than PAM on a larger data sample. We determined that the batch SOM with random initialisation is the best SOM algorithm to use for large and high-dimensional datasets.

In this chapter we introduce real-world data. This real-world data consist of transactional data from Bank C and will be used to determine if certain financial behaviours can be clustered together in the data. The format that will be used is similar to the simulation study: different sampling designs will be used to determine how the most value can be extracted from the data. The data is unsupervised, large and high-dimensional. This means that the data investigation can be quite subjective and a trial-and-error process can help with finding knowledge hidden in the data.

5.2 Bank C Transactional Data

The dataset that will be used in this research project was provided by Bank C and it comes from the transaction histories of the clients of the bank. The dataset, before any cleaning, records every transaction a client makes from their Bank C savings account. The nature of the transactions and the channel it was actioned through are also captured. There are details about the transactions available, such as amount, the account balance after the transaction and the transaction description. These transactions may be financial or non-financial, but the attention will be focused on financial transactions. Fees and interest payments for Bank C itself will be excluded, along with transfers clients make between their own accounts, since these do not truly reflect a flow of funds. The initial data extractions were done in the SQL query language from the data warehouse of Bank C.

In this data warehouse there is a specific group of datasets which is built up from pre-engineered features from transactional data. It is a work in progress and there are four datasets of data currently available to be used. Only certain variables from these features will be extracted. The first set is called Account Balance History and focuses on the movement

of the client's savings account balance from month to month. The second dataset is called Outflows History and is a summary of the outflows (debit transactions) from a client's account from month to month. The third dataset we have available is the Inflows History, which is a summary of the inflows (credit transactions) into a client's account, month to month. The last set that is available is called Top 50 Variables and is a summary of variables that are used very often in Bank C's data models. This includes various channels used for outflows, for example the cellphone application or cash withdrawals. It also includes branch visits and card-not-present sales, for example Snapscan or Zapper. There are some variables specifically capturing negative information, for example the number of times the client had insufficient funds or disputed a debit order.

5.3 Data Sampling

The variables in all four of these datasets are available as snapshots per client at the end of the month. The last date in the data sample for this research project was 31 August 2018. In the data extraction these variables were summarised for the last 3 months, which would mean a summary of the client's transactional activity from 1 June 2018 to 31 August 2018. A 6 month summary, for 1 March 2018 to 31 August 2018, and a 12 month summary, 1 September 2017 to 31 August 2018, were also created. The reason for creating these summaries is to determine whether the clustering structure in the data remains constant for different time periods of observation for the same clients.

It was also decided that each client in the sample had to have at least R2000 deposited into their account every month from September 2017 to August 2018. This inherently requires that the clients in the sample would have active accounts at Bank C from at least September 2017. Similarly it was required that at least 5 debit transactions were to be performed each month from September 2017 to August 2018. These requirements were set in place to ensure that the clients in the sample were financially active throughout the observation period. After these requirements were implemented a sample of 1 735 276 Bank C clients were left.

From the 3 month summary dataset for the 1 735 276 clients, three more datasets were created. These consisted of roughly 10%, 6% and 3% of the clients and resulted in three datasets of sizes 173 648, 104 463 and 52 145, respectively. These subsets are much smaller than the original dataset, but they are still representative of the full populations as the data were sampled randomly. The three datasets were named 3M_SAMP1, 3M_SAMP2 and 3M_SAMP3, respectively, for ease of reference. The sampling was done randomly and without replacement. The reason for creating the three datasets is again to determine how stable the clustering structure in the data is with three samples of three different sizes. Sampling was also needed as there were constraints on computing power and resources.

These same clients were also extracted from the 6 month and 12 month samples, creating 6 further datasets denoted by 6M_SAMP1, 6M_SAMP2, 6M_SAMP3, 12M_SAMP1, 12M_SAMP2 and 12M_SAMP3. For clarity, all the samples ending in _SAMP1 will contain the same clients, similarly for the datasets ending in _SAMP2 and _SAMP3. This leaves a total of 9 different datasets to investigate and to cluster.

Table 8: Account Balance History Variables (Last 3 months)

Variable Name	Description
Ave_Days_Above_XXX_L3M	Number of days the balance was above RX, divided by the number of days with a positive balance. Calculated for R10, R100, R500, R1000, R2500, R5000, R10000, R20000 and R50000. (9 Variables)
Avg_Dep_Bal_L3M	Average balance of days with a positive balance.
Dep_Bal_Spend_Ratio_L3M	(Maximum balance - Minimum balance) divided by the (Maximum balance), only calculated for days with a positive balance.
AVG_TTSBX_L3M	The average time in days it takes a client to spend X% of the maximum balance. Calculated for 90%, 75%, 50% and 25%. (4 Variables)

Table 9: Outflows and Inflows History Variables (Last 3 months)

Name	Description
Ratio_Out_Above_Xk_L3M	Number of outflows above RX, divided by the total number of outflows. Calculated for R1k, R5k, R10k and R20k. (4 Variables)
Ratio_In_Above_Xk_L3M	Number of inflows above RX, divided by the total number of inflows. Calculated for R1k, R5k, R10k and R20k. (4 Variables)

5.4 Variable Set Description

A total of 120 variables were collected for each of the 9 datasets. In Table 8 the variables collected from the Account Balance History dataset can be found. It is important to note that the discussion here is for the 3 month summary variables, but the same was done for the 6 and 12 month summaries. The letter 'X' acts as a placeholder in the variable names and can be replaced with the different numbers mentioned in the variable descriptions. Account Balance History contributed 15 variables to each of the datasets.

The Outflows History and Inflows History variables were combined in Table 9. Each of these datasets contributed 4 variables to the total collection of variables. In Outflows History and Inflows History there also exist variables for the total amount and number of outflows and inflows. Similar variables, summing the total debit and credit transactions, are also present in the Top 50 Variables dataset and were chosen to rather be collected from there.

Before discussing the variables in the Top 50 Variables set, some definitions for the transaction types are needed here and these can be found in Table 10. This is not an exhaustive list of transaction types, but they are the types that have been identified in the data so far. The Other category should catch all the unidentified transactions. Further work is being done at Bank C to investigate more detailed transaction types, but the scope thereof lies outside this research project.

Table 10: Definitions of Variable Names

Name	**Definition**
Ct	Credit transactions or Inflows into the client's account
Dt	Debit transactions or Outflows from the client's account
USSD	Cellphone Banking transactions, excluding transactions from the Cellphone Banking Application
App	Cellphone Banking Application transactions
INET	Internet Banking transactions
DO	Debit Orders
SO	Stop Orders
CW	Cash Withdrawals
POS	Point of Sale transactions, *i.e.* any transaction where the client's card is "swiped"
CNP	Card-not-present transactions, for example Snapscan, Zapper or Capitec Masterpass
Branch	Transactions performed in a Branch
Elec	Electricity purchase transaction
Air	Airtime purchase transaction
Other	Any transaction that was not classified into the categories above

In Table 11 we can find a summary of the variables coming from the Top 50 Variables set. This dataset contributed 97 variables to our total collection of variables. The reader will also notice that $97 > 50$, and it should be assumed that 50 is just an arbitrary number to represent the, however many, useful variables in the Bank C database.

5.5 Variable Cleaning and Transformations

The variables described above are measured on different scales and some of the variables have very large ranges. This necessitates some variable transformation, and also variable normalisation at a later stage. The steps that were followed to clean the datasets and prepare them for model fitting will be explained for the dataset called 3M_SAMP1.

Initial 5-point summaries of all the variables were constructed and these can be seen in Appendix D for the full 120 variables. From this initial summary it is clear that a log-transformation of the variables that are not on a 0-1 scale is needed. This transformation can draw in the outliers and place them on a more interpretable scale. This will allow visual inspection of the variables to be more valuable. The first concern in this regard is negative values present in the variables Avg_Dep_Bal_L3M and Dep_Bal_Spend_Ratio_L3M, as seen in the output below.

Table 11: Top 50 Variables (Last 3 months)

Name	Description
Val_Z_L3M	Monetary value of Z transactions. Z can be replaced by Ct_Tran, Dt_Tran, USSD, App, INET, DO, SO, CW, POS, CNP, Branch, Elec, Air and Other. (14 Variables)
Ave_Z_L3M	Average (per active months) monetary value of Z transactions. Z can be replaced by Ct, Dt, USSD, App, INET, DO, SO, CW, POS, CNP, Branch, Elec and Air. (13 Variables)
Num_Z_L3M	Count of Z transactions. Z can be replaced by Ct_Tran, Dt_Tran, USSD, App, INET, DO, SO, CW, POS, CNP, Branch, Elec and Air. (13 Variables)
Num_Z_Months_L3M	Average number of months containing Z transactions. Z can be replaced by USSD, App, INET, DO, SO, CW, POS, CNP, Branch, Elec and Air. (11 Variables)
Time_Since_Z_L3M	Average number of months since a Z transaction was performed. Z can be replaced by USSD, App, INET, DO, SO, CW, POS, CNP, Branch, Elec and Air. (11 Variables)
Z_Perc_L3M	Ratio of monetary value of Z transactions to monetary value of all debit transactions. Z can be replaced by USSD, App, INET, DO, SO, CW, POS, CNP, Branch, Elec, Air and Other. (12 Variables)
Z_NumPerc_L3M	Ratio of count of Z transactions to count of all debit transactions. Z can be replaced by USSD, App, INET, DO, SO, CW, POS, CNP, Branch, Elec and Air. (11 Variables)
Ratio_CW_POS_L3M	Ratio of monetary value of Cash Withdrawals to monetary value of POS transactions.
CW_Util_L3M	Ratio of monetary value of Cash Withdrawals to total available funds (Month end Balance + all Credit transactions).
CSWEEP_PX_NumPerc_L3M	Ratio of number of times a Cash Withdrawal of X% of the Balance was made, divided by the number of Cash Withdrawals. Calculated for 90%, 80% and 70%. (3 Variables)
InsufFunds_L3M	Number of times an Insufficient Funds flag was raised.
Naedo_L3M	Number of Non-Authenticated Early Debit Orders (NAEDO).
DO_InsufFunds_L3M	Number of times a Debit Order Insufficient Funds flag was raised.
DO_Dispute_L3M	Number of times a Debit Order was disputed.
Num_Loan_L3M	Number of Term Loans.
Num_Loan_Months_L3M	Average number of months containing Term Loans.
Quality_Banking_L3M	Flag for a Quality Banking client.

```
##        V1                 V2
##  Min.   :   -499   Min.   :-36.0220
##  1st Qu.:    453   1st Qu.:  0.9757
##  Median :    984   Median :  0.9901
##  Mean   :   3518   Mean   :  0.9607
##  3rd Qu.:   2208   3rd Qu.:  0.9950
##  Max.   :5905004   Max.   : 10.5341
```

Even though negative values make sense for these two variables, as a client can have an overdraft on their savings account, it cannot be subjected to a log-transformation. Avg_Dep_Bal_L3M and Dep_Bal_Spend_Ratio_L3M were therefore limited to have a minimum value of zero. Another concern for the log-transformation is the large number of zero values in the data. This problem was solved by adding a constant of one to all the variables and applying the log-transform after that, thus a $\log(x+1)$ transform was used.

Some indicator variables were also created at this point, called Flag_CNP_L3M, Flag_Branch_L3M, Flag_SO_L3M, Flag_INET_L3M, Flag_InsufFunds_L3M, Flag_DO_InsufFunds_L3M and Flag_DO_Dispute_L3M. The reason for this was that the CNP, INET, SO, Branch, InsfFunds, DO_InsufFunds and DO_Dispute variables were very sparse and could lead to skewing of the data. Including an indicator variable, taking the value 1 when these types of transactions were present, and dropping all the other variables in these categories could limit the skewing effect or the noise that they were adding. These added variables increased the variable count to 127.

5.6 Outlier Detection

For outlier detection only the variables Val_Ct_Tran_L3M, Val_Dt_Tran_L3M, Num_Ct_Tran_L3M and Num_Dt_Tran_L3M were considered. The reason for this was that it would be undesirable for a client to be a complete outlier on their total transactions and transaction amounts, but the patterns in the more granular variables should be preserved. For example, if a client spent all of his income through the USSD channel it may be seen as an outlier, but it captures a certain behaviour that could provide valuable insight. The only clients that we would like to exclude are those in a much higher income bracket than the majority and also possible fraudulent transactions. For outlier detection, boxplots were visually inspected and these can be seen in Figure 25(a) to (d).

Based on Figure 25(a) it was decided to limit the value of the log-transform of credit transactions to be less than 14, so that Val_Ct_Tran_L3M < 14. Based on Figure 25(b) it was decided to limit the log-transform of the value of debit transactions to be more than 8 and less than 14, so that $8 <$ Val_Dt_Tran_L3M < 14. Figure 25(c) indicates that a limit of 6.25 should be placed on the log-transform of the number of credit transactions, so that Num_Ct_Tran_L3M < 6.25. Based on Figure 25(d) it was decided to limit the log-transform of the number of debit transactions to 6.5, so that Num_Dt_Tran_L3M < 6.5. After removing 131 of the clients through these limitations, a dataset of size 173 517 still remains. In Appendix E the histograms of all the variables at this point in the data cleaning

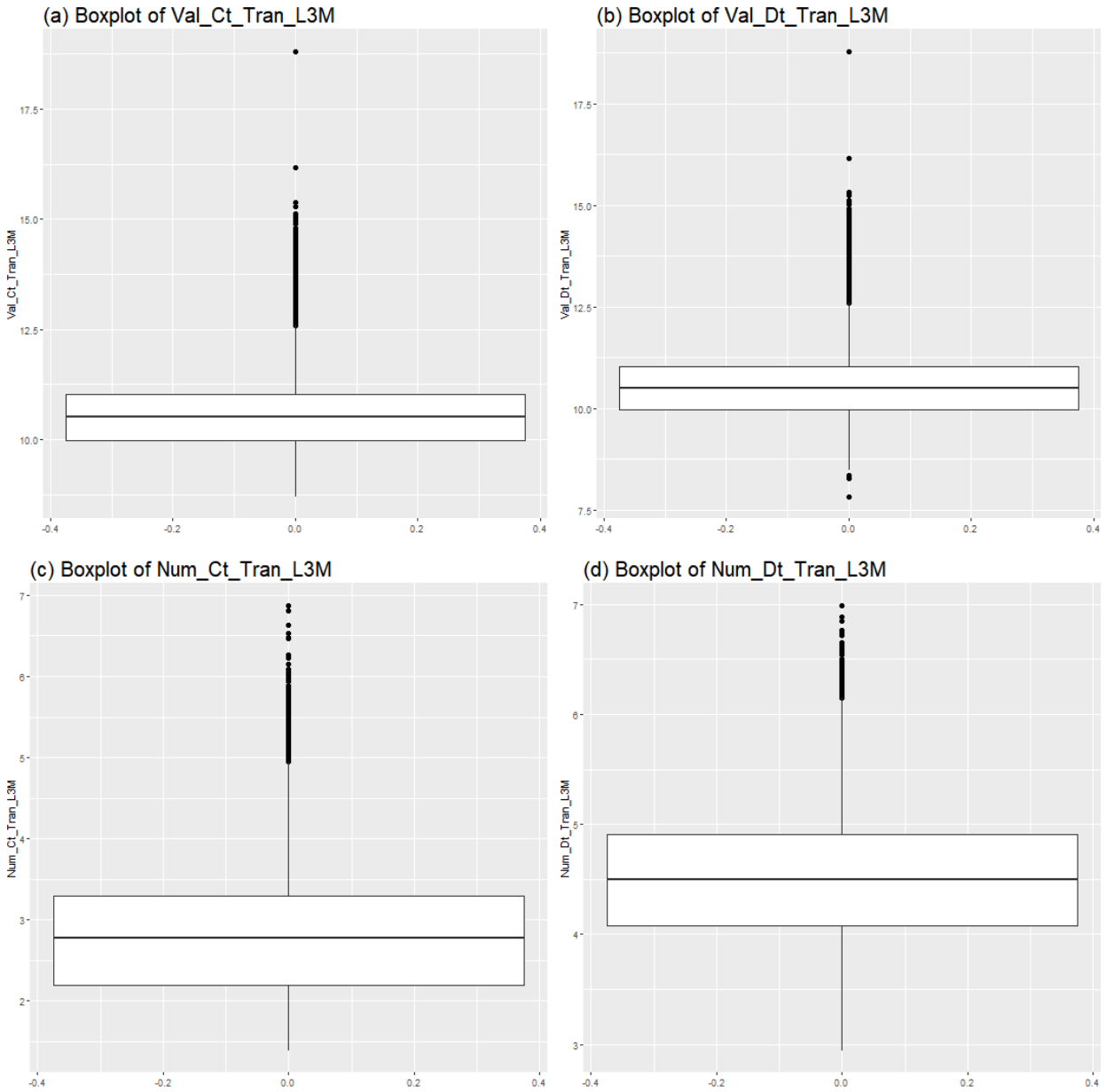


Figure 25: Boxplots of (a) Val_Ct_Tran_L3M, (b) Val_Dt_Tran_L3M, (c) Num_Ct_Tran_L3M and (d) Num_Dt_Tran_L3M

process, before data reduction and normalisation, can be seen.

It is important to note here that limiting the variables to exclude outliers is quite subjective and the decision of how many outliers to exclude is determined by the analyst. By excluding too many extreme values the concept of "cherry-picking" might start to step in and the model can only be applied to a very specific group of clients. The differentiating factors in the data might be lying around the edges of the data, and this valuable information will be removed if too many outliers are removed. It may therefore make sense to investigate different options regarding the omission of outliers in future work.

5.7 Dimension Reduction

The next step in the data cleaning process is to remove all unnecessary variables. Including variables in the dataset that do not serve a purpose may distract from the underlying clustering structure in the data. Firstly, we exclude variables that are very sparse. As a rule of thumb, all variables with more than 75% zeros were investigated to possibly be excluded. Some prior knowledge about the data and knowledge about previous model fitting trials fed into the decision of which variables to exclude.

It was decided to exclude the transaction channels of CNP, SO, Branch and INET, and only keep their indicator variables for the 3 month datasets. For the 6 month datasets only the CNP and Branch variables were excluded and for the 12 month datasets only the CNP channel was excluded. It was also decided to include the negative variables, such as insufficient funds and disputes, only as indicator variables in the 3 month datasets. They were included as the original variables in the 6 and 12 month datasets. It is to be expected that some variables contain more information after 12 months of observation than after 3 months.

The variables that were excluded from the 3 month datasets can be found in the data extract below, together with their percentage zeros. In Appendix F the percentage zero values can be found for the full set of variables. The Val_X_L3M variables are shown for the channels of INET, SO, CNP and Branch and the percentage zeros will be similar for all other variables involving these channels.

```
##                          Vars p_zeros
## 1               Val_INET_L3M   80.43
## 2                 Val_SO_L3M   75.43
## 3                Val_CNP_L3M   94.42
## 4             Val_Branch_L3M   89.86
## 5               Num_INET_L3M   80.43
## 6             InsufFunds_L3M   77.44
## 7          DO_InsufFunds_L3M   72.19
## 8             DO_Dispute_L3M   79.75
## 9  Ave_Days_Above_50000_L3M    95.8
## 10  Ratio_Out_Above_10k_L3M   94.52
## 11  Ratio_Out_Above_20k_L3M   96.89
```

After 34 variables are excluded due to sparsity, 93 variables are still left in the 3 month dataset. We need to consider the Pearson correlation amongst these variables. Only correlations between -0.7 and 0.7 were tolerated as this seemed to be a reasonable cut-off point. Figure 26 contains the correlation plot of all the variables that were excluded from the final sample. Between two variables that were highly correlated with each other, the one with the highest correlation with other variables was excluded first. There were 52 variables excluded in the correlation checking phase of the data cleaning process. This leaves 41 variables that will be used for data modelling. In Figure 27 the correlation plot of the final variable batch can be seen. Comparing Figure 27 to Figure 26 we can clearly see that Figure 27 has much lower correlations amongst the variables and no correlation above 0.7 or below -0.7.

5.8 Data Normalisation and Final Dataset

All of the variables are still measured on different scales and it is important to normalise the data before any model fitting. If model fitting is continued without normalisation the variables measured on the largest scales will have the largest influence on the clustering algorithm. The method of normalisation is chosen as the min-max scaling described in the literature review. The R function used for this purpose can be seen below.

```
Norm_func <- function(x) (x - min(x))/(max(x)-min(x))
```

Next we look at different groups of variables in the final dataset for 3M_SAMP1. The first group we would like to look at is the group of variables containing monetary values, transaction counts and monetary averages. The variables in this group are Val_Other_L3M, Num_Ct_Tran_L3M, Num_Dt_Tran_L3M, Num_CW_L3M, Ave_Ct_L3M and Naedo_L3M. All of these variables were normalised as they did not have measurements on a 0-1 scale. In Figure 28 we can see the histograms for these 6 variables after normalisation.

Val_Other_L3M and Naedo_L3M have quite a large group of zeros each. Apart from the zeros, Val_Other_L3M has a tail to the left and Naedo_L3M has a tail to the right. Num_Ct_Tran_L3M and Ave_Ct_L3M have similar distributions, which is to be expected, as they both relate to inflows on the client's account. Both of these variables have distributions that are skewed to the right. Num_Dt_Tran_L3M seems to be fairly normally distributed, and similarly Num_CW_L3M seems to be normally distributed, but with a slight tail to the left.

The second group of variables to investigate contains information on the monthly usage of the different transactional channels. The variables in this group are Num_DO_Months_L3M, Time_Since_USSD_L3M, Time_Since_CW_L3M, Time_Since_POS_L3M, Time_Since_Elec_L3M, Time_Since_Air_L3M and Num_Loan_Months_L3M. These variables were already expressed as a ratio to the total number of months, in this case 3, and did not need normalisation. Figure 29 shows the histograms for these 7 variables.

It is important to note that the Num_X_Months_L3M variables and the Time_Since_X_L3M variables are inversely related for the same value of X. This means that the histograms will be skewed in opposite directions, but emphasise the same kind of behviour. Most of

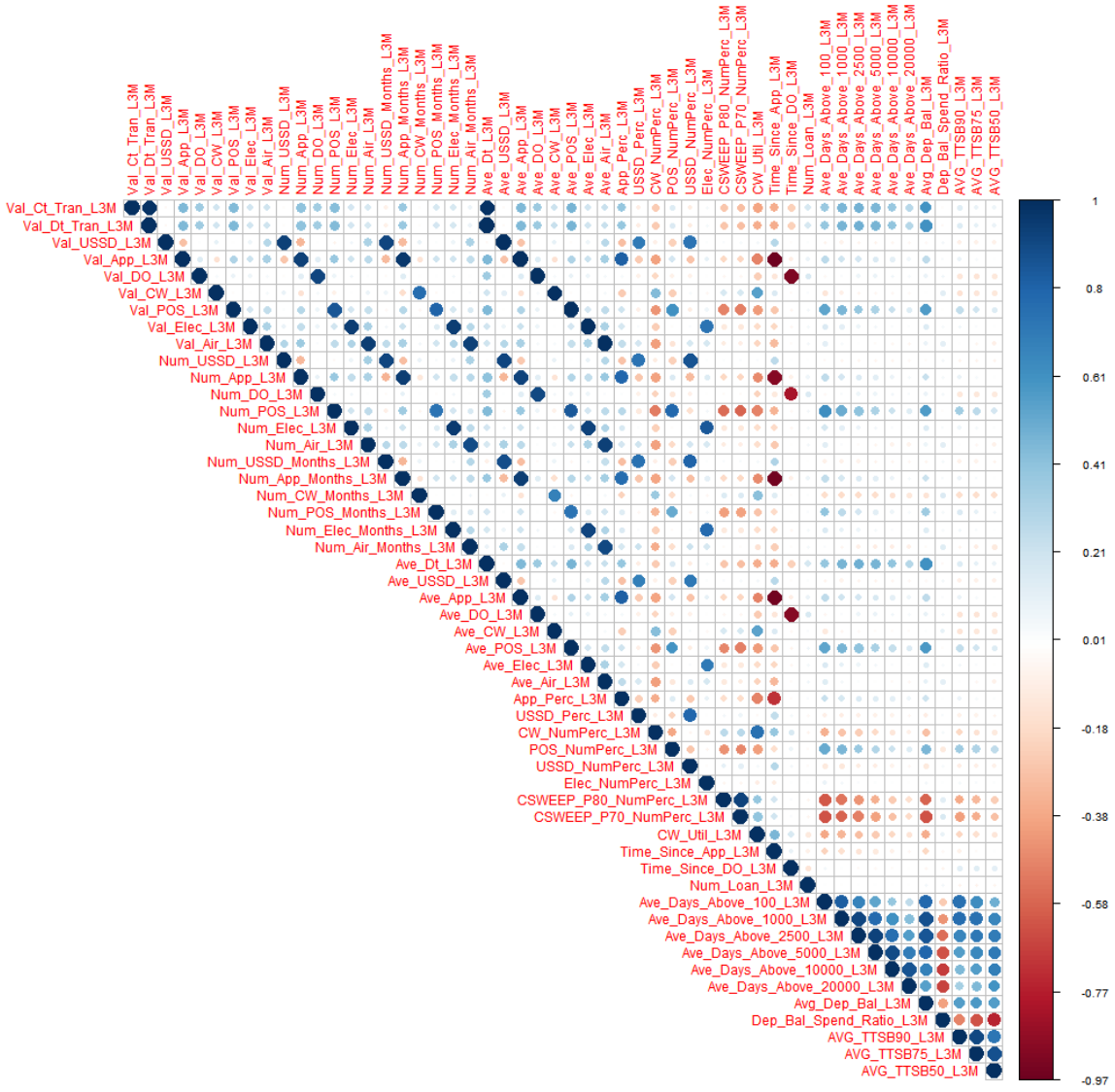


Figure 26: Correlation plot of 52 variables to be excluded

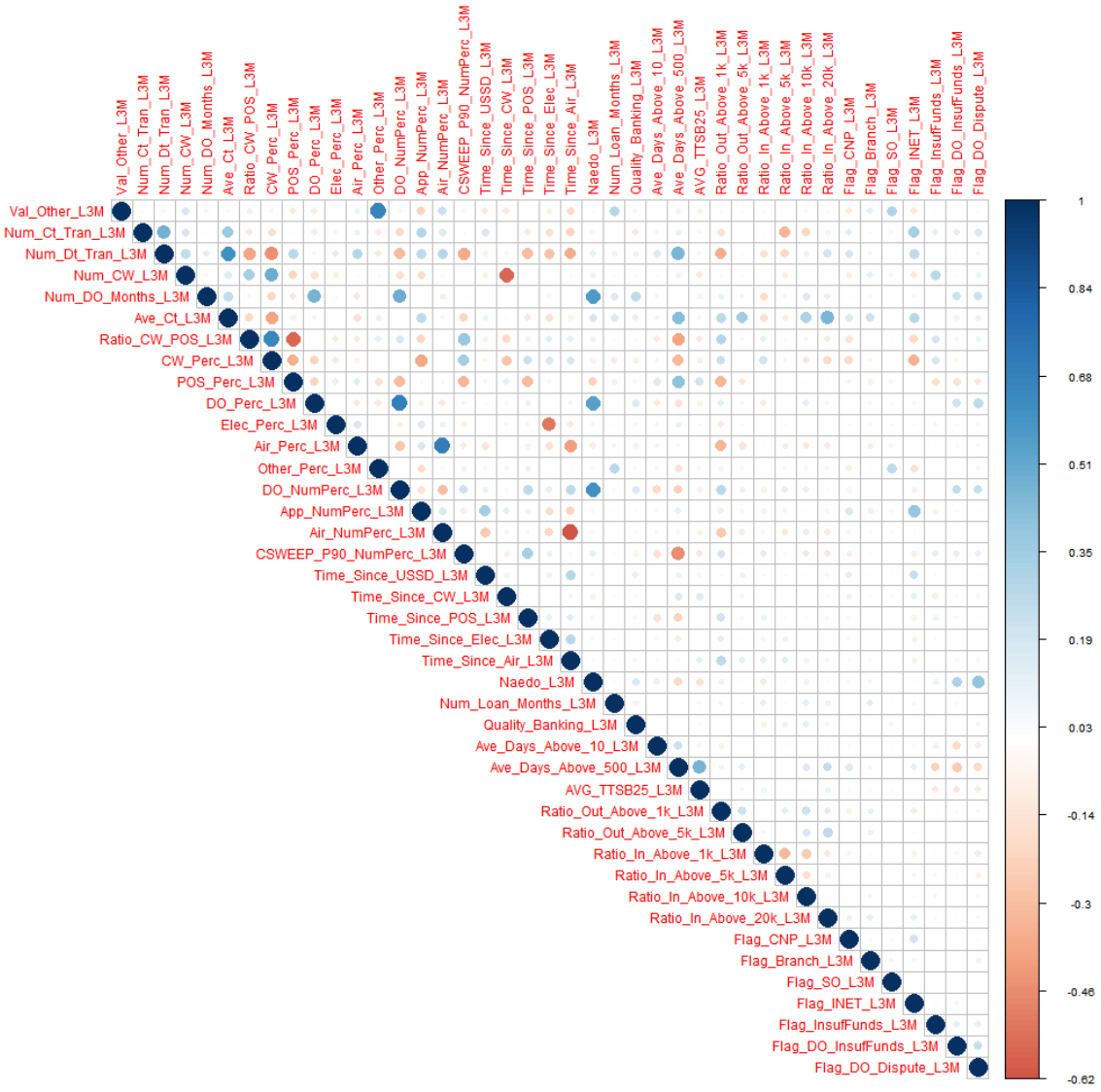


Figure 27: Correlation plot of 41 variables to keep

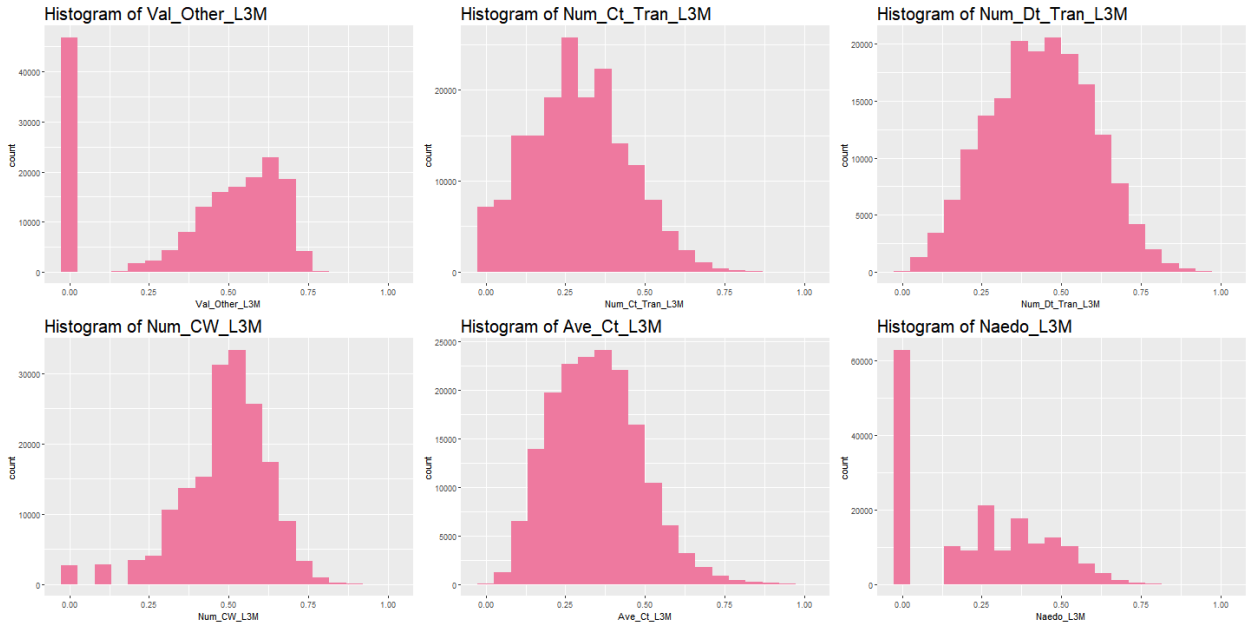


Figure 28: Histograms for the monetary values, transaction counts and monetary averages variables

the 7 variables resembled indicator variables, with large weights on the 0 and 1 values and small weights on the interior values. Num_DO_Months_L3M indicates that most clients in the sample have monthly debit orders linked to their savings accounts. Similarly, Time_Since_CW_L3M and Time_Since_POS_L3M show that most clients make cash withdrawals and POS transactions every month. Time_Since_USSD_L3M indicates that a large portion of the client sample do not make use of USSD transactions and a second smaller portion uses the transaction channel monthly. The same can be said for Time_Since_Elec_L3M: two groups in the data arise, a group that never buys electricity and a group that buys electricity at least monthly. Time_Since_Air_L3M has a similar pattern, but inversely to Time_Since_USSD_L3M and Time_Since_Elec_L3M. The two groups that emerge here are the monthly airtime buyers and a much smaller group that almost never buys airtime. According to Num_Loan_Months_L3M, most clients in the sample did not have any Term Loans.

The third group of variables we would like to look at is the group of variables containing ratios and percentages. The variables in this group are Ratio_CW_POS_L3M, CW_Perc_L3M, POS_Perc_L3M, DO_Perc_L3M, Elec_Perc_L3M, Air_Perc_L3M, Other_Perc_L3M, DO_NumPerc_L3M, App_NumPerc_L3M, Air_NumPerc_L3M and CSWEEP_P90_NumPerc_L3M. The only variable in this group that needed normalisation was Ratio_CW_POS_L3M. In Figure 30 we can see the histograms for the 11 variables in this group.

All of the distributions are clearly skewed to the right with tails of varying lengths and thickness. This is to be expected for all the percentage variables as we would not expect all of a clients' transactions to be spent through one channel only. Airtime and electricity make

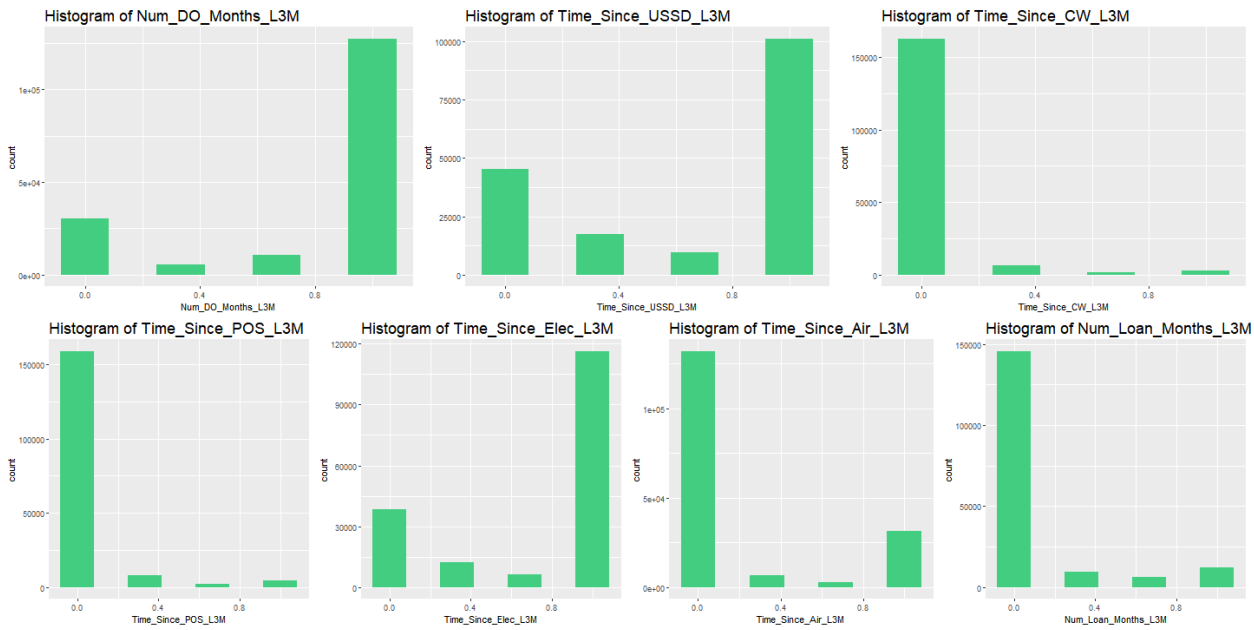


Figure 29: Histograms for the variables containing information on monthly frequency of the different channel usage

out a very small portion of the expenditure of most of the clients, but airtime makes out a larger portion of the number of transactions. There are also a large number of zero values present in this batch of variables. Also, as expected, cash withdrawals, POS transactions and debit orders make out larger portions of the total expenditure. These are more traditional channels compared to USSD or App.

The fourth group of variables contains information on the sizes of the inflows and outflows. The variables in this group are Ave_Days_Above_10_L3M, Ave_Days_Above_500_L3M, AVG_TTSB25_L3M, Ratio_Out_Above_1k_L3M, Ratio_Out_Above_5k_L3M, Ratio_In_Above_1k_L3M, Ratio_In_Above_5k_L3M, Ratio_In_Above_10k_L3M and Ratio_In_Above_20k_L3M. All of these variables had to be normalised to place them on a 0-1 scale. Figure 31 contains the histograms for these 9 variables.

Most of the variables are distributed skew to the right. Ave_Days_Above_10_L3M is skewed to the left, meaning that most clients have a balance above R10, on most days. Ave_Days_Above_500_L3M has a peak to the right as well as the left; this means that there is a group of clients who always have a balance above R500, but also a large group of clients who have a balance above R500 only 25% of the time. Overall it seems that larger amounts are coming into the clients' accounts than leaving the accounts. This deduction can be made because the Ratio_In_Above_Xk_L3M variables are more evenly distributed to the right than the Ratio_Out_Above_Xk_L3M variables. Some of these variables are very skew and will almost act as indicator variables in the clustering process. This is still an important task as it can identify small pockets of clients around the edges of the data.

The fifth group of variables we would like to look at is the group of variables containing indicator variables. The variables in this group are Quality_Banking_L3M, Flag_CNP_L3M,

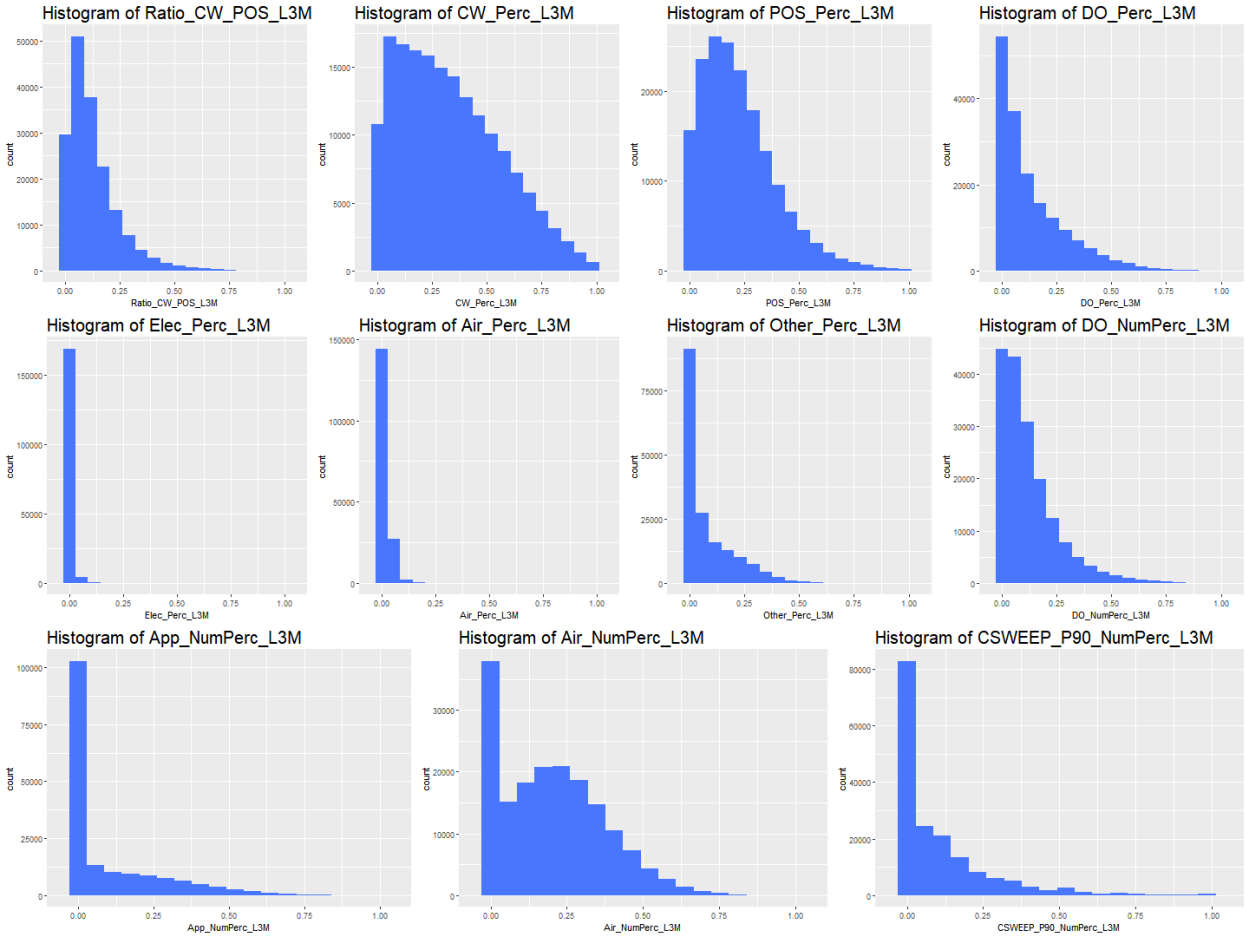


Figure 30: Histograms for the percentage and ratio variables

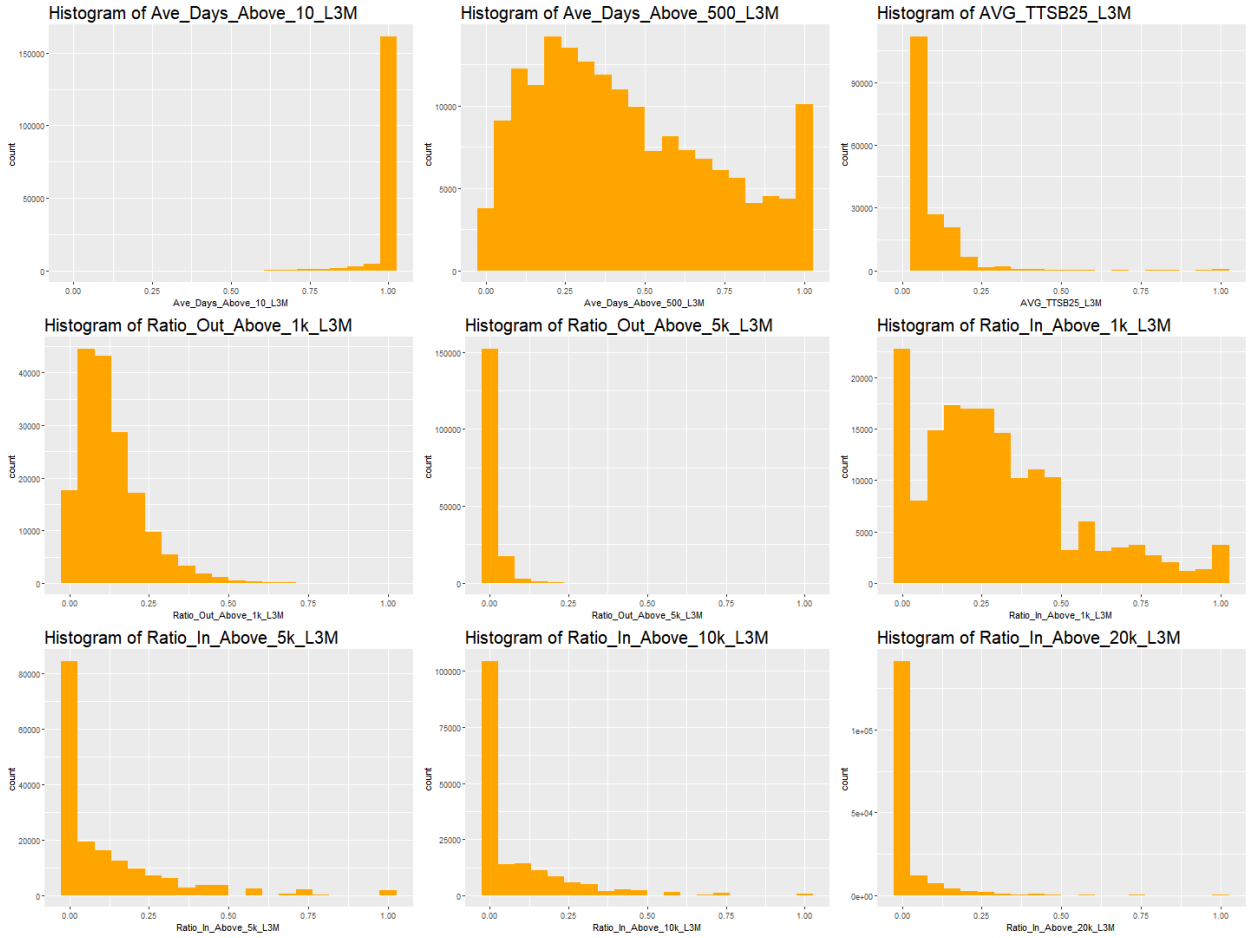


Figure 31: Histograms for the variables containing information on sizes of inflows and outflows

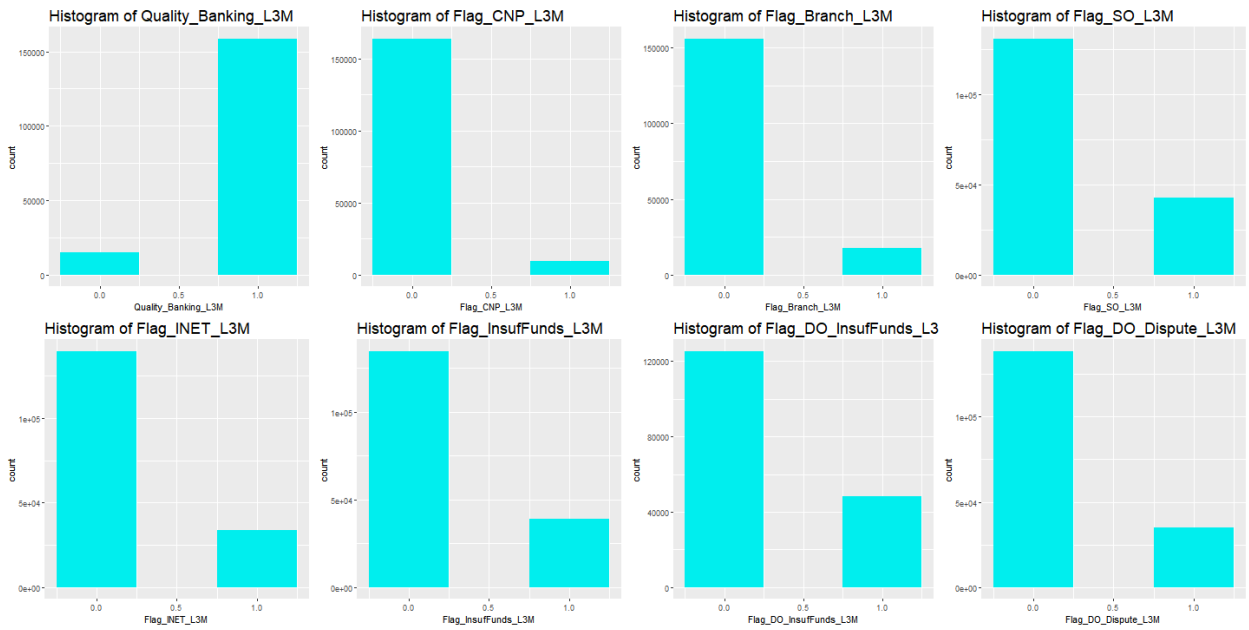


Figure 32: Histograms for the indicator variables

Flag_Branch_L3M, Flag_SO_L3M, Flag_INET_L3M, Flag_InsufFunds_L3M, Flag_DO_InsufFunds_L3M and Flag_DO_Dispute_L3M. All of these variables, except Quality_Banking_L3M, were created to compensate for variables that were otherwise too sparse to have an influence on the clustering of the data. These variables did not need normalisation as they are inherently on a 0-1 scale. In Figure 32 we can see the histograms of these 8 variables.

As with the sparse variables in the previous batch of variables, the indicators can help identify subtle differences between clients and can play an important role in clustering of the data. Quality_Banking_L3M indicates which clients Bank C sees as good quality clients, which in this case comprises of almost the whole sample. This variable shifts more in the 6 month and 12 month summary datasets. Flag_CNP_L3M also indicates a very small group of clients that utilise CNP transactions and might be seen as a proxy for financial sophistication or being versed in the newest technology. Stop orders and internet banking are slightly more utilised. The negative variables of insufficient fund and dispute indicators will also play a role in clustering of the clients, as they can indicate propensity to delinquency.

5.9 Summary

In this chapter we focused on the steps in the data cleaning process. The reasoning behind certain decisions pertaining to variable exclusion was also discussed. A similar method was followed for each of the 9 datasets and this process resulted in 9 clean and normalised datasets to be used in SOM fitting.

In the next chapter a SOM will be applied to each of the datasets to determine if SOMs

are useful as a dimension reduction technique in practice. We will also use Ward linkage to cluster the SOM prototypes to investigate the underlying clustering structure in the data. The cluster results from the 9 datasets will be compared to see if consistent client groups emerged across the datasets.

CHAPTER 6: DATA MODELLING

6 Data Modelling

6.1 Introduction

In the previous chapter we prepared the real-world Bank C data for SOM fitting by cleaning and transforming it to a usable format. The idea is to use the SOM as a dimension reduction technique and to fit Ward linkage clustering on the prototypes output by the SOM. The Ward linkage can help us determine if certain clustering behaviours for clients can be found in the data. There are 9 datasets available, three sets of varying size for the 3 month, 6 month and 12 month data summaries. We would like to find clusters that remain stable across the different datasets. Meaning that we would like the clusters to hold for varying cluster sizes, as well as varying observation windows.

6.2 Data Summary and Visualisation

In Table 12 we can find a summary of the 9 cleaned and normalised datasets. The available information on each dataset includes the original percentage of data sampled, the current sample size and the number of variables included. The datasets that contained the same amount of clients, for example all the SAMP1 datasets, now have slightly varying sample sizes because of outliers being removed from each dataset. The number of variables also differ slightly, with more variables being included for the longer observation windows. This makes sense, because some variables might have become stronger or the distributions might have shifted with more information available to them.

Before model fitting we inspect the data visually to determine if some information can already be extracted. As before, we plot the data observations on the first two PCs, because the data is high-dimensional and cannot be visualised otherwise. In contrast to the simulation

Table 12: Summary of the 9 Datasets

Dataset Name	Perc of Data	Sample Size	Num of Vars
3M_SAMP1	10	173517	41
3M_SAMP2	6	104439	38
3M_SAMP3	3	52120	39
6M_SAMP1	10	173585	43
6M_SAMP2	6	104395	44
6M_SAMP3	3	52107	41
12M_SAMP1	10	173588	49
12M_SAMP2	6	104397	49
12M_SAMP3	3	52102	48

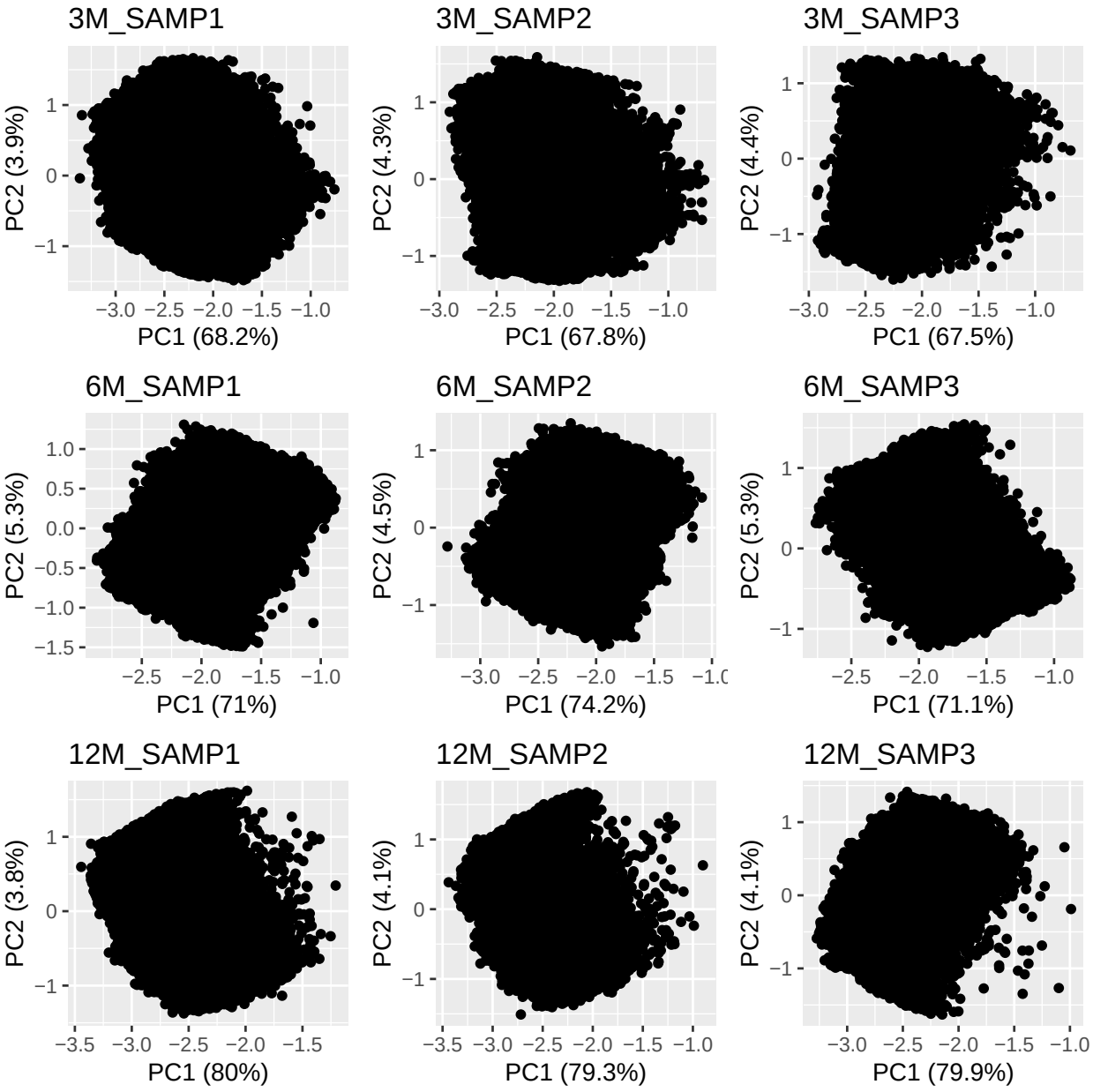


Figure 33: Normalised data plot on first two PCs for all 9 datasets

Table 13: Summary of PCs and SOM dimensions

Dataset Name	Perc Var Explained	Ratio of Eigen	SOM rows	SOM columns	Epochs
3M_SAMP1	72.1%	17.66	159	9	50
3M_SAMP2	72.1%	15.59	125	8	75
3M_SAMP3	71.9%	15.35	93	6	100
6M_SAMP1	76.3%	13.5	136	10	50
6M_SAMP2	78.7%	16.63	134	8	75
6M_SAMP3	76.4%	13.33	94	7	100
12M_SAMP1	83.8%	21.24	170	8	50
12M_SAMP2	83.4%	19.49	156	8	75
12M_SAMP3	84%	19.52	118	6	100

study chapter, we do not know the underlying clustering structure in the data and cannot colour-code the observations according to their true clusters.

In Figure 33 we find the plots of the normalised data observations on the first two PCs for all the datasets. No clear clustering structure is present in the data and the data is very compact. The 12 month summary sample has more observations that appear as outliers, even after outlier removal. It is difficult to make assumptions about what these points represent as the data we are looking at is high-dimensional. It is impossible to know what we should expect to see in the data plots. In the Simulation Study, clear clusters were visible in the data plots on the first two PCs. It is important to note the contrast between the simulated data and the real-world data. The simulated data had a Gaussian distribution and only 10 dimensions. In Figure 33 we have at least 38 dimensions and we do not know the combined distribution of the variables.

6.3 SOM fitting

We now fit a SOM to each of the datasets described above. We will use the Batch SOM with random initialisation and a hexagonal map topology, as was used in the simulation study. The ratio of the eigenvalues related to the first two PCs will be used to decide on the dimensions of the SOM prototype grids. In Table 13 a summary is given on the percentage variance explained by the first two PCs for each dataset. In this table the ratio of the first two eigenvalues can also be found, together with dimensions decided on for the prototype grids and the number of data epochs used in the Batch SOM algorithm.

The percentage variance explained by the first two PCs in Table 13 seems to be increasing from the 3 month to the 12 month summary data. More data epochs were used for smaller sample sizes to make sure that the prototype maps converged. A trial-and-error process was followed to determine the correct amount of epochs, but the values in Table 13 seem to have been sufficient. The dimensions of the SOM grids were decided on so that each prototype would receive between 70 and 150 of the observations, on average. This was also as a result of a trial-and-error process.

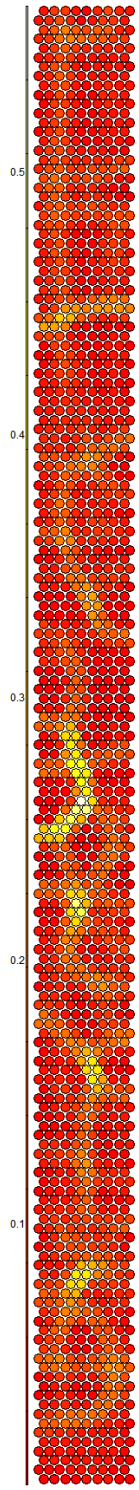


Figure 34: U-Matrix of the SOM of 3M_SAMP1

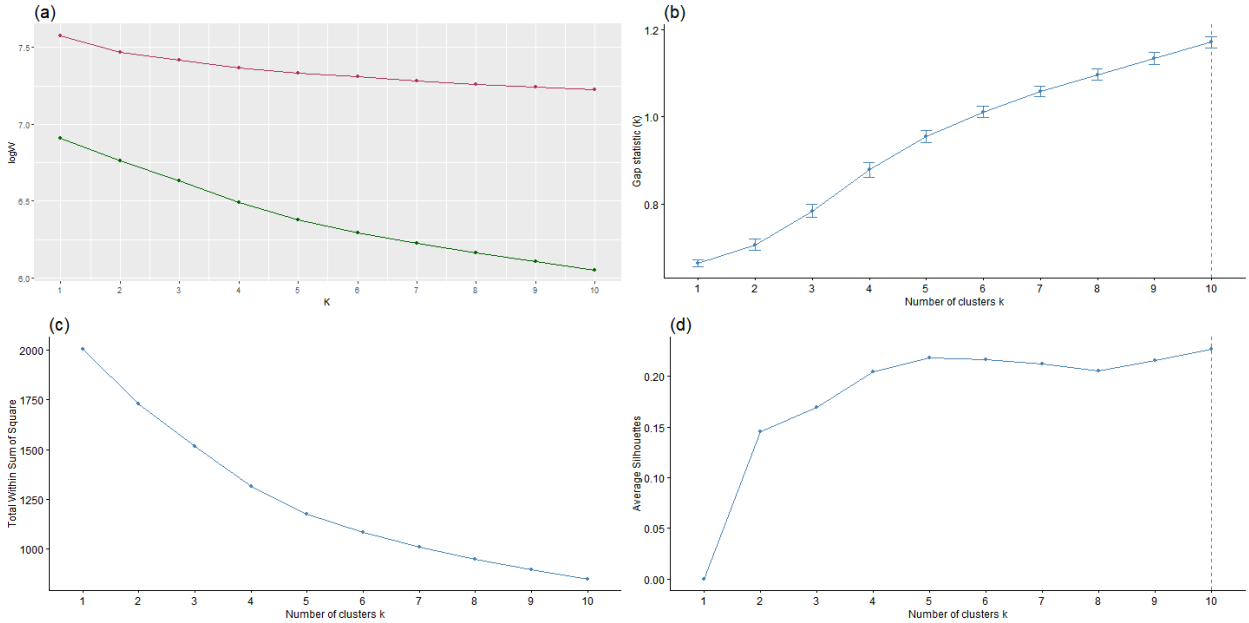


Figure 35: Gap statistic, elbow method and average silhouette method plot for the 3M_SAMP1 dataset

In Figure 34 the U-Matrix for the SOM map on dataset 3M_SAMP1 can be found. The distances between the prototypes form an interesting pattern and the lighter areas are supposed to indicate borders between clusters. Counting the darker areas is somewhat subjective, but it seems that around 25 areas can be counted. As they are not all lying equally far away from each other, there might be the possibility that two of these darker regions could still belong to the same cluster. Therefore some further techniques on choosing the optimal number of clusters will be implemented. In Appendix G, Figure 58 to Figure 60, the U-Matrices of the remaining 8 datasets is given. These Figures are a good representation of the output of the SOMs, even if it is almost impossible to decide on the appropriate number of clusters by visual inspection only.

6.4 Deciding on the Appropriate Values for K

For this purpose we will use the methods described and tested before, the gap statistic, the elbow method and the average silhouette method. We will also use the function *NbClust* to calculate 19 more possible values for K. A majority vote will have to be taken between all the outcomes. We will limit the maximum number of clusters to 10, for the sake of interpretability, even though this might return clusters that are still heterogeneous. It would be desirable for the number of clusters to be below 10 as a different treatment might have to be applied to each group by Bank C. The larger the number of clusters, the more difficult it becomes to determine which groups need a specific treatment and which groups should not be interfered with. It is recommended that in further studies, larger numbers for K are explored.

Table 14: Summary of Possible Values of K

Dataset Name	First Choice K	Second Choice K	Third Choice K
3M_SAMP1	5	10	4
3M_SAMP2	7	10	2
3M_SAMP3	10	3	7
6M_SAMP1	3	10	2
6M_SAMP2	3	10	2
6M_SAMP3	10	3	2
12M_SAMP1	10	3	2
12M_SAMP2	3	10	2
12M_SAMP3	10	2	4

In Figure 35(a) and (b) we find the plots of the gap statistic. These plots are very inconclusive as the gap statistic chose the largest option available for K. This is always a cause for suspicion as it indicates that a value larger than the maximum of K provided might be preferred. We can also see that the gap statistic plot looks like it is still increasing and not close to a turning point. An important note is that the gap statistic plot did not choose $K = 1$, so there is underlying clustering structure present in the data. The elbow method in Figure 35(c) is not helpful either. There is no clear "kink" in the plot, but there might be a subtle turning point at $K = 5$. In Figure 35(d) the average silhouette values for the different values of K can be seen and the maximum number of clusters were also chosen. There is an internal turning point at $K = 5$, which could indicate that 5 is also a possibility for K. The gap statistic, elbow method and average silhouette plots for the remaining datasets can be seen in Appendix H, Figure 61 to Figure 68.

In Table 14 a summary of the possible values for K for the different datasets can be found. There are some recurring numbers of clusters between the datasets, but it is not as similar as we would have hoped. The value $K = 10$ appears in each of the datasets' top three appropriate values for K, confirming that larger values of K need to be investigated. It is also important to remember that the datasets containing SAMP1 in the name consist of the same clients, similarly for SAMP2 and SAMP3 respectively. Therefore we would like to fit the same amount of clusters to the samples containing the same clients, to ensure that the results are comparable.

It was decided that $K = 5$ will be chosen for all the cluster modelling to follow. The value 5 only appears once in Table 14, whereas the values 2, 3 and 4 appeared often, suggesting that a low number of clusters may also be appropriate. The value $K = 5$ is a good halfway mark between $K = 2$ and $K = 10$, as well as returning an interpretable number of clusters. It also seemed to be an appropriate cutpoint in the dendrograms that will follow in the next section. Heterogeneous clusters can be identified to be investigated further.

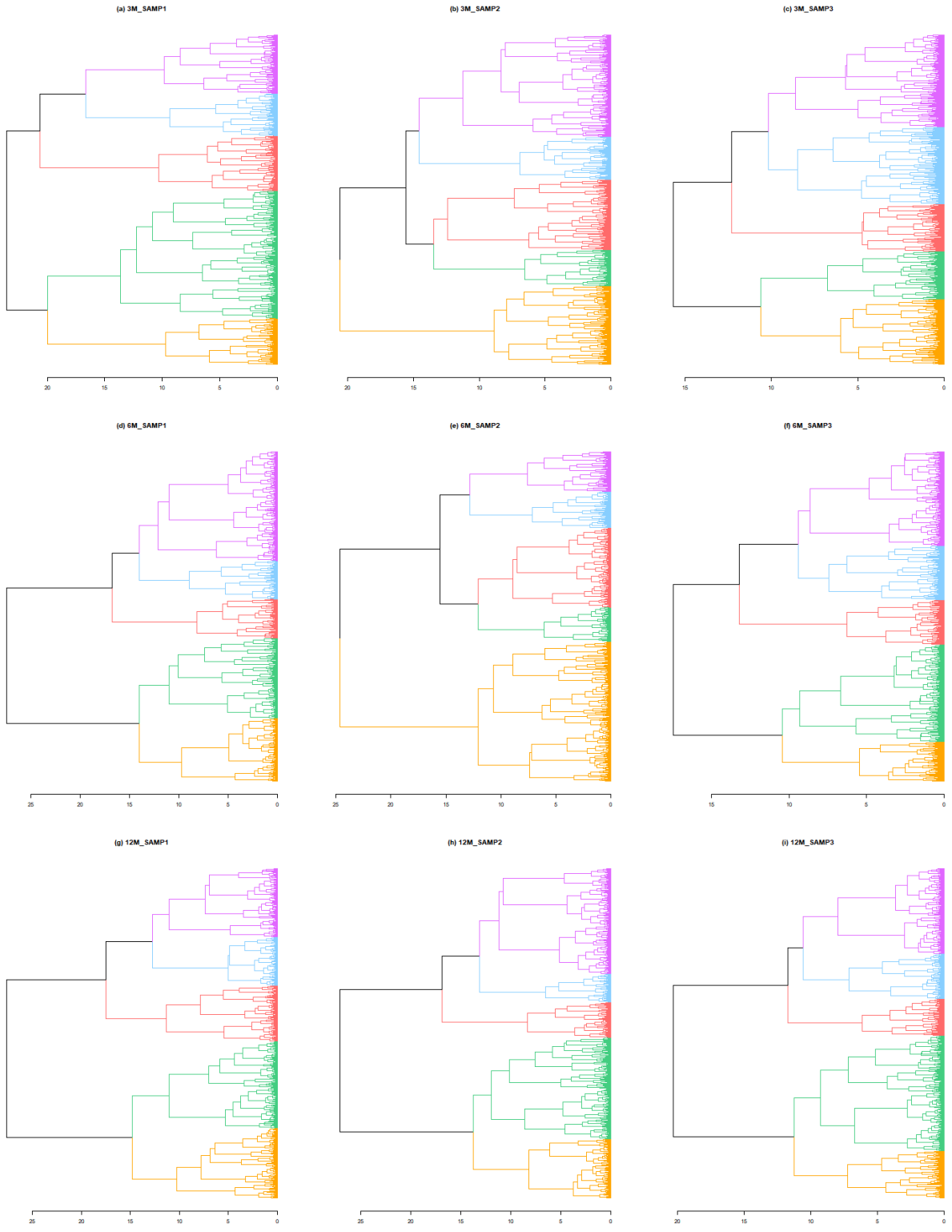


Figure 36: Dendrograms for (a) 3M_SAMP1, (b) 3M_SAMP2, (c) 3M_SAMP3, (d) 6M_SAMP1, (e) 6M_SAMP2, (f) 6M_SAMP3, (g) 12M_SAMP1, (h) 12M_SAMP2 and (i) 12M_SAMP3

6.5 Fitting Ward linkage clustering on the SOMs

Now that we have decided on the number of clusters, we fit Ward linkage clustering on top of each of the SOMs. We will first look at the dendrograms for the 9 datasets and these are shown in Figure 36(a) to (i). The clusters are more balanced in some dendrograms than others, for example Figure 36(c) and (g). We do not know how balanced the true clusters are and it is unlikely that they are all of the same size. Clusters around the edges of the data are expected to be smaller than interior clusters as they might contain more extreme cases of client behaviour. For the 12 month samples, Figure 36(g) to (i), the dendrograms look quite similar, indicating that the client distribution might have stabilised after 12 months of variable observation. The dendrograms for the 6 month samples, Figure 36(d) to (f), and the 3 month samples, Figure 36(a) to (c), are more varied and the cluster sizes, together with the clients they contain, change from sample to sample.

We would also like to visually inspect the colour-coded cluster plots on the two largest PCs for all 9 samples, together with the silhouette plots of the prototypes. These plots will give an insight into how compact and well-separated the formed clusters are. In Figure 37(a) to (f) the colour-coded clusters and the silhouette plots for the three 3 month datasets can be found. Figure 38(a) to (f) summarise the clusterings for the three 6 month datasets and Figure 39(a) to (f) provide a summary for the output of the three 12 month datasets.

A general comment, that is clear from the colour-coded cluster plots, is that a fuzzy clustering method might have been more appropriate for these datasets. The clusters are extremely overlapping and this might mean that clients do not belong to only one cluster. There might be a few possible behaviour categories each client could fall into, with a certain probability attached to each one.

Even though there is quite a lot of overlap, some clustering structure can be seen in the colour-coded plots. For example, in Figure 37(a) the observations of cluster 5 are located around the bottom-left of the dataset and the observations for cluster 4 are situated at the top. Some of the clusters group together in a more compact way than others and also have higher average silhouette values as a result, for example, cluster 5 in Figure 37(b) or cluster 5 in Figure 37(f). The average silhouette values for all the data clusterings are quite low, between 0.14 and 0.23. This might be due to the shapes of the clusters not being compact and spherical, and the fact that the clusters are also not well separated. It is important to note that the silhouette plots take the prototypes as input values and not the cluster labels mapped onto the original dataset. The reason for this is that silhouette calculation uses a dissimilarity matrix and our datasets are too large for dissimilarity matrix calculations.

6.6 Client Behaviour Profiles

To extract the most information from the clusters formed in the previous section we would like to develop client behaviour profiles. The output from the 3M_SAMP1 sample will be investigated in detail, as before, and will be used as a baseline for comparison. We will use histograms and distribution plots of the variable type groups, described in the Data Cleaning

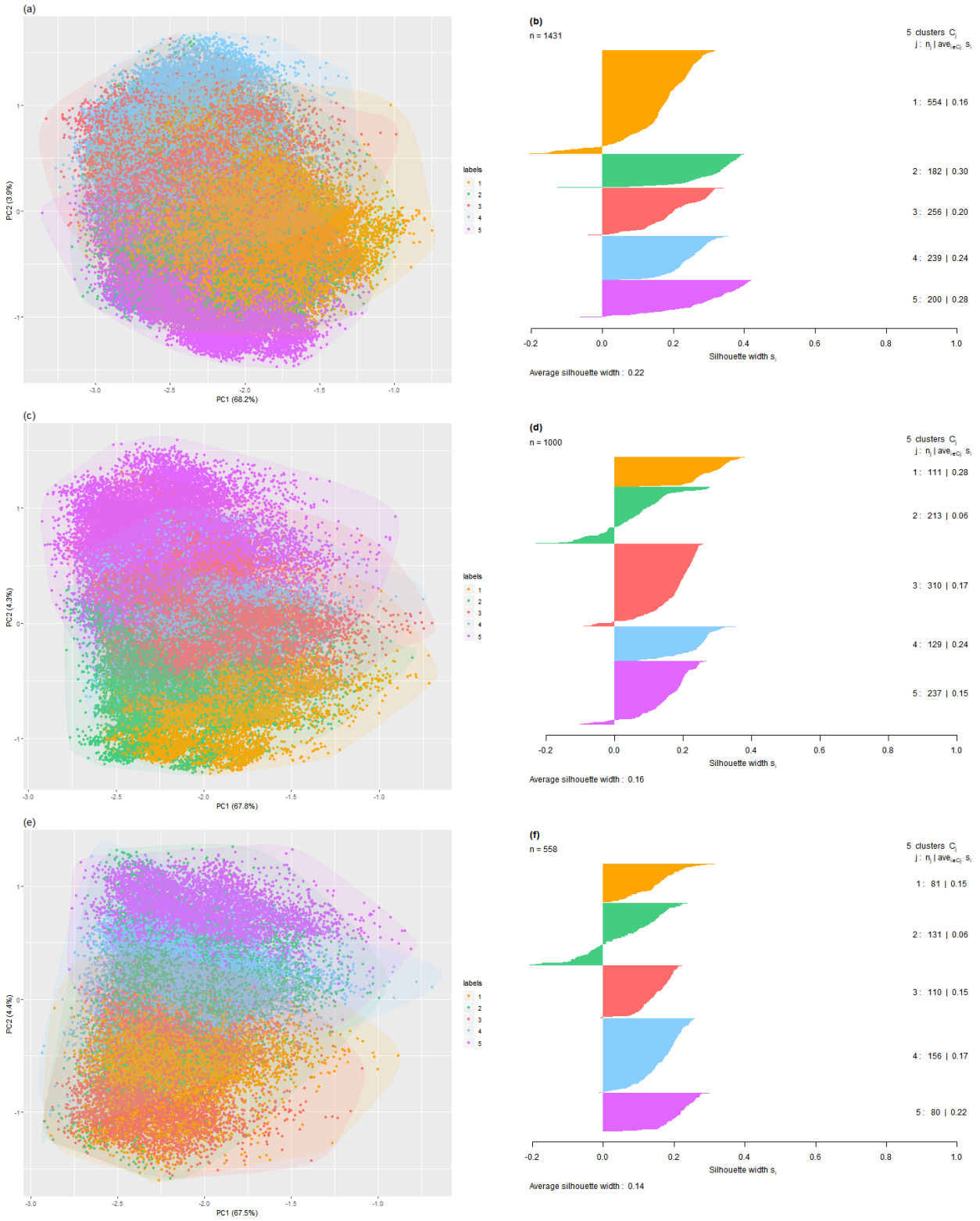


Figure 37: Colour-coded cluster plots on the first two PCs for (a) 3M_SAMP1, (c) 3M_SAMP2, (e) 3M_SAMP3 and Silhoutte plots for (b) 3M_SAMP1, (d) 3M_SAMP2, (f) 3M_SAMP3

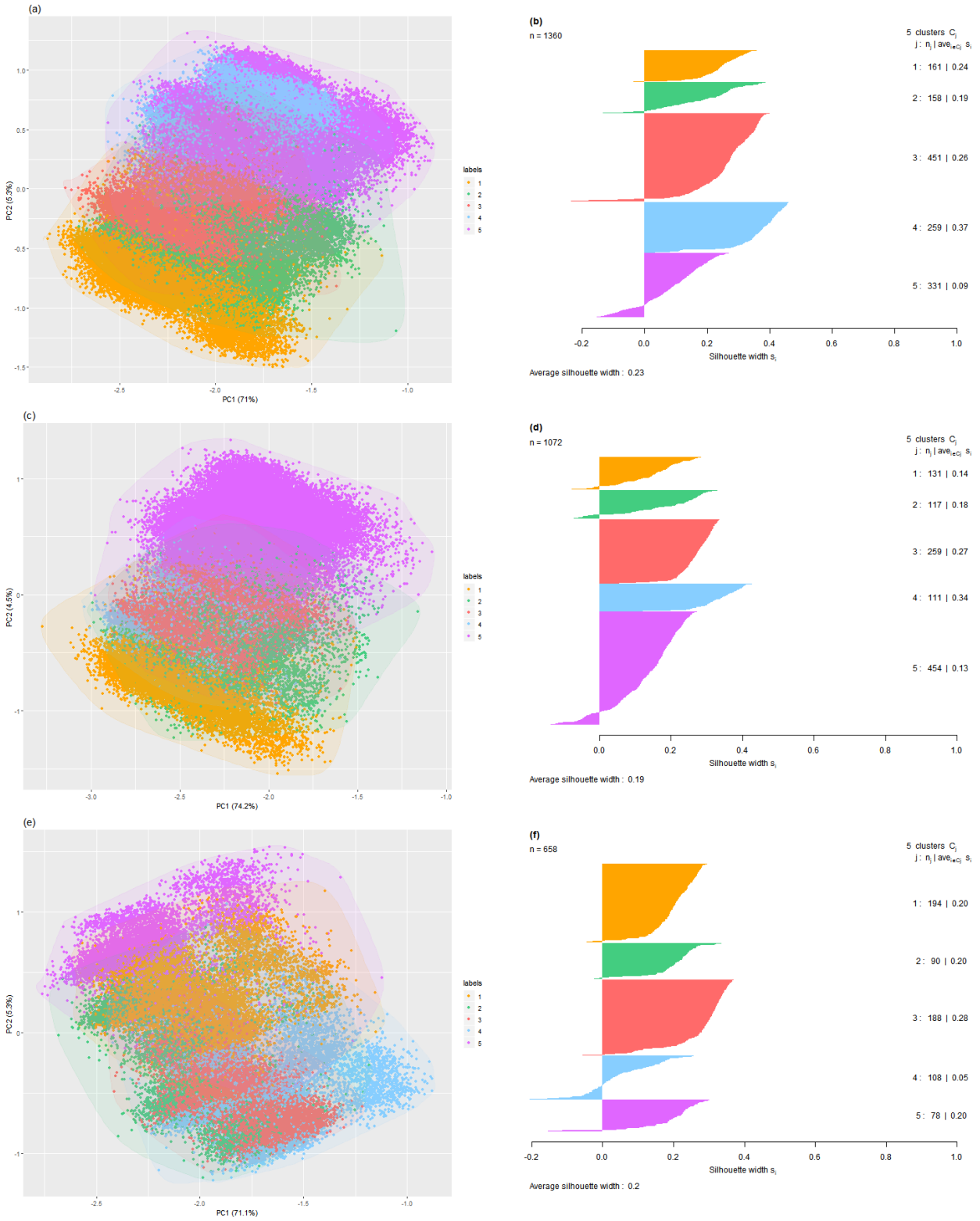


Figure 38: Colour-coded cluster plots on the first two PCs for (a) 6M_SAMP1, (c) 6M_SAMP2, (e) 6M_SAMP3 and Silhoutte plots for (b) 6M_SAMP1, (d) 6M_SAMP2, (f) 6M_SAMP3

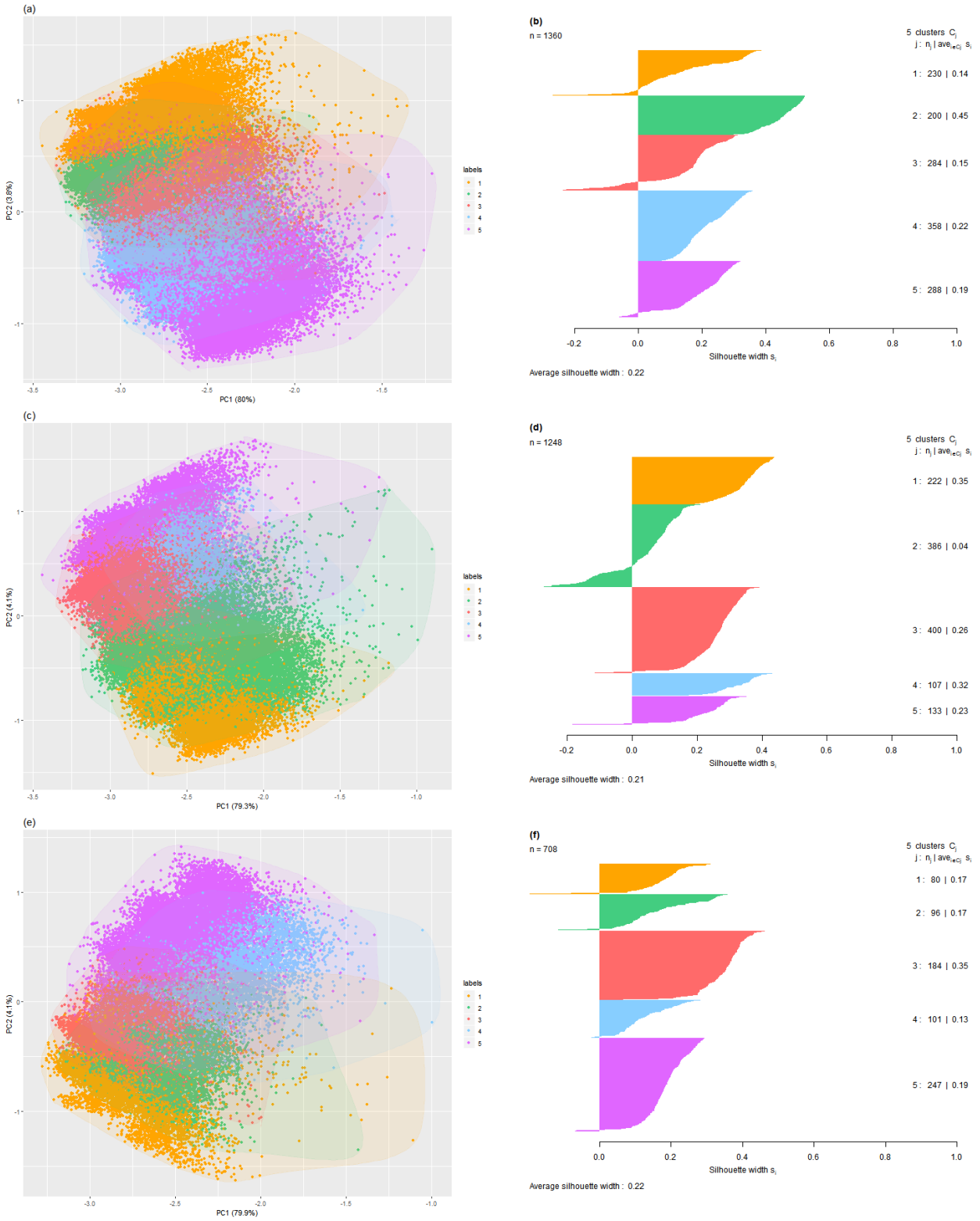


Figure 39: Colour-coded cluster plots on the first two PCs for (a) 12M_SAMP1, (c) 12M_SAMP2, (e) 12M_SAMP3 and Silhoutte plots for (b) 12M_SAMP1, (d) 12M_SAMP2, (f) 12M_SAMP3

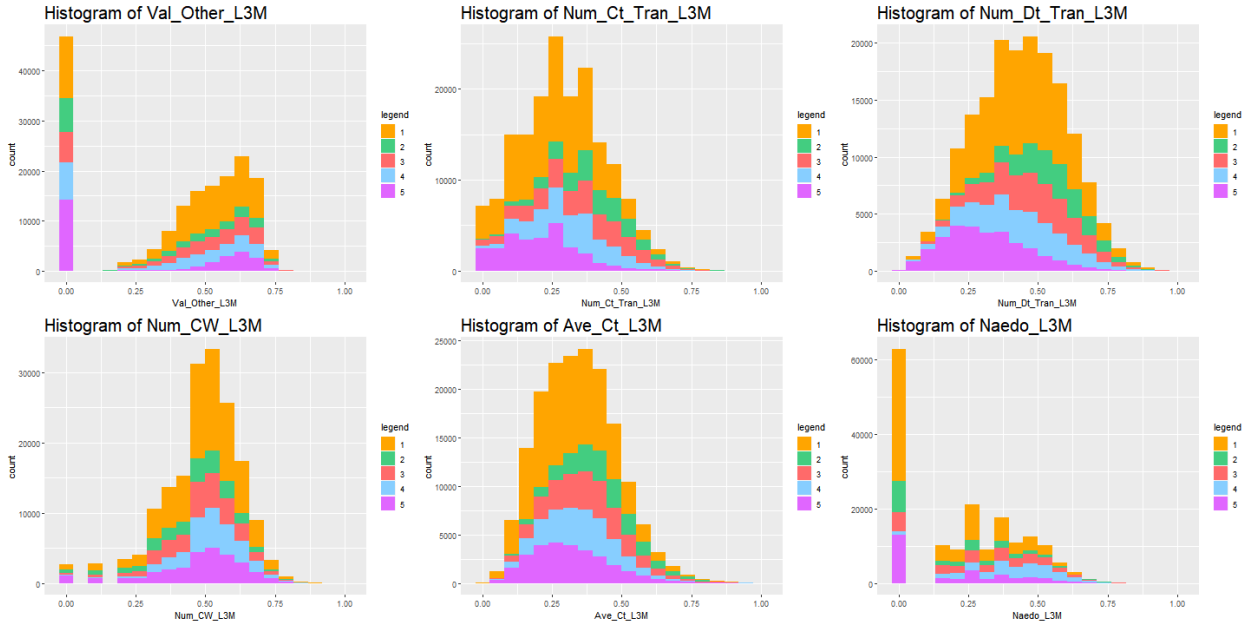


Figure 40: Histograms for the monetary values, transaction counts and monetary averages variables, colour-coded according to clusters for 3M_SAMP1

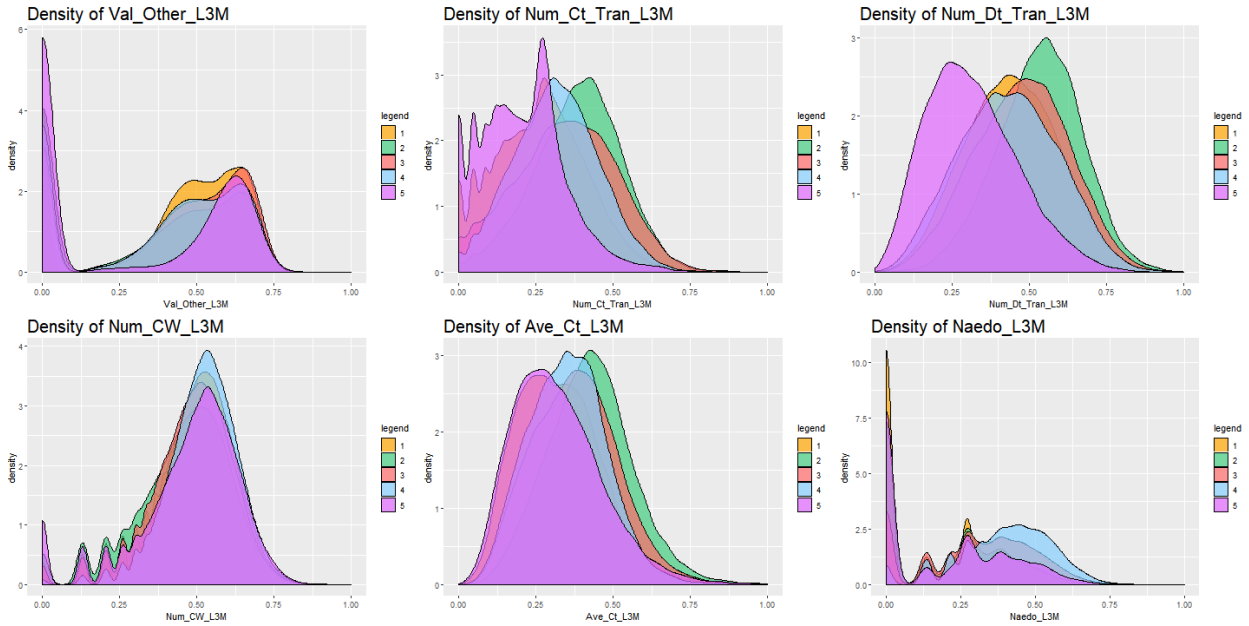


Figure 41: Density plots for the monetary values, transaction counts and monetary averages variables, colour-coded according to clusters for 3M_SAMP1

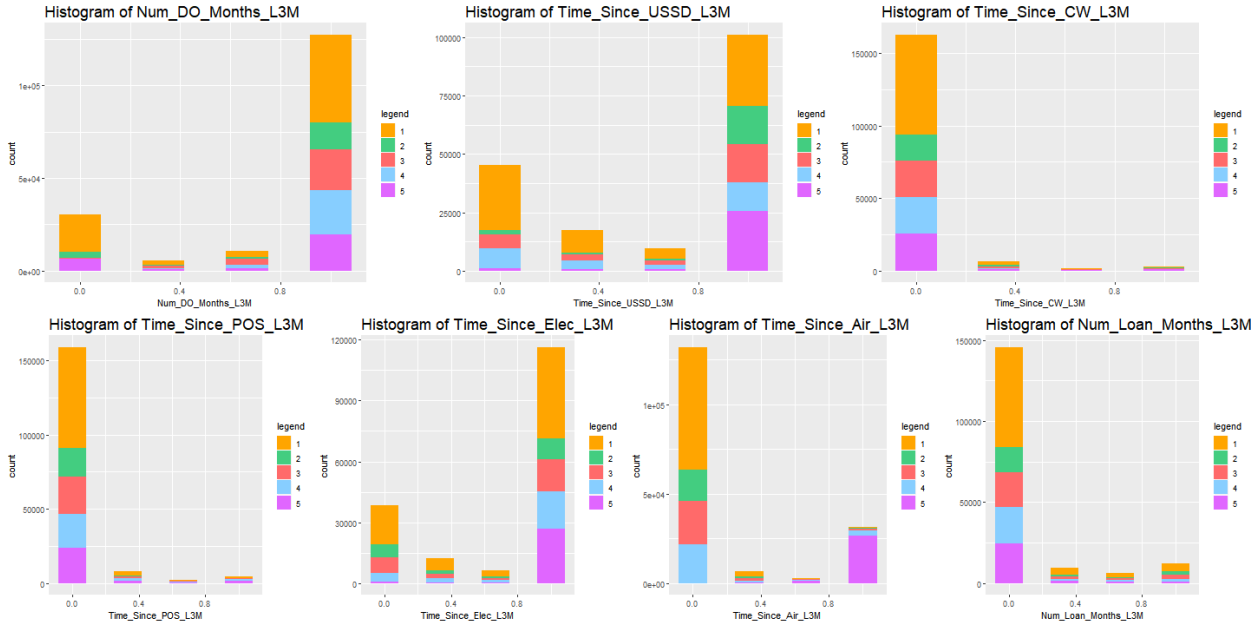


Figure 42: Histograms of the variables containing information on monthly frequency of the different channel's usage, colour-coded according to clusters for 3M_SAMP1

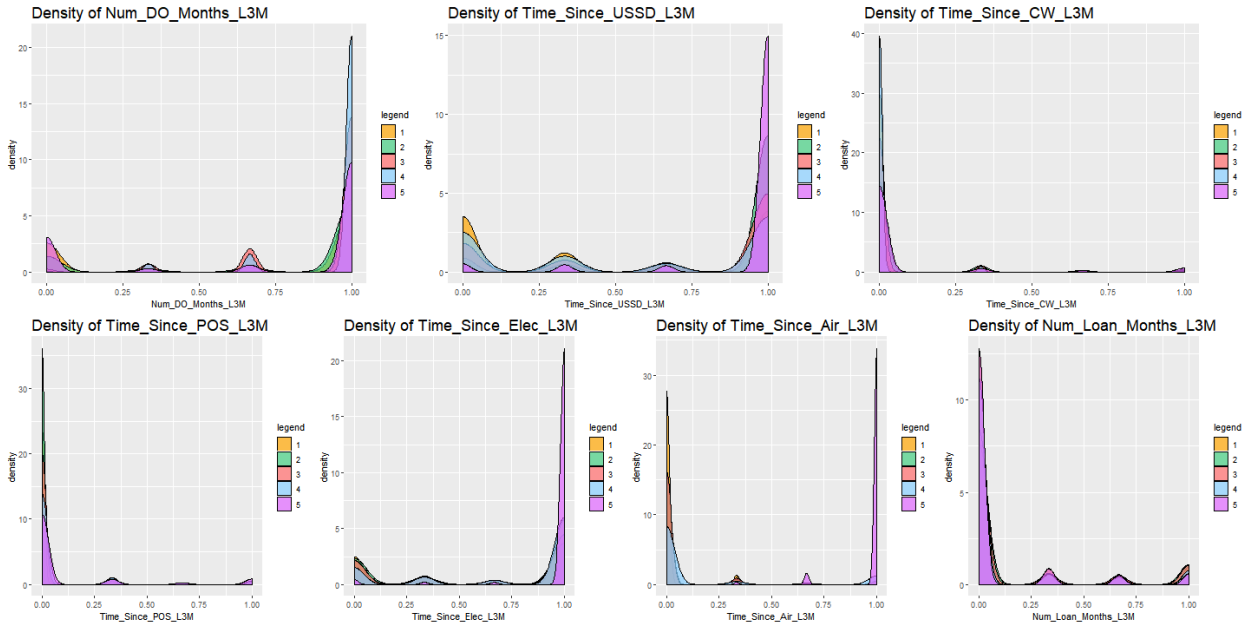


Figure 43: Density plots of the variables containing information on monthly frequency of the different channel's usage, colour-coded according to clusters for 3M_SAMP1

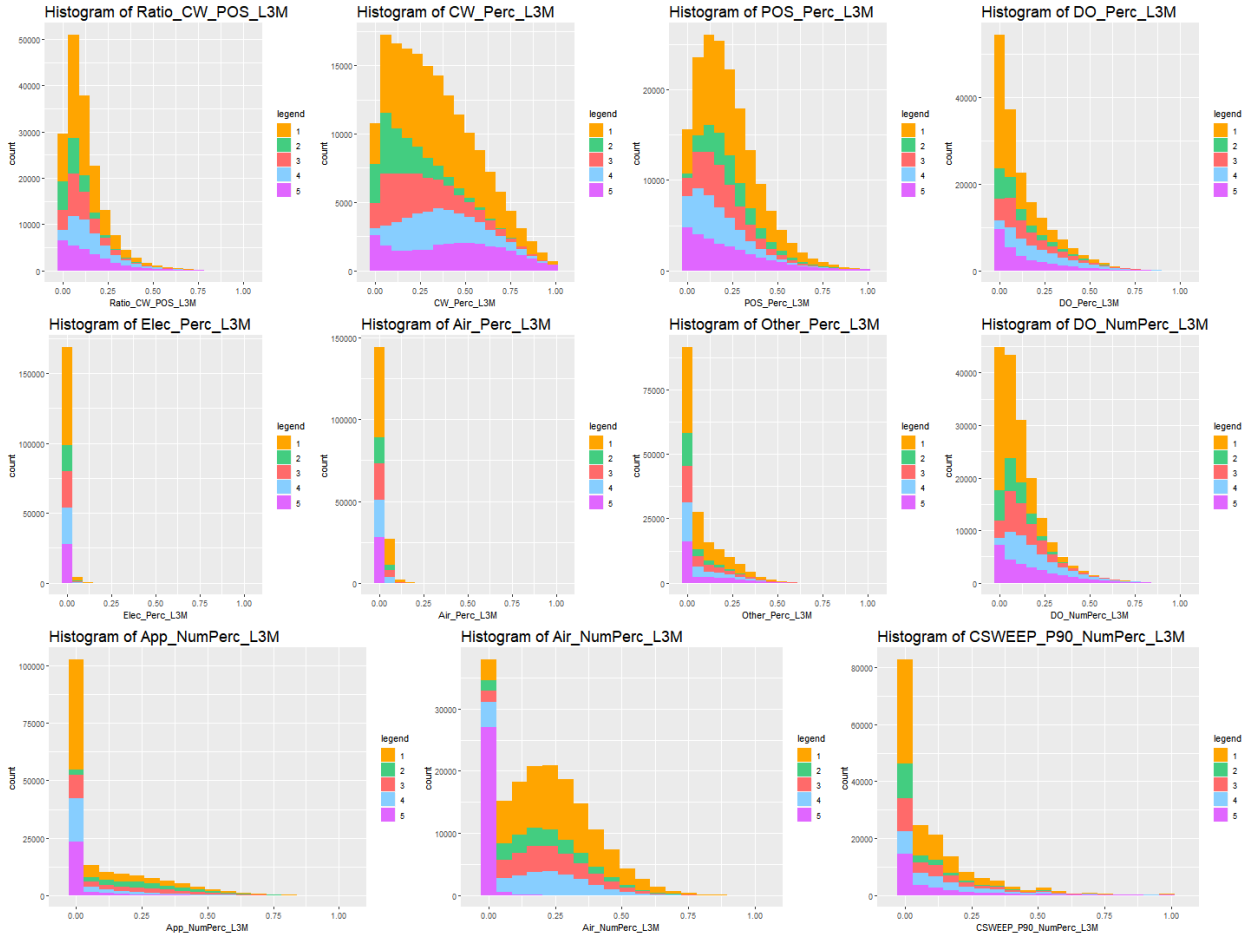


Figure 44: Histograms of percentage and ratio variables, colour-coded according to clusters for 3M_SAMP1

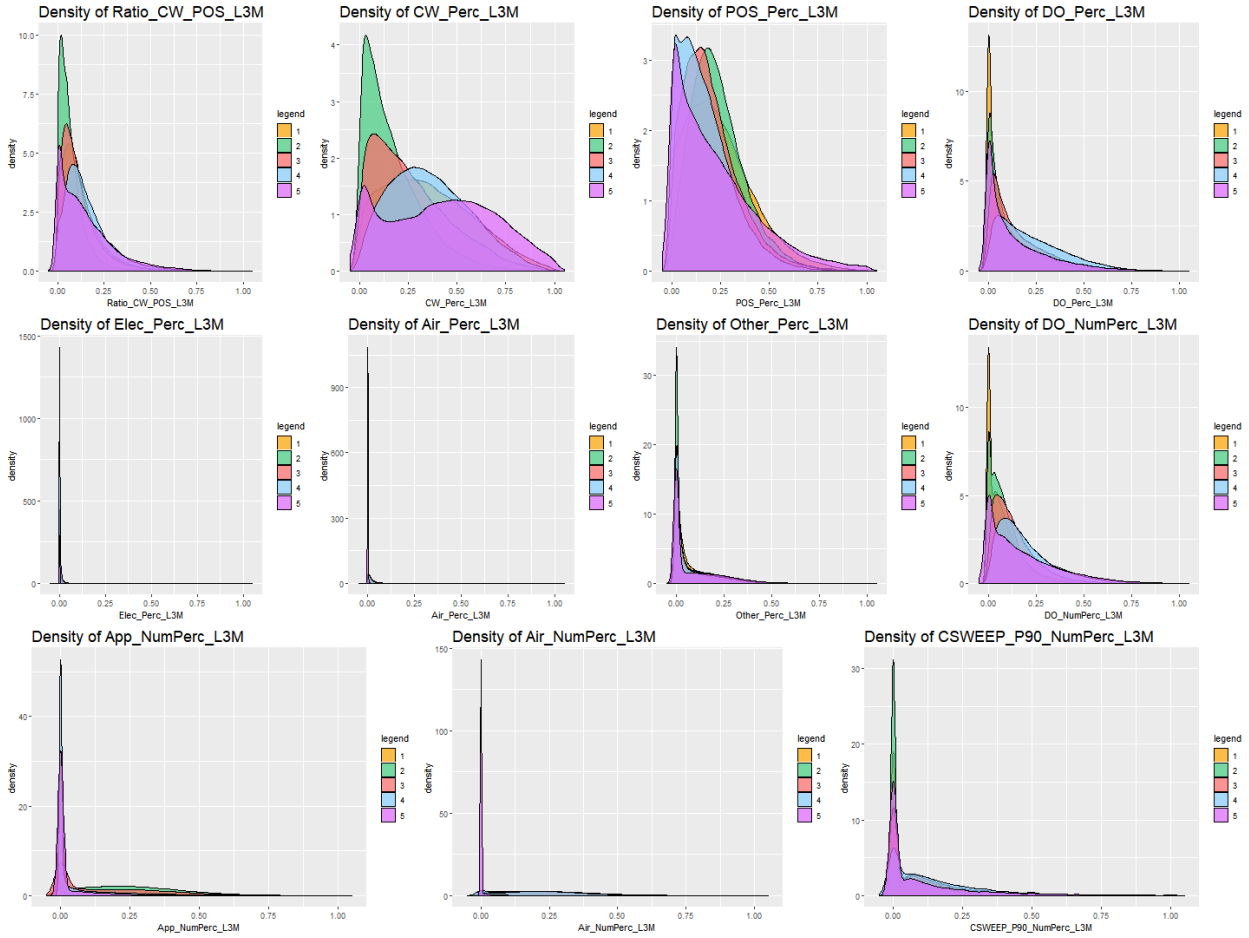


Figure 45: Density plots of percentage and ratio variables, colour-coded according to clusters for 3M_SAMP1

Figure 46: Histograms of the variables containing information on sizes of inflows and outflows, colour-coded according to clusters for 3M_SAMP1

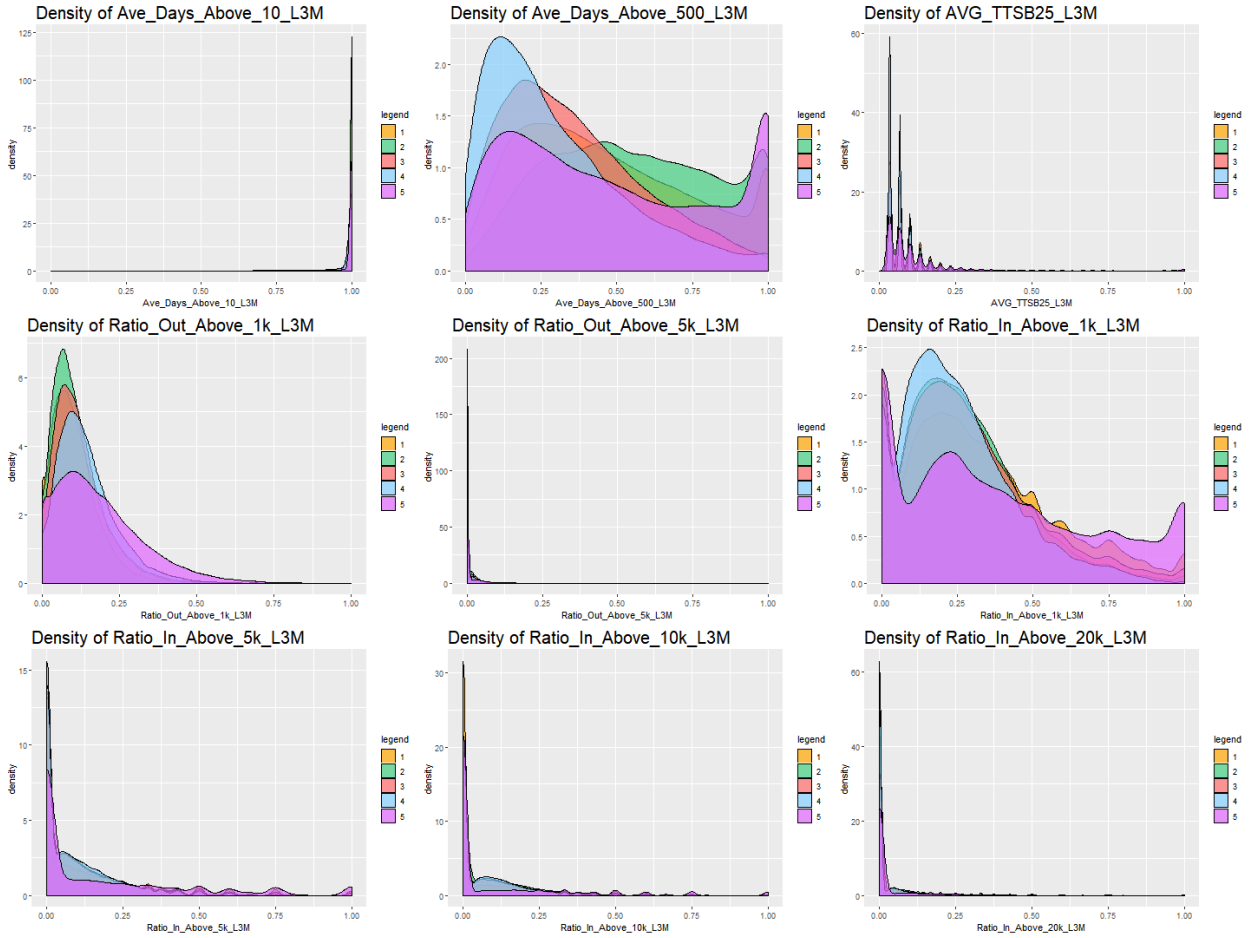


Figure 47: Density plots of the variables containing information on sizes of inflows and outflows, colour-coded according to clusters for 3M_SAMP1

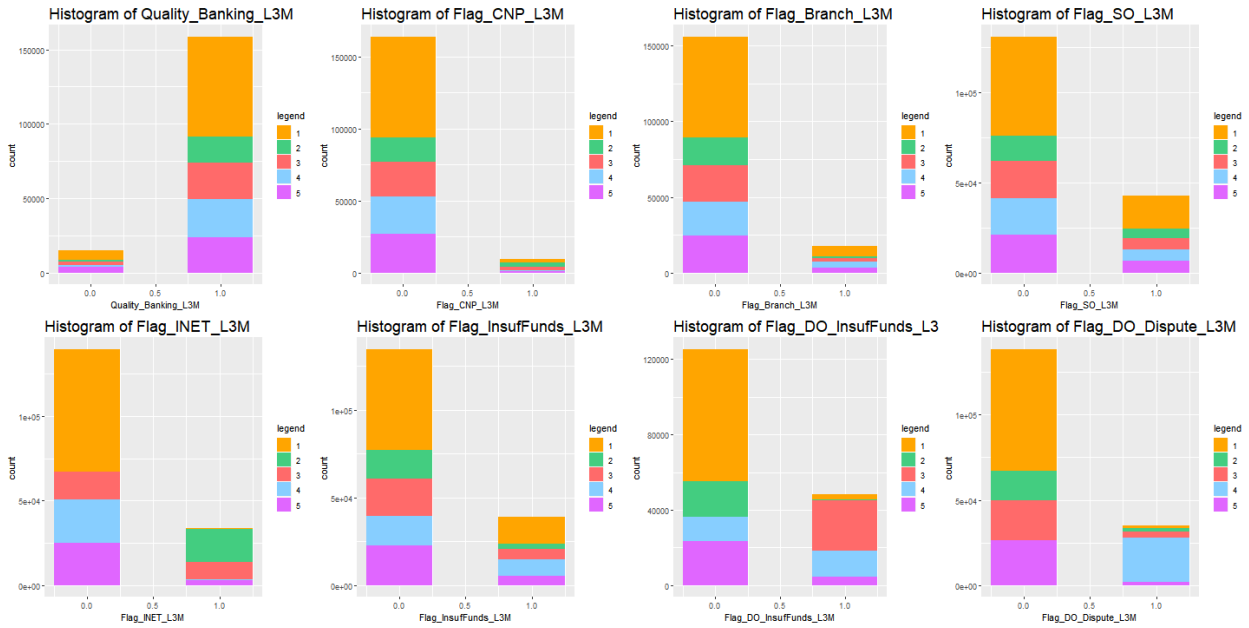


Figure 48: Histograms of indicator variables, colour-coded according to clusters for 3M_SAMP1

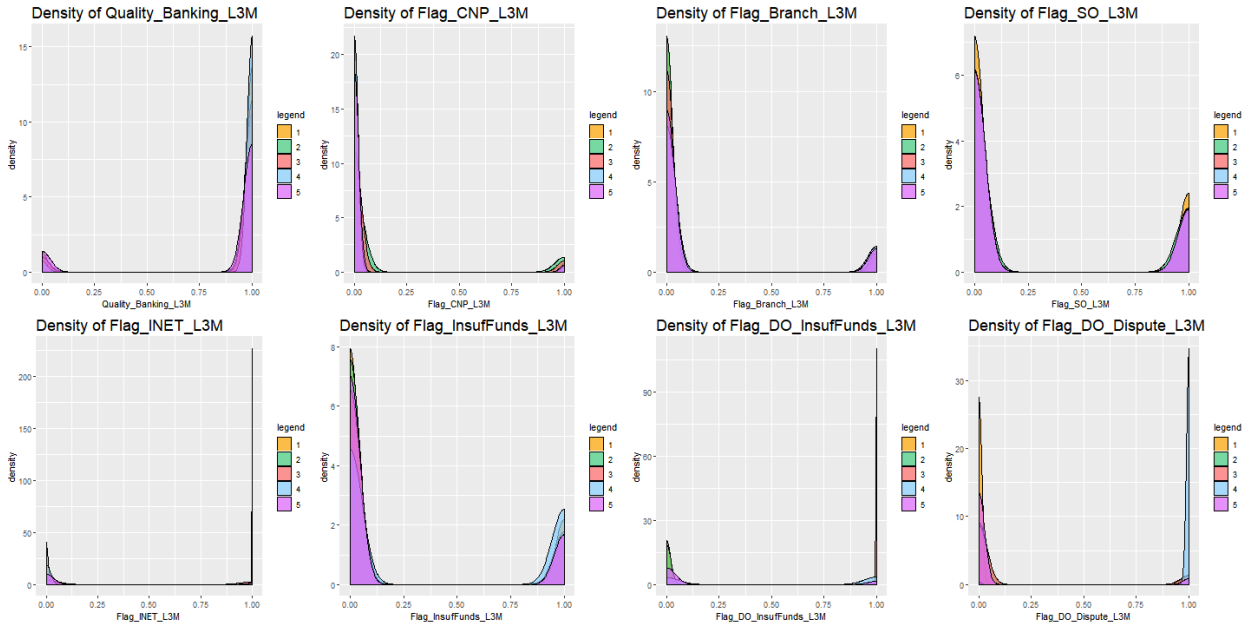


Figure 49: Density plots of indicator variables, colour-coded according to clusters for 3M_SAMP1

chapter, to determine if the different clusters represent different behavioural traits of clients. These histograms and distribution plots can be found in Figure 40 to Figure 49.

It is difficult to separate the clusters into vastly different behavioural characteristic groups as the clusters are overlapping. The main characteristics of the clients in each cluster is listed, together with an extrapolation of what the average client would look like in South Africa. These extrapolations could contain personal bias and prior knowledge and should be re-evaluated if any more information becomes available.

Cluster 1:

- Clients make very little use of Debit Orders and NAEDOs
- Clients make use of the USSD channel
- Clients frequently buy electricity and airtime
- Clients have lower inflow values and a lower number of inflows

Cluster 1 could represent a lower socio-economic class of clients. The usage of USSD transactions might indicate that the clients do not have smartphones. The repeated electricity transactions may indicate that the clients live in apartments or small houses, as most larger houses in South Africa would receive monthly electricity bills and not make use of prepaid electricity. The repeated airtime transactions can also indicate that prepaid cellphones are used, as cellphone contracts are paid monthly through Debit Orders.

This cluster of clients possibly has lower interaction with emails, but could possibly be reached through SMS or through a phone call. If financial education is made available to these clients it might have to be delivered in person to be effective.

Cluster 2:

- Clients make the most use of the INET, App and CNP channels
- Clients rarely utilise the USSD channel
- Clients rely less on cash withdrawals and cash usage
- Clients have the largest number of inflows and outflows
- Clients have the highest inflow values, on average

Cluster 2 may possibly represent the technologically sophisticated clients in the middle to upper socio-economic classes. These clients probably have smartphones and make use of CNP transactions, that are only used by more “tech savvy” clients. These clients are also very financially active and have moved away from cash usage.

The clients in Cluster 2 possibly do not want to be bothered by phone calls or SMS. The best way to reach these clients would possibly be through emails, so they can respond in their own time, or through notifications on the Bank C cellphone application. If any further financial education is needed it could most likely be communicated via email.

Cluster 3:

- Clients frequently buy airtime
- Clients rely less on cash withdrawals and cash usage
- Clients had the largest number of DO Insufficient Fund flags

Cluster 3 does not have particular traits that sets it apart from the other clusters. This cluster might contain clients that could not fit into the other four clusters. This cluster can be investigated further to determine if subclusters are needed.

Cluster 4:

- Clients make the most use of Debit Orders and NAEDOs
- Clients have the most Debit Order disputes, but this might follow from the high Debit Order usage
- Clients frequently buy airtime
- Clients do not utilise the App channel
- Majority of the clients are Quality Banking clients

Cluster 4 may contain some older, more traditional clients that prefer the usage of Debit Orders, but do not prefer to make use of a cellphone banking application for banking. These clients are mostly Quality Banking clients, which will put them in a good income bracket and they display consistent loyal behaviour towards Bank C.

Seeing as Cluster 4 consists of more traditional clients it might be necessary to use traditional communication channels. These would include phone calls, or even letters in the post. Some traditional clients are using internet banking and emails, but are not fully comfortable with cellphone banking. Financial education for this group might mean education on how to use a cellphone banking application and what the benefits would be of switching over to this banking channel.

Cluster 5:

- Clients do not really make use of the USSD channel
- Clients do not buy electricity or airtime
- Clients mostly rely on cash withdrawals and cash usage
- Clients had the lowest number of debit transactions, indicating lower financial activity
- Clients have lower inflow values

Cluster 5 seems to be heterogeneous as there are variables, like Ratio_In_Above_1k_L3M and Ave_Days_Above_500_L3M, where the distributions have more than one peak. This could indicate a mixture distribution and that the clients could be split into smaller groups. The distributions for these two variables seem to be leaning towards uniformity, which could also indicate that these variables do not have an influence on cluster 5. Regardless of the concerns stated here, Cluster 5 contains clients with lower financial activity. This might be due to limited finances or this might indicate that Bank C is not the primary bank. Another possibility is that these clients mostly utilise cash and there is no way to know how it was spent.

It is not completely clear what the best channels of communication for Cluster 5 would be. It would be beneficial to get more information from these clients, either by encouraging less cash based transactions or by convincing them of the benefits of switching to Bank C as primary bank.

In Appendix I the histograms and density plots of the variables for the 3M_SAMP2 dataset

can also be found, together with summaries of the cluster traits. It was found that natural borders between the clusters form in terms of technological sophistication and possible income level and economic standing.

The 5 clusters outlined above appear consistently throughout the different data samples, with minor variations in the characteristics. The results from the remaining 7 datasets were investigated and the four main clusters of lower socio-economic class clients, technologically sophisticated clients, older and more traditional clients and low financial activity clients were present, to some extent, in all of them. A fifth cluster capturing all the clients who did not fit into the other clusters were also present. This result is very comforting, as consistently finding 5 similar clusters in all the datasets confirm that the clients can indeed be separated into groups. Different communication methods can be applied to each group and financial education and product recommendations can be made. The final result will hopefully be to drive up client satisfaction and quality of life.

It would have been useful to name the different clusters to provide archetypes associated with each cluster. This proves difficult as two of the clusters still need further investigation. Naming the clusters is recommended for a second round investigation once the clusters are more homogeneous.

It would have also been insightful to compare the different cluster outcomes for the different time horizons using external criteria, such as the Rand index or the Jaccard index. This might have given an indication of the similarities between the different cluster outcomes. It is recommended for further investigation.

6.7 Summary

In this chapter we investigated the real-world transactional datasets provided by Bank C. SOMs with Ward linkage clustering fit on top was used to partition the data into 5 different clusters. The clustering structure held consistently for the different datasets. These clusters represented lower socio-economic class clients, technologically sophisticated clients, older and more traditional clients and low financial activity clients. These groups of clients have to be approached in different ways and would have different requirements for a good banking experience.

More research remains to be done. Firstly, into larger values for K and secondly, into the possibility of using a fuzzy clustering technique. It is also possible to reduce the number of variables further, to try and reduce noise and enhance the signal. The clusters that have been formed can also be investigated further by mapping the original sample of roughly 1.7 million clients onto the clusters. This will be a good test to determine if the clustering structure holds for the whole dataset.

In the next chapter the conclusions from this study will be discussed, along with a summary of the scope of this study and recommendations for further work and research.

CHAPTER 7: CONCLUSION

7 Summary, Conclusion and Recommendations

7.1 Introduction

This study focused on clustering and the various methods available for this task. Traditional clustering methods, as well as Self-Organising Maps (SOMs) and CLARA, were investigated. The aim of investigating the different methods was to find a traditional method that performs well when applied on top of a SOM. The purpose of this combination of methods was to investigate high-dimensional transactional banking data and to determine whether client behaviour groups could be formed in the data. Data extraction, cleaning and normalisation was also part of the process.

7.2 Summary of Main Findings

Chapter 2 provided a literature review on traditional clustering methods, originally designed for small and low-dimensional datasets. Traditional clustering methods included K-means, K-medoids (PAM) and agglomerative and divisive hierarchical clustering. The algorithms for these methods were discussed in detail. The different forms of linkage available to agglomerative hierarchical clustering were also discussed. We investigated different methods for choosing the appropriate number of clusters which were the gap statistic, the elbow method and the average silhouette method.

We summarised criteria to use for cluster validation, even though there is no agreed upon way by researchers of quantifying the quality of a clustering. The indices used to compare a clustering of the data to the true cluster labels were Rand, Jaccard and Fowlkes and Mallows. The indices used to compare the compactness and isolation of two sets of clusters were the average silhouette value, the Dunn index and the Davies-Bouldin index.

In Chapter 3 we looked at large and high-dimensional datasets and clustering methods that would be more appropriate for this. CLARA was described as an adaption to the PAM algorithm, by including a random sampling step, for large datasets. SOMs, which is the main focus of this study, were discussed in detail. The batch and online versions of the SOM algorithm were investigated, as well as competitive learning, which is the foundation of the SOM algorithm. We discussed different ways of choosing the dimensions for and initialising the SOM prototype grid, as well as the initialisation for the learning rate and the neighbourhood function. A SOM adaption, called Growing SOMs, were also discussed. This algorithm allows the SOM grid to grow to the size that is appropriate for the data. We also discussed the possibility of using SOMs for dimension reduction and fitting other clustering methods with the prototypes as the inputs.

We would like the SOM to learn slowly, thus the learning rate and neighbourhood functions were recommended to be set very large and to slowly decrease over many iterations of the

algorithm. It was recommended that the batch SOM is the most suited SOM algorithm to be used in practice as it needs limited parameter tuning and converges faster than the online SOM. A hexagonal map topology was recommended to be used as it resembles the input space more closely. It was also recommended that random initialisation be used when datasets are nonlinear and complex.

Chapter 4 contained a simulation study of the clustering methods discussed before. Two simulated Gaussian 10-dimensional data scenarios were used to fit the different clustering methods and to compare the outcomes. Scenario A, with only three balanced clusters, was simpler than Scenario B, with 11 unbalanced clusters, and was used to explain all of the R functions and packages that would be needed. Scenario B was used to compare the different SOM algorithms and also to determine which traditional clustering method fits the best on top of a SOM.

From Scenario A we concluded that Ward linkage clustering performs the best from the traditional clustering methods. Ward linkage returned compact clusters and also returned the correct cluster labels, even though the clusters were close together. Scenario B helped confirm the recommendations made in the literature review and showed that the batch SOM with random initialisation and a hexagonal grid topology is the most appropriate for complex data exhibiting nonlinear relationships between the input variables. Scenario B was also used to confirm that CLARA performs better than PAM with a large dataset. Ward linkage was determined to be the most appropriate traditional clustering algorithm to fit on top of the batch SOM. The batch SOM as a dimension reduction technique was very successful and increased the accuracy of the classification of the data points of Scenario B.

In Chapter 5 we focused our attention on a real-world dataset. This dataset is transactional information of clients provided by Bank C. The goal is to find client behavioural groups in the transactional data. These groups can be communicated with in different ways, they might have different banking product needs and can also be offered financial education material if needed. Before model fitting, the data had to be extracted and cleaned. Nine different samples, of varying sizes and observation windows, were extracted from the original data. A log-transformation was applied to increase the interpretability of the variables. Outliers were also eliminated to alleviate the skewness of the data. The dimensions of the data were reduced by investigating the number of zero values in each variable and also the correlations between the variables. Lastly, the data had to be normalised to place all the variables on the same measurement scale. This resulted in 9 clean datasets with the number of observations varying between roughly 50 000 and 174 000. The number of variables are varying from 38 to 49.

Chapter 6 was the data modelling chapter. We applied a batch SOM to each of the datasets as a dimension reduction tool. Ward linkage was fit on top of each of these SOMs and resulted in 5 clusters each of different client behavioral groups. These clusters represented lower socio-economic class clients, technologically sophisticated clients, older and more traditional clients and low financial activity clients. The fifth cluster represented clients that did not have a clear set of characteristics that could be combined into a client profile.

7.3 Recommendations

Further research can be done into fuzzy clustering methods as the real-world dataset seemed more suited to this kind of clustering. The data is very compact and there is large overlap between the clusters found by the SOM with Ward linkage on top.

Different methods of outlier detection is another topic that should also be investigated further. The transactional datasets could have benefited from more thorough outlier detection and removal, as some of the variables were very skewed. Similarly, variable reduction methods for unsupervised learning could also be investigated. It would have been beneficial if there was a way to determine the influence of each of the variables on the clustering.

Thirdly, there is an opportunity to consider a higher number of clusters than 5. Evidence for this was returned by the gap statistic and the average silhouette method. This would be more time consuming, but would possibly lead to more homogeneous clusters.

Spending time developing an R package for GSOMs would also be very useful. There was no way to test the GSOMs algorithm in R, which was rather disappointing.

There are also further developments available for SOMs, for example the Adaptive Moving Self-Organising Map (AMSOM) that also incorporates dynamic SOM grid growth or SOMs grown on irregular maps, like mnemonic SOMs, that can be investigated. The SOM was proven as a worthwhile dimension reduction tool and learning about further possible applications can be very useful.

References

Akinduko, A.A., Mirkes, E.M. & Gorban, A.N. 2016. SOM: Stochastic initialization versus principal components. *Information sciences*, 364-365:213-221.

Alahakoon, D., Halgamuge, S.K. & Srinivasan, B. 2000. Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE transactions on neural networks*, 11(3):601-614.

Auguie, B. 2017. gridExtra: Miscellaneous functions for "grid" graphics. R package version 2.3. https://CRAN.R-project.org/package=gridExtra.

Boelaert, J., Bendhaiba, L., Olteanu, M. & Villa-Vialaneix, N. 2014. SOMbrero: An R package for numeric and non-numeric self-organizing maps, in Villmann T., Schleif F.M., Kaden M. & Lange M. (eds.) *Advances in self-organizing maps and learning vector quantization. Advances in intelligent systems and computing, vol 295.* Mittweida, Germany: Springer. 219-228.

Charrad, M., Ghazzali, N., Boiteau, V. & Niknafs, A. 2014. NbClust: An R package for determining the relevant number of clusters in a data set. *Journal of statistical software*, 61(6):1-36.

Dahl, D.B. 2016. xtable: Export tables to LaTeX or HTML. R package version 1.8-2. https://CRAN.R-project.org/package=xtable.

Davies, D.L. & Bouldin, D.W. 1979. A cluster separation measure. *IEEE transactions on pattern analysis and machine intelligence*, 2:224-227.

Dunn, J.C. 1974. Well-separated clusters and optimal fuzzy partitions. *Journal of cybernetics*, 4(1):95-104.

Everitt, B.S., Landau, S., Leese, M. & Stahl, D. 2011. *Cluster analysis.* 5th ed. Chichester, West Sussex: John Wiley & Sons, Ltd.

Forgy, E.W. 1965. Cluster analysis of multivariate data: Efficiency versus interpretability of classifications. *Biometrics*, 21:768-769.

Fowlkes, E.B. & Mallows, C.L., 1983. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553-569.

Galili, T. 2015. dendextend: An R package for visualizing, adjusting, and comparing trees of hierarchical clustering. *Bioinformatics*, 31(22):3718-3720.

Hastie, T., Tibshirani, R. & Friedman, J. 2009. *The elements of statistical learning: Data mining, inference, and prediction.* 2nd ed. New York: Springer.

Jaccard, P. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547-579.

Jain, A.K. & Dubes, R.C. 1988. *Algorithms for clustering data.* Upper Saddle River, NJ: Prentice-Hall, Inc.

James, G., Witten, D., Hastie, T. & Tibshirani, R. 2013. *An introduction to statistical learning with applications in R.* New York: Springer.

Kassambara, A. & Mundt, F. 2017. factoextra: Extract and visualize the results of multivariate data analyses. R package version 1.0.5. https://CRAN.R-project.org/package=factoextra.

Kaufman, L. & Rousseeuw, P.J. 1986. Clustering large data sets, in Gelsema, E.S. & Kanal, L.N. (eds.). *Pattern recognition in practice II.* North Holland: Elsevier. 425-437.

Kaufman, L. & Rousseeuw, P.J. 1986. Clustering by means of medoids, in Dodge, Y. (ed.). *Statistical data analysis based on the L_1-norm and related methods.* North-Holland: Elsevier. 405-416.

Kaufman, L. & Rousseeuw, P.J. 1990. *Finding groups in data: An introduction to cluster analysis.* New York: John Wiley & Sons, Inc.

Kohonen, T. 1982. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59-69.

Kohonen, T. 2001. *Self-organizing maps.* 3rd ed. Berlin, Heidelberg: Springer-Verlag.

Kohonen, T. 2013. Essentials of the self-organizing map. *Neural networks*, 37:52-65.

Lloyd, S., 1982. Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129-137.

Macnaughton-Smith, P., Williams, W.T., Dale, M.B. & Mockett, L.G., 1964. Dissimilarity analysis: A new technique of hierarchical sub-division. *Nature*, 202:1034-1035.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1(14):281-297.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M. & Hornik, K. 2018. cluster: Cluster analysis basics and extensions. R package version 2.0.7-1. https://CRAN.R-project.org/package=cluster.

Mayer, R., Merkl, D. & Rauber, A., 2005. Mnemonic SOMs: Recognizable shapes for self-organizing maps. *Proceedings of the 5th Workshop On Self-Organizing Maps Paris (WSOM 2005)*, 131-138.

Murrell, P. 2014. gridBase: Integration of base and grid graphics. R package version 0.4-7. https://CRAN.R-project.org/package=gridBase.

Murtagh, F. & Legendre, P. 2014. Ward's hierarchical agglomerative clustering method: Which algorithms implement Ward's criterion? *Journal of classification* 31:274-295.

Nieweglowski, L. 2015. clv: Cluster validation techniques. R package version 0.3-2.1. https://CRAN.R-project.org/package=clv.

R Core Team, 2018. R: A language and environment for statistical computing. R Foundation for Statistical Computing. Vienna, Austria. https://www.R-project.org/.

Rand, W.M., 1971. Objective criteria for the evaluation of clustering methods. *Journal of the American statistical association*, 66(336):846-850.

Rousseeuw, P.J. 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53-65.

Rudis, B., Bolker, B. & Schulz, J. 2017. ggalt: Extra coordinate systems, 'geoms', statistical transformations, scales and fonts for 'ggplot2'. R package version 0.4.0. https://CRAN.R-project.org/package=ggalt.

Simovici, D.A. 2011. The PAM clustering algorithm. Unpublished class notes (Data Mining CS738). Boston: University of Massachusetts.

Tibshirani, R., Guenther, W. & Hastie, T. 2001. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal statistical society series B (Statistical methodology)*, 63(2):411-423.

Venables, W.N. & Ripley, B.D. 2002. *Modern applied statistics with S.* 4th ed. New York: Springer.

Vesanto, J. & Alhoniemi, E. 2000. Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, 11(3):586-600.

Villa-Vialaneix N., Bendhaiba L. & Olteanu M. 2018. SOMbrero: SOM Bound to realize Euclidean and relational outputs. R package version 1.2-3. https://CRAN.R-project.org/package=SOMbrero.

Weatherwax, J.L. & Epstein, D. 2018. A solution manual and notes for: The elements of statistical learning by Jerome Friedman, Trevor Hastie and Robert Tibshirani. Unpublished solution manual.

Wehrens, R. & Buydens, L.M.C. 2007. Self- and super-organising maps in R: The kohonen package. *Journal of Statistical Software*, 21(5).

Wehrens, R. & Kruisselbrink, J. 2018. Supervised and unsupervised self-organising maps. R package version 3.0.6. https://cran.r-project.org/web/packages/kohonen/.

Wickham, H. 2016. *ggplot2: Elegant graphics for data analysis.* New York: Springer-Verlag.

Xie, Y. 2018. knitr: A general-purpose package for dynamic report generation in R. R package version 1.20. https://yihui.name/knitr/.

Xu, R. & Wunsch II, D.C. 2009. *Clustering.* Hoboken, New Jersey: John Wiley & Sons, Inc.

Appendix A

$$
\begin{aligned}
W(C) &= \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k}\|\boldsymbol{x}_i-\boldsymbol{x}_{i'}\|^2 \\
&= \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k}(\boldsymbol{x}_i-\boldsymbol{x}_{i'})^T(\boldsymbol{x}_i-\boldsymbol{x}_{i'}) \\
&= \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k}\left(\boldsymbol{x}_i^T\boldsymbol{x}_i-2\boldsymbol{x}_i^T\boldsymbol{x}_{i'}+\boldsymbol{x}_{i'}^T\boldsymbol{x}_{i'}\right) \\
&= \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\left[N_k(\boldsymbol{x}_i^T\boldsymbol{x}_i)-2\boldsymbol{x}_i^T(N_k\bar{\boldsymbol{x}}_k)+\sum_{C(i')=k}\boldsymbol{x}_{i'}^T\boldsymbol{x}_{i'}\right] \\
&= \frac{1}{2}\sum_{k=1}^{K}\left[N_k\sum_{C(i)=k}\boldsymbol{x}_i^T\boldsymbol{x}_i-2(N_k\bar{\boldsymbol{x}}_k)^T(N_k\bar{\boldsymbol{x}}_k)+N_k\sum_{C(i')=k}\boldsymbol{x}_{i'}^T\boldsymbol{x}_{i'}\right] \\
&= \frac{1}{2}\sum_{k=1}^{K}N_k\left[2\sum_{C(i)=k}\boldsymbol{x}_i^T\boldsymbol{x}_i-2N_k\bar{\boldsymbol{x}}_k^T\bar{\boldsymbol{x}}_k\right] \\
&= \sum_{k=1}^{K}N_k\left[\sum_{C(i)=k}\boldsymbol{x}_i^T\boldsymbol{x}_i-N_k\bar{\boldsymbol{x}}_k^T\bar{\boldsymbol{x}}_k+2\sum_{C(i)=k}\boldsymbol{x}_i^T\bar{\boldsymbol{x}}_k-2\sum_{C(i)=k}\boldsymbol{x}_i^T\bar{\boldsymbol{x}}_k\right] \\
&= \sum_{k=1}^{K}N_k\left[\sum_{C(i)=k}\boldsymbol{x}_i^T\boldsymbol{x}_i-N_k\bar{\boldsymbol{x}}_k^T\bar{\boldsymbol{x}}_k+2N_k\bar{\boldsymbol{x}}_k^T\bar{\boldsymbol{x}}_k-2\sum_{C(i)=k}\boldsymbol{x}_i^T\bar{\boldsymbol{x}}_k\right] \\
&= \sum_{k=1}^{K}N_k\left[\sum_{C(i)=k}\boldsymbol{x}_i^T\boldsymbol{x}_i+N_k\bar{\boldsymbol{x}}_k^T\bar{\boldsymbol{x}}_k-2\sum_{C(i)=k}\boldsymbol{x}_i^T\bar{\boldsymbol{x}}_k\right] \\
&= \sum_{k=1}^{K}N_k\sum_{C(i)=k}\|\boldsymbol{x}_i-\bar{\boldsymbol{x}}_k\|^2
\end{aligned}
$$

where $\sum_{C(i)=k}\boldsymbol{x}_i = N_k\bar{\boldsymbol{x}}_k$ (Weatherwax & Epstein, 2018:110-112).

Appendix B

$$
\begin{aligned}
d(\boldsymbol{z}_i, \boldsymbol{z}_{i'}) &= \sum_{\ell=1}^{p} (z_{i\ell} - z_{i'\ell})^2 \\
&= \sum_{\ell=1}^{p} \left[x_{i\ell} \left(\frac{w_\ell}{\sum_{\ell=1}^{p} w_\ell} \right)^{\frac{1}{2}} - x_{i'\ell} \left(\frac{w_\ell}{\sum_{\ell=1}^{p} w_\ell} \right)^{\frac{1}{2}} \right]^2 \\
&= \sum_{\ell=1}^{p} \left[\left(\frac{w_\ell}{\sum_{\ell=1}^{p} w_\ell} \right)^{\frac{1}{2}} (x_{i\ell} - x_{i'\ell}) \right]^2 \\
&= \sum_{\ell=1}^{p} \left(\frac{w_\ell}{\sum_{\ell=1}^{p} w_\ell} \right) (x_{i\ell} - x_{i'\ell})^2 \\
&= \frac{\sum_{\ell=1}^{p} w_\ell (x_{i\ell} - x_{i'\ell})^2}{\sum_{\ell=1}^{p} w_\ell} \\
&= d^{(w)}(\boldsymbol{x}_i, \boldsymbol{x}_{i'})
\end{aligned}
$$

so that the weighted Euclidean distance for X is equal to the unweighted Euclidean distance for Z (Weatherwax & Epstein, 2018:110-112). where $\sum_{C(i)=k} \boldsymbol{x}_i = N_k \bar{\boldsymbol{x}}_k$ (Weatherwax & Epstein, 2018:110-112).

Appendix C

The piece of code below generates spherical and elongated clusters and can also add noise points. The types of clusters can be a mix of spherical and elongated clusters. The default state of the function is that there are no elongated clusters and that no noise points are present.

```
GenXDat <- function(N_vec, mu_mat, cov_mat, elong_num = 0){
  #N_vec: vector of cluster sizes
  #mu_mat: matrix with mean vectors as rows
  #cov_mat: covariance matrix generated by CovMatFunc
  #elong_num: number of elongated clusters

  Num_clus <- length(N_vec)
  Num_p <- ncol(mu_mat)
  XDat_Mat <- NULL
  XDat_Mat_Norm <- NULL

  if(elong_num == Num_clus){

    XDat_Mat_Norm1 <- NULL
    XDat_Mat_Norm2 <- NULL
    XDat_Mat_Norm3 <- NULL

    for(i in (1:Num_clus)){
      set.seed(123456 + i*115)
      XDat_Mat_Norm1 <- mvrnorm(n = (ceiling((N_vec[i])/3)), mu = mu_mat[i,],
                                Sigma = cov_mat)
      XDat_Mat_Norm2 <- mvrnorm(n = (ceiling((N_vec[i])/3)), mu = mu_mat[i,]+3,
                                Sigma = cov_mat)
      XDat_Mat_Norm3 <- mvrnorm(n = N_vec[i] - (2*ceiling((N_vec[i])/3)),
                                mu = mu_mat[i,]+1.5, Sigma = cov_mat)
      XDat_Mat_Norm <- rbind(XDat_Mat_Norm1, XDat_Mat_Norm2, XDat_Mat_Norm3)
      XDat_Mat_Norm <- cbind(XDat_Mat_Norm, i)
      XDat_Mat <- rbind(XDat_Mat, XDat_Mat_Norm)
    }

  }else if(elong_num > 0 & elong_num != Num_clus){

    XDat_Mat_Norm1 <- NULL
    XDat_Mat_Norm2 <- NULL
    XDat_Mat_Norm3 <- NULL

    for(i in (1:elong_num)){
      set.seed(123456 + i*115)
```

```
      XDat_Mat_Norm1 <- mvrnorm(n = (ceiling((N_vec[i])/3)), mu = mu_mat[i,],
                                Sigma = cov_mat)
      XDat_Mat_Norm2 <- mvrnorm(n = (ceiling((N_vec[i])/3)), mu = mu_mat[i,]+3,
                                Sigma = cov_mat)
      XDat_Mat_Norm3 <- mvrnorm(n = N_vec[i] - (2*ceiling((N_vec[i])/3)),
                                mu = mu_mat[i,]+1.5, Sigma = cov_mat)
      XDat_Mat_Norm <- rbind(XDat_Mat_Norm1, XDat_Mat_Norm2, XDat_Mat_Norm3)
      XDat_Mat_Norm <- cbind(XDat_Mat_Norm, i)
      XDat_Mat <- rbind(XDat_Mat, XDat_Mat_Norm)
    }


    for(i in ((elong_num+1):Num_clus)){
      set.seed(7896 + i*115)
      XDat_Mat_Norm <- mvrnorm(n = N_vec[i], mu = mu_mat[i,], Sigma = cov_mat)
      XDat_Mat_Norm <- cbind(XDat_Mat_Norm, i)
      XDat_Mat <- rbind(XDat_Mat, XDat_Mat_Norm)
    }

  }else{

    for(i in (1:Num_clus)){
      set.seed(7896 + i*115)
      XDat_Mat_Norm <- mvrnorm(n = N_vec[i], mu = mu_mat[i,], Sigma = cov_mat)
      XDat_Mat_Norm <- cbind(XDat_Mat_Norm, i)
      XDat_Mat <- rbind(XDat_Mat, XDat_Mat_Norm)
    }
  }

  colnames(XDat_Mat) <- c(paste("X", 1:Num_p, sep = ""), "Class")
  return(XDat_Mat)
}
```

Appendix D

Here a 5-point summary for each of the variables in dataset 3M_SAMP1 can be seen. The variables have not been cleaned and can be seen in their raw states, containing outliers and on the original measurement scales.

```
##  Val_Ct_Tran_L3M     Val_Dt_Tran_L3M      Val_USSD_L3M
##  Min.   :     6046   Min.   :     2522   Min.   :      0
##  1st Qu.:    21714   1st Qu.:    21458   1st Qu.:      0
##  Median :    36703   Median :    36345   Median :      0
##  Mean   :    53683   Mean   :    53075   Mean   :   2525
##  3rd Qu.:    61497   3rd Qu.:    61028   3rd Qu.:   2250
##  Max.   :145671395   Max.   :143100952   Max.   :2204310
##   Val_App_L3M       Val_INET_L3M        Val_DO_L3M
##  Min.   :      0   Min.   :       0   Min.   :     0.0
##  1st Qu.:      0   1st Qu.:       0   1st Qu.:   573.5
##  Median :      0   Median :       0   Median :  2779.8
##  Mean   :  10806   Mean   :    2706   Mean   :  6728.6
##  3rd Qu.:   9436   3rd Qu.:       0   3rd Qu.:  8676.8
##  Max.   :4114724   Max.   :10379077   Max.   :611655.8
##    Val_SO_L3M         Val_CW_L3M      Val_POS_L3M       Val_CNP_L3M
##  Min.   :     0.0   Min.   :     0   Min.   :      0   Min.   :     0.0
##  1st Qu.:     0.0   1st Qu.:  5200   1st Qu.:   2967   1st Qu.:     0.0
##  Median :     0.0   Median :  9876   Median :   7036   Median :     0.0
##  Mean   :   923.1   Mean   : 13014   Mean   :  10817   Mean   :   123.6
##  3rd Qu.:     0.0   3rd Qu.: 16920   3rd Qu.:  13950   3rd Qu.:     0.0
##  Max.   :252535.8   Max.   :406000   Max.   :1488517   Max.   :284759.1
##  Val_Branch_L3M       Val_Elec_L3M      Val_Air_L3M      Val_Other_L3M
##  Min.   :        0   Min.   :    0.0   Min.   :    0.0   Min.   :      0
##  1st Qu.:        0   1st Qu.:    0.0   1st Qu.:   84.0   1st Qu.:      0
##  Median :        0   Median :    0.0   Median :  386.0   Median :    784
##  Mean   :     3152   Mean   :  180.2   Mean   :  667.7   Mean   :   3343
##  3rd Qu.:        0   3rd Qu.:  100.0   3rd Qu.:  894.0   3rd Qu.:   4600
##  Max.   :143018645   Max.   :40100.0   Max.   :26845.0   Max.   :1000000
##  Num_Ct_Tran_L3M  Num_Dt_Tran_L3M   Num_USSD_L3M      Num_App_L3M
##  Min.   :  3.00   Min.   :  18.0   Min.   :  0.000   Min.   :  0.00
##  1st Qu.:  8.00   1st Qu.:  58.0   1st Qu.:  0.000   1st Qu.:  0.00
##  Median : 15.00   Median :  89.0   Median :  0.000   Median :  0.00
##  Mean   : 20.73   Mean   : 104.1   Mean   :  4.624   Mean   : 13.29
##  3rd Qu.: 26.00   3rd Qu.: 134.0   3rd Qu.:  6.000   3rd Qu.: 18.00
##  Max.   :963.00   Max.   :1080.0   Max.   :363.000   Max.   :616.00
##   Num_INET_L3M        Num_DO_L3M       Num_SO_L3M        Num_CW_L3M
##  Min.   :   0.000   Min.   :  0.00   Min.   :  0.000   Min.   :  0.00
##  1st Qu.:   0.000   1st Qu.:  3.00   1st Qu.:  0.000   1st Qu.:  8.00
##  Median :   0.000   Median :  8.00   Median :  0.000   Median : 14.00
```

```
##  Mean   :   1.154   Mean   : 10.47   Mean   :  1.186   Mean   : 16.06
##  3rd Qu.:   0.000   3rd Qu.: 15.00   3rd Qu.:  0.000   3rd Qu.: 21.00
##  Max.   :1080.000   Max.   :241.00   Max.   :309.000   Max.   :207.00
##   Num_POS_L3M      Num_CNP_L3M       Num_Branch_L3M     Num_Elec_L3M
##  Min.   :  0.0   Min.   :  0.0000   Min.   : 0.0000   Min.   :  0.000
##  1st Qu.: 12.0   1st Qu.:  0.0000   1st Qu.: 0.0000   1st Qu.:  0.000
##  Median : 27.0   Median :  0.0000   Median : 0.0000   Median :  0.000
##  Mean   : 36.8   Mean   :  0.5691   Mean   : 0.1882   Mean   :  1.543
##  3rd Qu.: 50.0   3rd Qu.:  0.0000   3rd Qu.: 0.0000   3rd Qu.:  1.000
##  Max.   :634.0   Max.   :290.0000   Max.   :49.0000   Max.   :141.000
##   Num_Air_L3M      Num_USSD_Months_L3M Num_App_Months_L3M
##  Min.   :  0.00   Min.   :0.0000      Min.   :0.0000
##  1st Qu.:  4.00   1st Qu.:0.0000      1st Qu.:0.0000
##  Median : 16.00   Median :0.0000      Median :0.0000
##  Mean   : 23.72   Mean   :0.2604      Mean   :0.4044
##  3rd Qu.: 33.00   3rd Qu.:0.6666      3rd Qu.:1.0000
##  Max.   :472.00   Max.   :1.0000      Max.   :1.0000
##  Num_INET_Months_L3M Num_DO_Months_L3M Num_SO_Months_L3M Num_CW_Months_L3M
##  Min.   :0.00000     Min.   :0.0000    Min.   :0.0000    Min.   :0.0000
##  1st Qu.:0.00000     1st Qu.:0.6666    1st Qu.:0.0000    1st Qu.:1.0000
##  Median :0.00000     Median :1.0000    Median :0.0000    Median :1.0000
##  Mean   :0.09726     Mean   :0.7850    Mean   :0.2063    Mean   :0.9371
##  3rd Qu.:0.00000     3rd Qu.:1.0000    3rd Qu.:0.0000    3rd Qu.:1.0000
##  Max.   :1.00000     Max.   :1.0000    Max.   :1.0000    Max.   :1.0000
##  Num_POS_Months_L3M Num_CNP_Months_L3M Num_Branch_Months_L3M
##  Min.   :0.0000     Min.   :0.00000    Min.   :0.00000
##  1st Qu.:1.0000     1st Qu.:0.00000    1st Qu.:0.00000
##  Median :1.0000     Median :0.00000    Median :0.00000
##  Mean   :0.9148     Mean   :0.03843    Mean   :0.04296
##  3rd Qu.:1.0000     3rd Qu.:0.00000    3rd Qu.:0.00000
##  Max.   :1.0000     Max.   :1.00000    Max.   :1.00000
##  Num_Elec_Months_L3M Num_Air_Months_L3M   Ave_Ct_L3M
##  Min.   :0.0000      Min.   :0.0000     Min.   :    2015
##  1st Qu.:0.0000      1st Qu.:0.6666     1st Qu.:    7238
##  Median :0.0000      Median :1.0000     Median :   12235
##  Mean   :0.2250      Mean   :0.7600     Mean   :   17895
##  3rd Qu.:0.3333      3rd Qu.:1.0000     3rd Qu.:   20499
##  Max.   :1.0000      Max.   :1.0000     Max.   :48557132
##    Ave_Dt_L3M         Ave_USSD_L3M     Ave_App_L3M       Ave_INET_L3M
##  Min.   :     841   Min.   :     0   Min.   :      0   Min.   :      0
##  1st Qu.:    7153   1st Qu.:     0   1st Qu.:      0   1st Qu.:      0
##  Median :   12115   Median :     0   Median :      0   Median :      0
##  Mean   :   17692   Mean   :  1289   Mean   :   3870   Mean   :   1530
##  3rd Qu.:   20343   3rd Qu.:  1360   3rd Qu.:   3533   3rd Qu.:      0
##  Max.   :47700317   Max.   :734770   Max.   :1371575   Max.   :3459692
```

```
##    Ave_DO_L3M         Ave_SO_L3M         Ave_CW_L3M      Ave_POS_L3M
##  Min.   :     0.0   Min.   :     0.0   Min.   :     0   Min.   :     0
##  1st Qu.:   232.3   1st Qu.:     0.0   1st Qu.:  1850   1st Qu.:  1084
##  Median :   970.0   Median :     0.0   Median :  3367   Median :  2406
##  Mean   :  2277.1   Mean   :   362.5   Mean   :  4430   Mean   :  3679
##  3rd Qu.:  2931.8   3rd Qu.:     0.0   3rd Qu.:  5700   3rd Qu.:  4700
##  Max.   :220000.0   Max.   :100000.0   Max.   :135333   Max.   :496173
##   Ave_CNP_L3M       Ave_Branch_L3M       Ave_Elec_L3M
##  Min.   :    0.00   Min.   :        0   Min.   :    0.00
##  1st Qu.:    0.00   1st Qu.:        0   1st Qu.:    0.00
##  Median :    0.00   Median :        0   Median :    0.00
##  Mean   :   53.65   Mean   :     2747   Mean   :   75.42
##  3rd Qu.:    0.00   3rd Qu.:        0   3rd Qu.:   60.00
##  Max.   :94919.70   Max.   :143018645   Max.   :13366.67
##   Ave_Air_L3M     Ratio_CW_POS_L3M    CW_Perc_L3M      POS_Perc_L3M
##  Min.   :   0.0   Min.   :   0.000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:  39.0   1st Qu.:   0.472   1st Qu.:0.1401   1st Qu.:0.0958
##  Median : 136.0   Median :   1.280   Median :0.2991   Median :0.1930
##  Mean   : 228.3   Mean   :   7.610   Mean   :0.3321   Mean   :0.2244
##  3rd Qu.: 303.0   3rd Qu.:   3.440   3rd Qu.:0.4922   3rd Qu.:0.3159
##  Max.   :8948.3   Max.   :6053.000   Max.   :1.0000   Max.   :1.0000
##   DO_Perc_L3M     INET_Perc_L3M      App_Perc_L3M    USSD_Perc_L3M
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0166   1st Qu.:0.00000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :0.0775   Median :0.00000   Median :0.0000   Median :0.00000
##  Mean   :0.1340   Mean   :0.02464   Mean   :0.1282   Mean   :0.05934
##  3rd Qu.:0.2046   3rd Qu.:0.00000   3rd Qu.:0.2200   3rd Qu.:0.06820
##  Max.   :0.9937   Max.   :1.00000   Max.   :1.0000   Max.   :0.99900
##  Elec_Perc_L3M       Air_Perc_L3M      CNP_Perc_L3M       SO_Perc_L3M
##  Min.   :0.000000   Min.   :0.00000   Min.   :0.000000   Min.   :0.00000
##  1st Qu.:0.000000   1st Qu.:0.00230   1st Qu.:0.000000   1st Qu.:0.00000
##  Median :0.000000   Median :0.01060   Median :0.000000   Median :0.00000
##  Mean   :0.003728   Mean   :0.01615   Mean   :0.001689   Mean   :0.02211
##  3rd Qu.:0.002400   3rd Qu.:0.02230   3rd Qu.:0.000000   3rd Qu.:0.00000
##  Max.   :0.469100   Max.   :0.55520   Max.   :0.995300   Max.   :0.95790
##  Branch_Perc_L3M  Other_Perc_L3M    CW_NumPerc_L3M   POS_NumPerc_L3M
##  Min.   :0.0000   Min.   :0.00000   Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.:0.0873   1st Qu.:0.1747
##  Median :0.0000   Median :0.02400   Median :0.1578   Median :0.3033
##  Mean   :0.0189   Mean   :0.07974   Mean   :0.1912   Mean   :0.3240
##  3rd Qu.:0.0000   3rd Qu.:0.12660   3rd Qu.:0.2580   3rd Qu.:0.4500
##  Max.   :0.9994   Max.   :0.93230   Max.   :1.0000   Max.   :1.0000
##  DO_NumPerc_L3M   INET_NumPerc_L3M   App_NumPerc_L3M  USSD_NumPerc_L3M
##  Min.   :0.0000   Min.   :0.000000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:0.0272   1st Qu.:0.000000   1st Qu.:0.0000   1st Qu.:0.00000
```

```
##  Median :0.0845   Median :0.000000   Median :0.0000   Median :0.00000
##  Mean   :0.1218   Mean   :0.009624   Mean   :0.1031   Mean   :0.04489
##  3rd Qu.:0.1733   3rd Qu.:0.000000   3rd Qu.:0.1703   3rd Qu.:0.06380
##  Max.   :0.9696   Max.   :1.000000   Max.   :1.0000   Max.   :0.96150
##  Elec_NumPerc_L3M  Air_NumPerc_L3M  CNP_NumPerc_L3M    SO_NumPerc_L3M
##  Min.   :0.00000   Min.   :0.0000   Min.   :0.000000   Min.   :0.00000
##  1st Qu.:0.00000   1st Qu.:0.0506   1st Qu.:0.000000   1st Qu.:0.00000
##  Median :0.00000   Median :0.1886   Median :0.000000   Median :0.00000
##  Mean   :0.01321   Mean   :0.2024   Mean   :0.003572   Mean   :0.01459
##  3rd Qu.:0.01470   3rd Qu.:0.3142   3rd Qu.:0.000000   3rd Qu.:0.00000
##  Max.   :0.70790   Max.   :0.9714   Max.   :0.937500   Max.   :0.80640
##  Branch_NumPerc_L3M CSWEEP_P90_NumPerc_L3M CSWEEP_P80_NumPerc_L3M
##  Min.   :0.000000   Min.   :0.0000         Min.   :0.0000
##  1st Qu.:0.000000   1st Qu.:0.0000         1st Qu.:0.0000
##  Median :0.000000   Median :0.0400         Median :0.1250
##  Mean   :0.002192   Mean   :0.1064         Mean   :0.1866
##  3rd Qu.:0.000000   3rd Qu.:0.1538         3rd Qu.:0.2857
##  Max.   :0.439000   Max.   :1.0000         Max.   :1.0000
##  CSWEEP_P70_NumPerc_L3M  CW_Util_L3M     Time_Since_USSD_L3M
##  Min.   :0.0000         Min.   :0.0000   Min.   :0.0000
##  1st Qu.:0.0476         1st Qu.:0.1278   1st Qu.:0.0000
##  Median :0.1935         Median :0.2750   Median :1.0000
##  Mean   :0.2507         Mean   :0.3125   Mean   :0.6531
##  3rd Qu.:0.3888         3rd Qu.:0.4598   3rd Qu.:1.0000
##  Max.   :1.0000         Max.   :5.2903   Max.   :1.0000
##  Time_Since_App_L3M Time_Since_INET_L3M Time_Since_DO_L3M
##  Min.   :0.0000     Min.   :0.0000      Min.   :0.0000
##  1st Qu.:0.0000     1st Qu.:1.0000      1st Qu.:0.0000
##  Median :1.0000     Median :1.0000      Median :0.0000
##  Mean   :0.5586     Mean   :0.8944      Mean   :0.1887
##  3rd Qu.:1.0000     3rd Qu.:1.0000      3rd Qu.:0.0000
##  Max.   :1.0000     Max.   :1.0000      Max.   :1.0000
##  Time_Since_SO_L3M Time_Since_CW_L3M Time_Since_POS_L3M Time_Since_CNP_L3M
##  Min.   :0.0000    Min.   :0.00000   Min.   :0.00000    Min.   :0.0000
##  1st Qu.:1.0000    1st Qu.:0.00000   1st Qu.:0.00000    1st Qu.:1.0000
##  Median :1.0000    Median :0.00000   Median :0.00000    Median :1.0000
##  Mean   :0.7724    Mean   :0.03517   Mean   :0.05076    Mean   :0.9516
##  3rd Qu.:1.0000    3rd Qu.:0.00000   3rd Qu.:0.00000    3rd Qu.:1.0000
##  Max.   :1.0000    Max.   :1.00000   Max.   :1.00000    Max.   :1.0000
##  Time_Since_Branch_L3M Time_Since_Elec_L3M Time_Since_Air_L3M
##  Min.   :0.0000        Min.   :0.0000      Min.   :0.0000
##  1st Qu.:1.0000        1st Qu.:0.3333      1st Qu.:0.0000
##  Median :1.0000        Median :1.0000      Median :0.0000
##  Mean   :0.9262        Mean   :0.7182      Mean   :0.2072
##  3rd Qu.:1.0000        3rd Qu.:1.0000      3rd Qu.:0.0000
```

```
##  Max.   :1.0000        Max.   :1.0000      Max.   :1.0000
##  InsufFunds_L3M      Naedo_L3M       DO_InsufFunds_L3M DO_Dispute_L3M
##  Min.   : 0.0000   Min.   :  0.000   Min.   : 0.000    Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.:  0.000   1st Qu.: 0.000    1st Qu.: 0.0000
##  Median : 0.0000   Median :  3.000   Median : 0.000    Median : 0.0000
##  Mean   : 0.4632   Mean   :  4.577   Mean   : 0.899    Mean   : 0.7891
##  3rd Qu.: 0.0000   3rd Qu.:  7.000   3rd Qu.: 1.000    3rd Qu.: 0.0000
##  Max.   :51.0000   Max.   :163.000   Max.   :63.000    Max.   :45.0000
##   Num_Loan_L3M     Num_Loan_Months_L3M Quality_Banking_L3M
##  Min.   :  0.000   Min.   :0.000       Min.   :0.0000
##  1st Qu.:  0.000   1st Qu.:0.000       1st Qu.:1.0000
##  Median :  0.000   Median :0.000       Median :1.0000
##  Mean   :  1.863   Mean   :0.338       Mean   :0.9133
##  3rd Qu.:  0.000   3rd Qu.:0.000       3rd Qu.:1.0000
##  Max.   :317.000   Max.   :3.000       Max.   :1.0000
##  Ave_Days_Above_10_L3M Ave_Days_Above_100_L3M Ave_Days_Above_500_L3M
##  Min.   :0.0500        Min.   :0.0000         Min.   :0.0000
##  1st Qu.:1.0000        1st Qu.:0.3804         1st Qu.:0.2065
##  Median :1.0000        Median :0.6195         Median :0.3913
##  Mean   :0.9893        Mean   :0.6065         Mean   :0.4391
##  3rd Qu.:1.0000        3rd Qu.:0.8586         3rd Qu.:0.6413
##  Max.   :1.0000        Max.   :1.0000         Max.   :1.0000
##  Ave_Days_Above_1000_L3M Ave_Days_Above_2500_L3M Ave_Days_Above_5000_L3M
##  Min.   :0.0000          Min.   :0.0000          Min.   :0.0000
##  1st Qu.:0.1304          1st Qu.:0.0326          1st Qu.:0.0000
##  Median :0.2608          Median :0.1195          Median :0.0326
##  Mean   :0.3400          Mean   :0.2028          Mean   :0.1167
##  3rd Qu.:0.4891          3rd Qu.:0.2717          3rd Qu.:0.1304
##  Max.   :1.0000          Max.   :1.0000          Max.   :1.0000
##  Ave_Days_Above_10000_L3M Ave_Days_Above_20000_L3M
##  Min.   :0.00000          Min.   :0.00000
##  1st Qu.:0.00000          1st Qu.:0.00000
##  Median :0.00000          Median :0.00000
##  Mean   :0.05834          Mean   :0.02684
##  3rd Qu.:0.03260          3rd Qu.:0.00000
##  Max.   :1.00000          Max.   :1.00000
##  Ave_Days_Above_50000_L3M Avg_Dep_Bal_L3M   Dep_Bal_Spend_Ratio_L3M
##  Min.   :0.000000         Min.   :   -499   Min.   :-36.0220
##  1st Qu.:0.000000         1st Qu.:    453   1st Qu.:  0.9757
##  Median :0.000000         Median :    984   Median :  0.9901
##  Mean   :0.009734         Mean   :   3518   Mean   :  0.9607
##  3rd Qu.:0.000000         3rd Qu.:   2208   3rd Qu.:  0.9950
##  Max.   :1.000000         Max.   :5905004   Max.   : 10.5341
##  AVG_TTSB90_L3M  AVG_TTSB75_L3M   AVG_TTSB50_L3M   AVG_TTSB25_L3M
##  Min.   : 0.00   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
```

```
##  1st Qu.: 8.00   1st Qu.: 4.000   1st Qu.: 2.000   1st Qu.: 1.000
##  Median :12.00   Median : 7.000   Median : 4.000   Median : 2.000
##  Mean   :13.44   Mean   : 8.957   Mean   : 5.171   Mean   : 2.804
##  3rd Qu.:18.00   3rd Qu.:12.000   3rd Qu.: 6.000   3rd Qu.: 3.000
##  Max.   :30.00   Max.   :30.000   Max.   :30.000   Max.   :30.000
##  Ratio_Out_Above_1k_L3M Ratio_Out_Above_5k_L3M Ratio_Out_Above_10k_L3M
##  Min.   :0.0000         Min.   :0.000000       Min.   :0.000000
##  1st Qu.:0.0588         1st Qu.:0.000000       1st Qu.:0.000000
##  Median :0.1075         Median :0.000000       Median :0.000000
##  Mean   :0.1300         Mean   :0.007723       Mean   :0.001135
##  3rd Qu.:0.1756         3rd Qu.:0.000000       3rd Qu.:0.000000
##  Max.   :1.0000         Max.   :0.821400       Max.   :0.470500
##  Ratio_Out_Above_20k_L3M Ratio_In_Above_1k_L3M Ratio_In_Above_5k_L3M
##  Min.   :0.0000000       Min.   :0.0000        Min.   :0.0000
##  1st Qu.:0.0000000       1st Qu.:0.1250        1st Qu.:0.0000
##  Median :0.0000000       Median :0.2500        Median :0.0357
##  Mean   :0.0006073       Mean   :0.3052        Mean   :0.1210
##  3rd Qu.:0.0000000       3rd Qu.:0.4285        3rd Qu.:0.1666
##  Max.   :0.5172000       Max.   :1.0000        Max.   :1.0000
##  Ratio_In_Above_10k_L3M Ratio_In_Above_20k_L3M
##  Min.   :0.00000        Min.   :0.00000
##  1st Qu.:0.00000        1st Qu.:0.00000
##  Median :0.00000        Median :0.00000
##  Mean   :0.08419        Mean   :0.02939
##  3rd Qu.:0.12500        3rd Qu.:0.00000
##  Max.   :1.00000        Max.   :1.00000
```

Appendix E

In Figure 50 to Figure 57 we find the histograms of the 127 variables of the 3M_SAMP1 sample. These variables have been log-transformed and the outliers have been removed. The next step is to start the dimension reduction process.

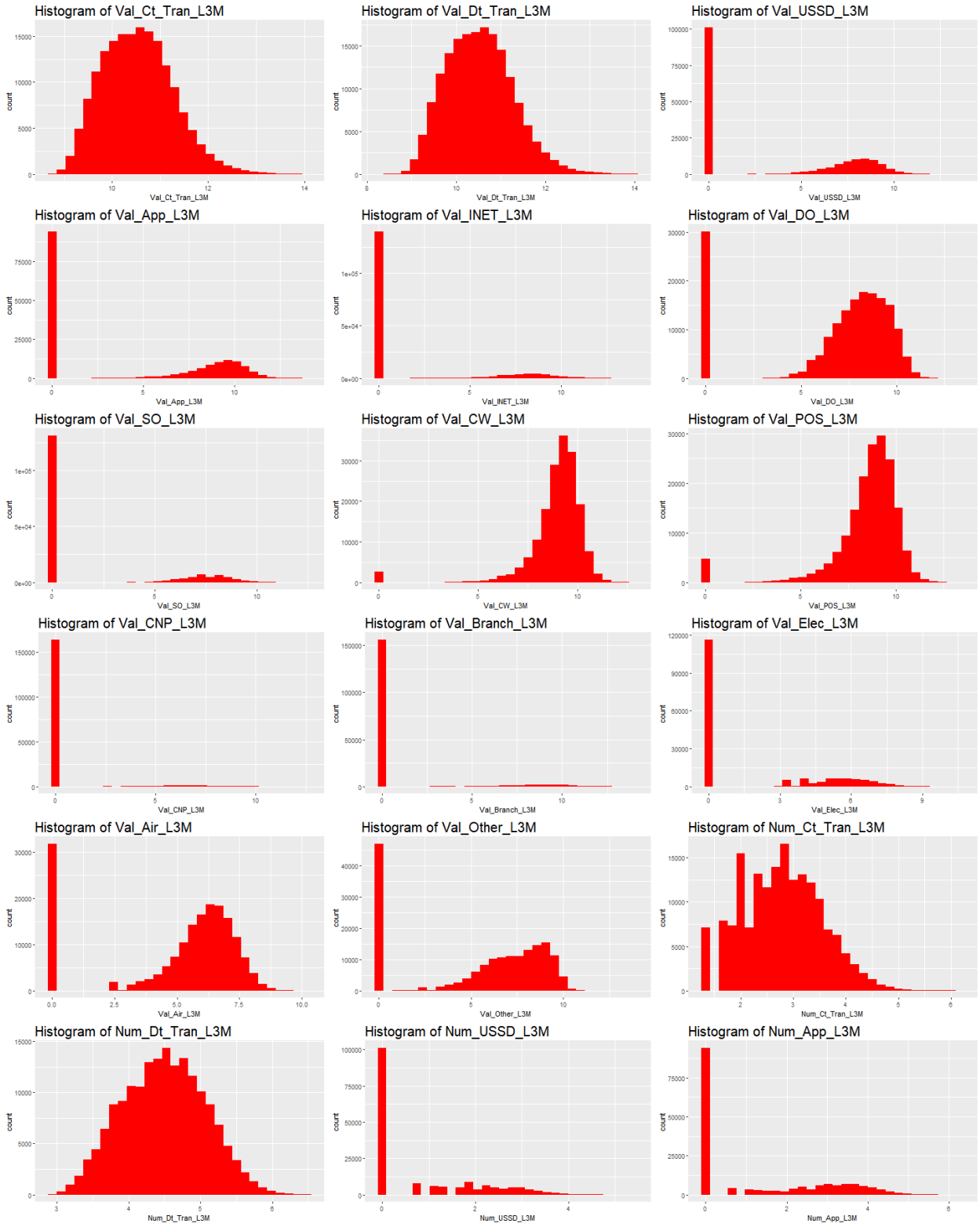


Figure 50: Variable 1 to 15

Figure 51: Variable 19 to 36

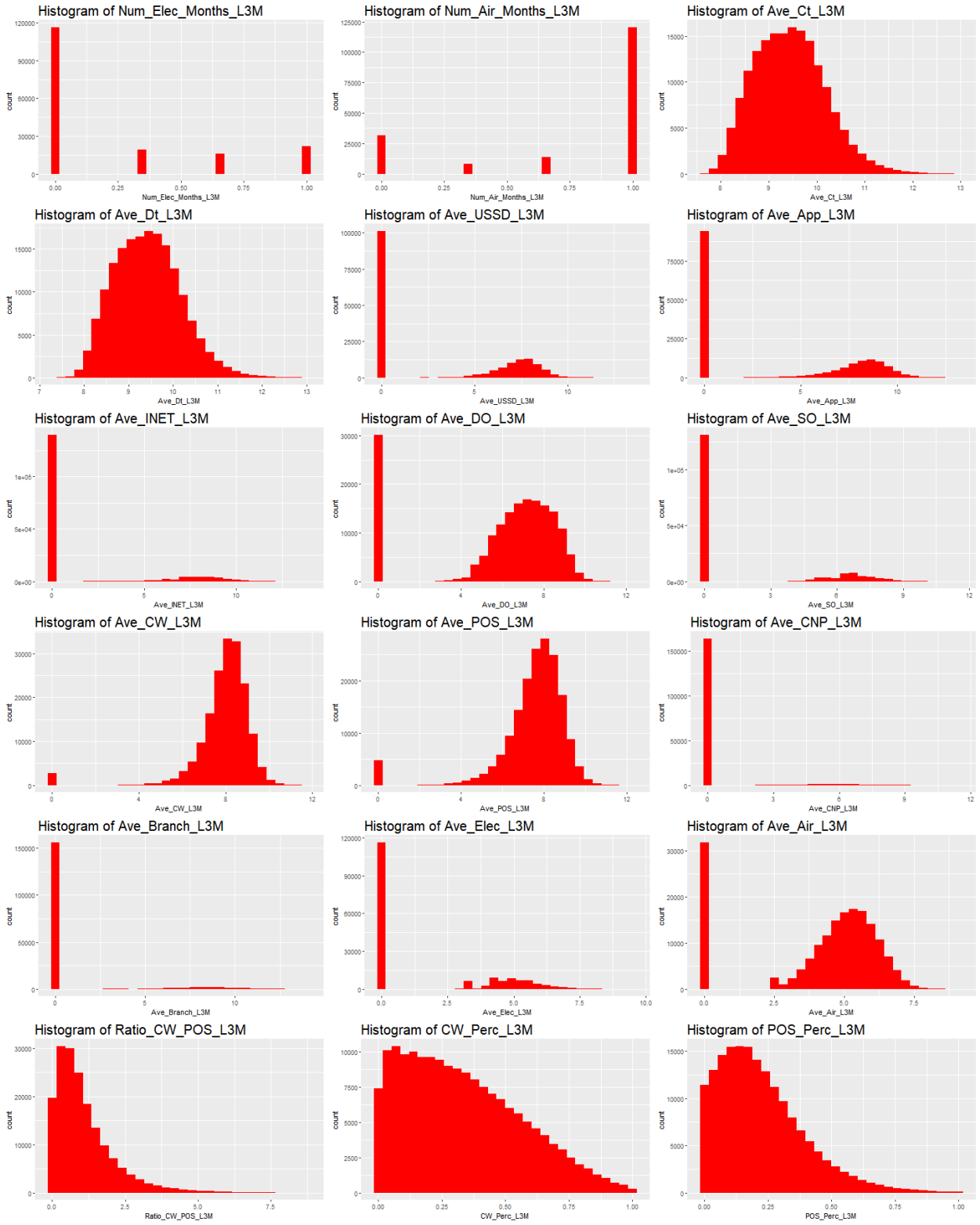


Figure 52: Variable 37 to 54

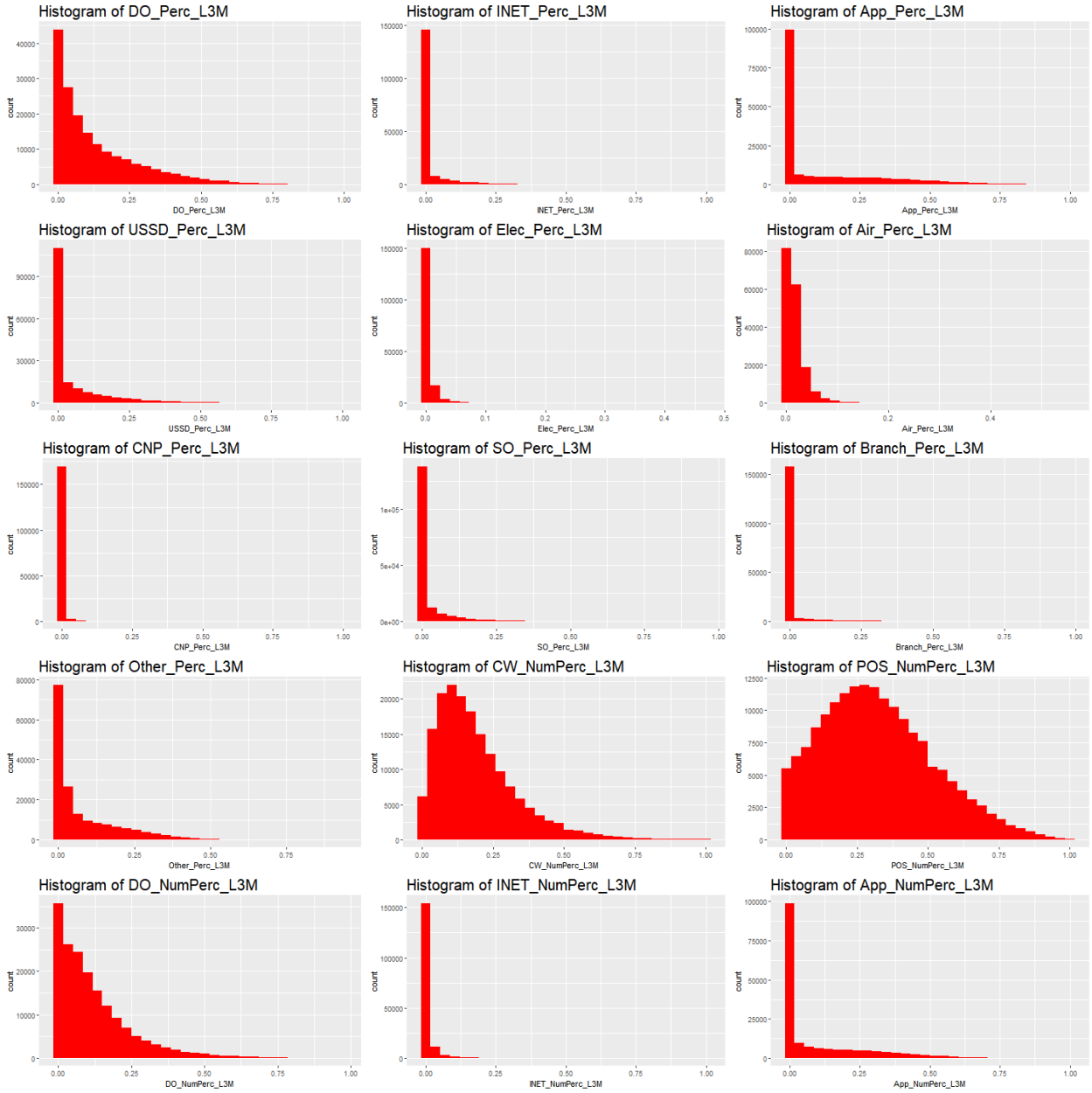


Figure 53: Variable 55 to 69

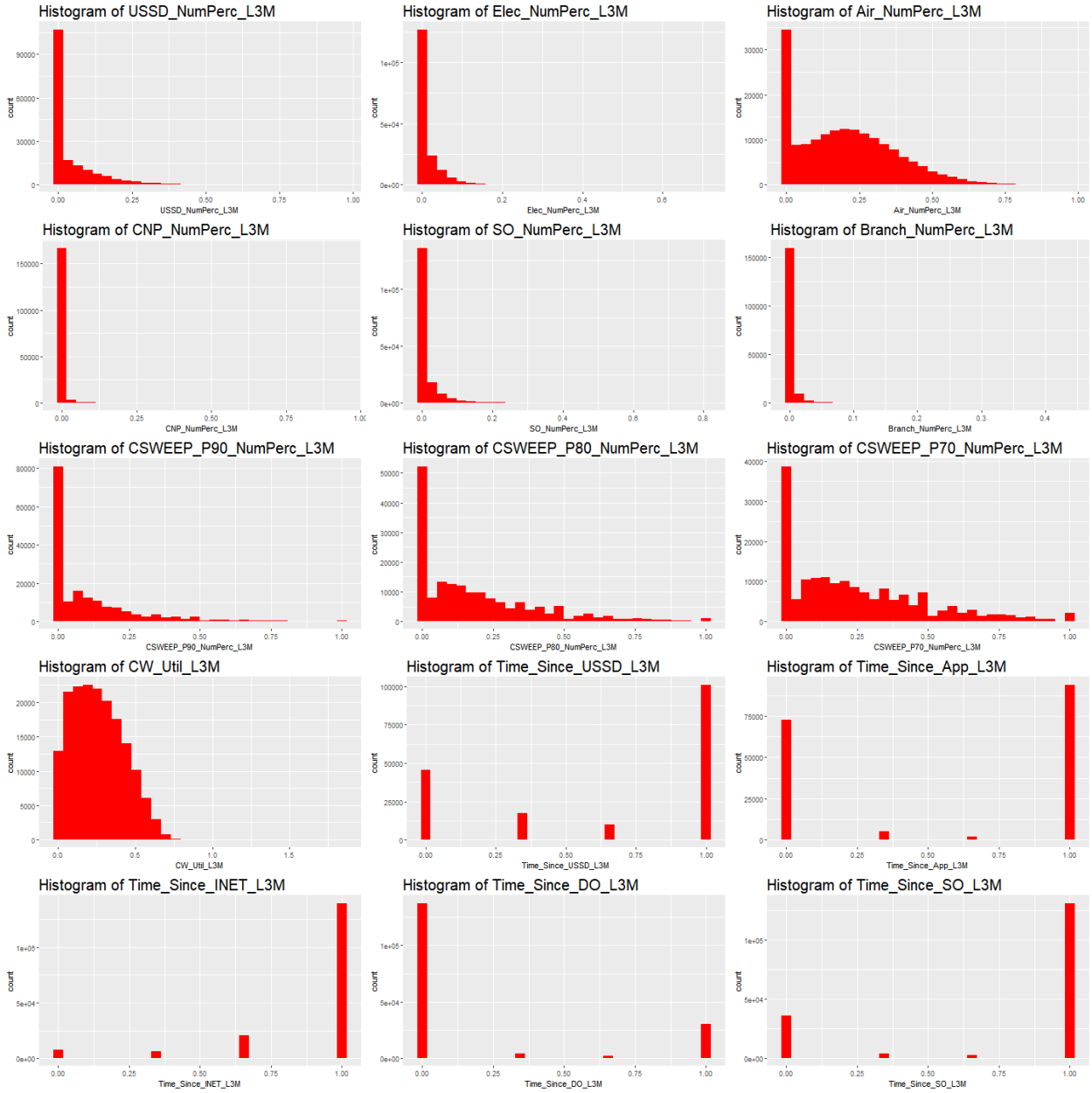


Figure 54: Variable 70 to 84

Figure 55: Variable 85 to 99

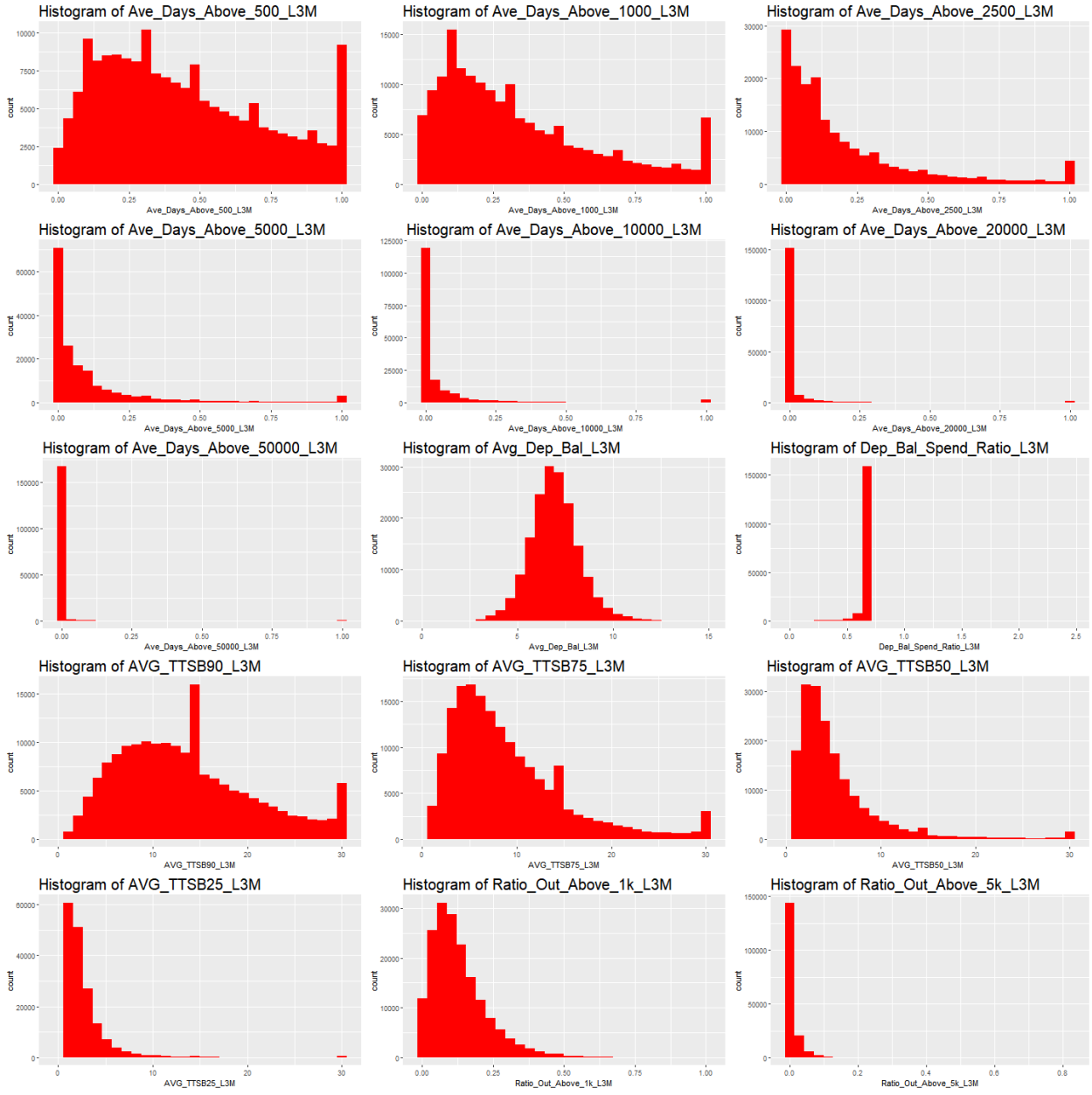


Figure 56: Variable 100 to 114

Figure 57: Variable 115 to 127

Appendix F

Here a list of the percentage zero values for each variable in dataset 3M_SAMP1 can be seen. The variables have not been cleaned and can be seen in their raw states, containing outliers and on the original measurement scales.

	Vars	p_zeros
1	Val_Ct_Tran_L3M	0
2	Val_Dt_Tran_L3M	0
3	Val_USSD_L3M	58.22
4	Val_App_L3M	54.16
5	Val_INET_L3M	80.43
6	Val_DO_L3M	17.34
7	Val_SO_L3M	75.43
8	Val_CW_L3M	1.56
9	Val_POS_L3M	2.7
10	Val_CNP_L3M	94.42
11	Val_Branch_L3M	89.86
12	Val_Elec_L3M	66.99
13	Val_Air_L3M	18.27
14	Val_Other_L3M	26.98
15	Num_Ct_Tran_L3M	0
16	Num_Dt_Tran_L3M	0
17	Num_USSD_L3M	58.22
18	Num_App_L3M	54.16
19	Num_INET_L3M	80.43
20	Num_DO_L3M	17.34
21	Num_SO_L3M	75.43
22	Num_CW_L3M	1.56
23	Num_POS_L3M	2.7
24	Num_CNP_L3M	94.42
25	Num_Branch_L3M	89.86
26	Num_Elec_L3M	66.99
27	Num_Air_L3M	18.27
28	Num_USSD_Months_L3M	58.22
29	Num_App_Months_L3M	54.16
30	Num_INET_Months_L3M	80.43
31	Num_DO_Months_L3M	17.34
32	Num_SO_Months_L3M	75.43
33	Num_CW_Months_L3M	1.56
34	Num_POS_Months_L3M	2.7
35	Num_CNP_Months_L3M	94.42
36	Num_Branch_Months_L3M	89.86
37	Num_Elec_Months_L3M	66.99
38	Num_Air_Months_L3M	18.27

```
## 39                Ave_Ct_L3M       0
## 40                Ave_Dt_L3M       0
## 41              Ave_USSD_L3M   58.22
## 42               Ave_App_L3M   54.16
## 43              Ave_INET_L3M   80.43
## 44                Ave_DO_L3M   17.34
## 45                Ave_SO_L3M   75.43
## 46                Ave_CW_L3M    1.56
## 47               Ave_POS_L3M     2.7
## 48               Ave_CNP_L3M   94.42
## 49            Ave_Branch_L3M   89.86
## 50              Ave_Elec_L3M   66.99
## 51               Ave_Air_L3M   18.27
## 52          Ratio_CW_POS_L3M    4.13
## 53               CW_Perc_L3M    1.56
## 54              POS_Perc_L3M    2.71
## 55               DO_Perc_L3M   17.35
## 56             INET_Perc_L3M   80.53
## 57              App_Perc_L3M    54.2
## 58             USSD_Perc_L3M   58.23
## 59             Elec_Perc_L3M   66.99
## 60              Air_Perc_L3M   18.31
## 61              CNP_Perc_L3M   94.45
## 62               SO_Perc_L3M   75.46
## 63           Branch_Perc_L3M   89.88
## 64            Other_Perc_L3M   27.04
## 65            CW_NumPerc_L3M    1.56
## 66           POS_NumPerc_L3M     2.7
## 67            DO_NumPerc_L3M   17.34
## 68          INET_NumPerc_L3M   80.43
## 69           App_NumPerc_L3M   54.16
## 70          USSD_NumPerc_L3M   58.22
## 71          Elec_NumPerc_L3M   66.99
## 72           Air_NumPerc_L3M   18.27
## 73           CNP_NumPerc_L3M   94.42
## 74            SO_NumPerc_L3M   75.43
## 75       Branch_NumPerc_L3M   89.86
## 76    CSWEEP_P90_NumPerc_L3M   46.51
## 77    CSWEEP_P80_NumPerc_L3M   29.99
## 78    CSWEEP_P70_NumPerc_L3M   22.23
## 79               CW_Util_L3M    1.56
## 80       Time_Since_USSD_L3M   26.12
## 81        Time_Since_App_L3M   41.81
## 82       Time_Since_INET_L3M    4.24
## 83         Time_Since_DO_L3M   79.17
```

```
## 84          Time_Since_SO_L3M   20.71
## 85          Time_Since_CW_L3M   93.63
## 86         Time_Since_POS_L3M   91.47
## 87         Time_Since_CNP_L3M    3.97
## 88      Time_Since_Branch_L3M     4.4
## 89        Time_Since_Elec_L3M   22.22
## 90         Time_Since_Air_L3M   76.08
## 91             InsufFunds_L3M   77.44
## 92                  Naedo_L3M   36.16
## 93          DO_InsufFunds_L3M   72.19
## 94             DO_Dispute_L3M   79.75
## 95               Num_Loan_L3M    83.9
## 96        Num_Loan_Months_L3M    83.9
## 97        Quality_Banking_L3M    8.67
## 98      Ave_Days_Above_10_L3M       0
## 99     Ave_Days_Above_100_L3M    0.18
## 100    Ave_Days_Above_500_L3M    0.76
## 101   Ave_Days_Above_1000_L3M    2.49
## 102   Ave_Days_Above_2500_L3M   12.26
## 103   Ave_Days_Above_5000_L3M   33.53
## 104 Ave_Days_Above_10000_L3M   62.28
## 105 Ave_Days_Above_20000_L3M   83.92
## 106 Ave_Days_Above_50000_L3M    95.8
## 107           Avg_Dep_Bal_L3M    0.01
## 108  Dep_Bal_Spend_Ratio_L3M    0.01
## 109            AVG_TTSB90_L3M       0
## 110            AVG_TTSB75_L3M       0
## 111            AVG_TTSB50_L3M       0
## 112            AVG_TTSB25_L3M    0.01
## 113   Ratio_Out_Above_1k_L3M    4.66
## 114   Ratio_Out_Above_5k_L3M    77.2
## 115  Ratio_Out_Above_10k_L3M   94.52
## 116  Ratio_Out_Above_20k_L3M   96.89
## 117    Ratio_In_Above_1k_L3M   13.04
## 118    Ratio_In_Above_5k_L3M   47.43
## 119   Ratio_In_Above_10k_L3M   59.26
## 120   Ratio_In_Above_20k_L3M   80.63
## 121              Flag_CNP_L3M   94.42
## 122           Flag_Branch_L3M   89.86
## 123               Flag_SO_L3M   75.43
## 124             Flag_INET_L3M   80.43
## 125       Flag_InsufFunds_L3M   77.44
## 126   Flag_DO_InsufFunds_L3M   72.19
## 127      Flag_DO_Dispute_L3M   79.75
```

Appendix G

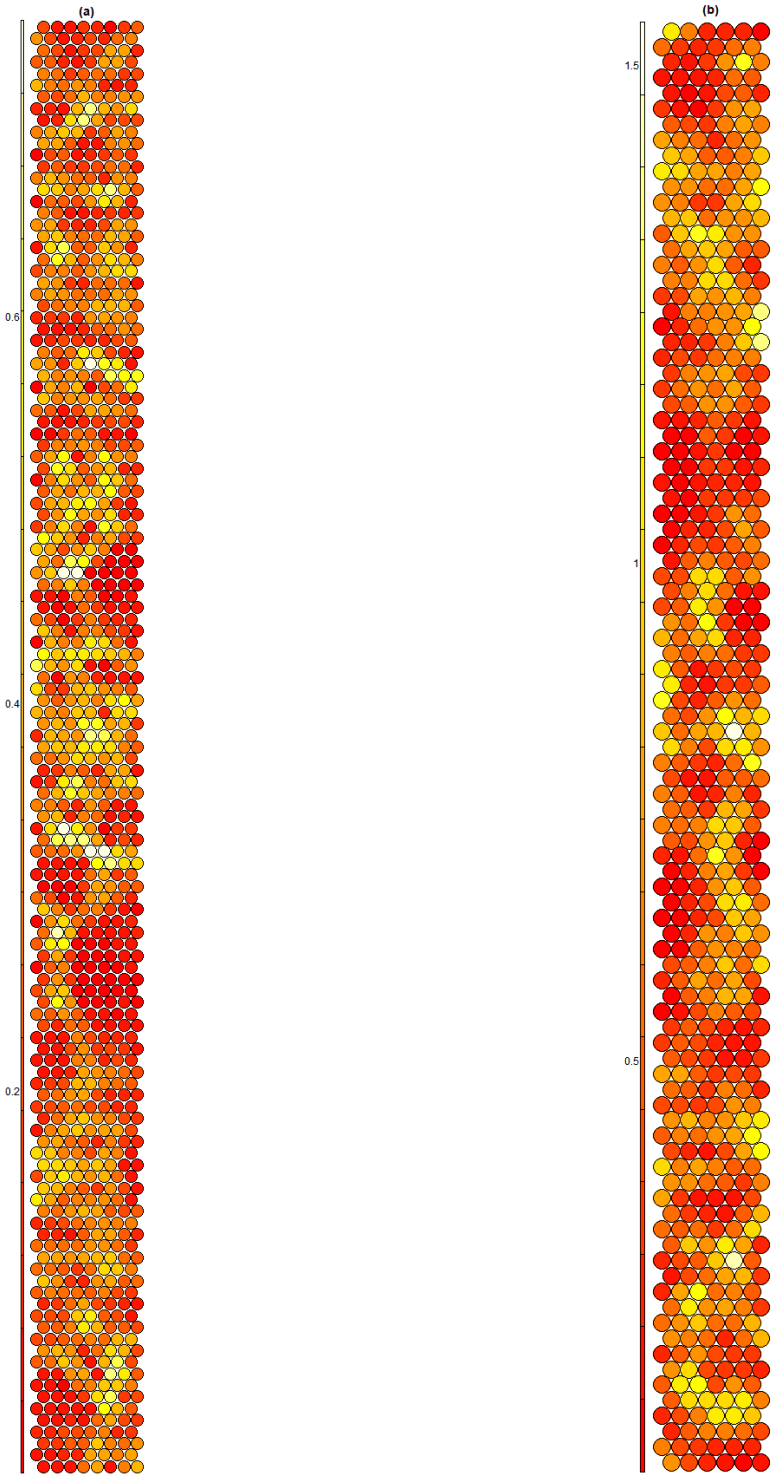


Figure 58: U-matrices of (a) the SOM of 3M_SAMP2 and (b) the SOM of 3M_SAMP3

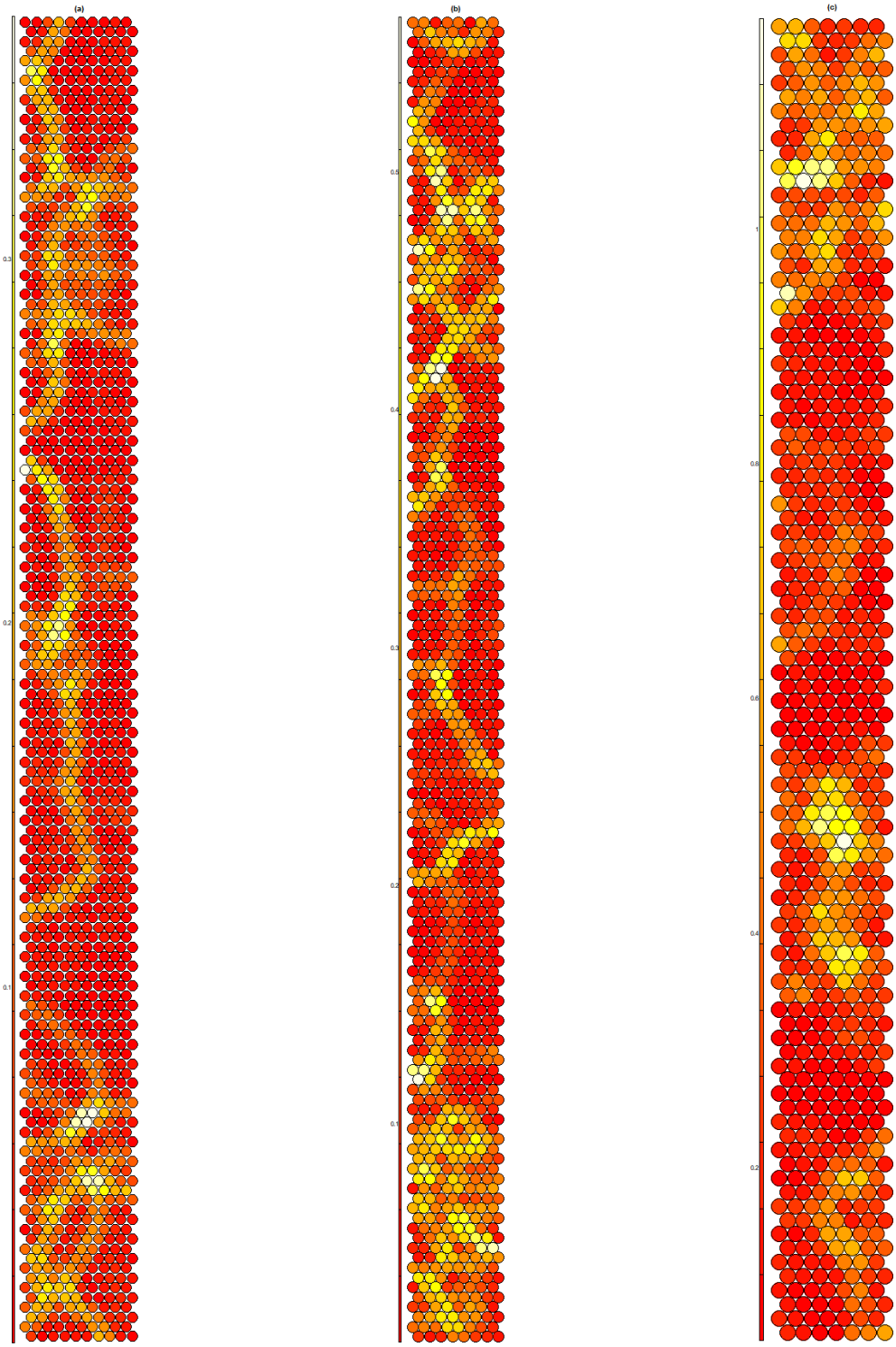


Figure 59: U-Matrix of (a) the SOM of 6M_SAMP1, (b) the SOM of 6M_SAMP2 and (c) the SOM of 6M_SAMP3

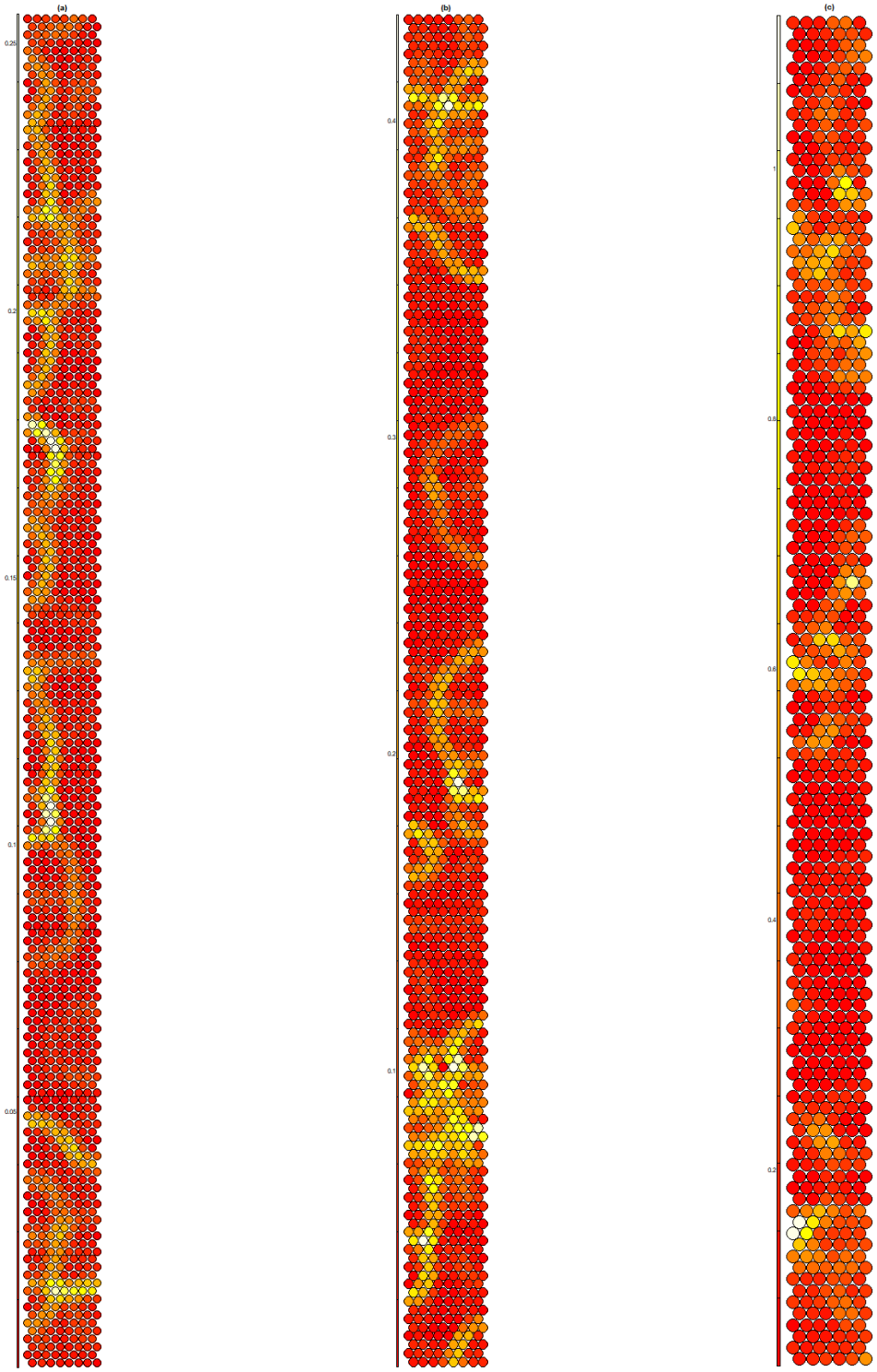


Figure 60: U-Matrix of (a) the SOM of 12M_SAMP1, (b) the SOM of 12M_SAMP2 and (c) the SOM of 12M_SAMP3

Appendix H

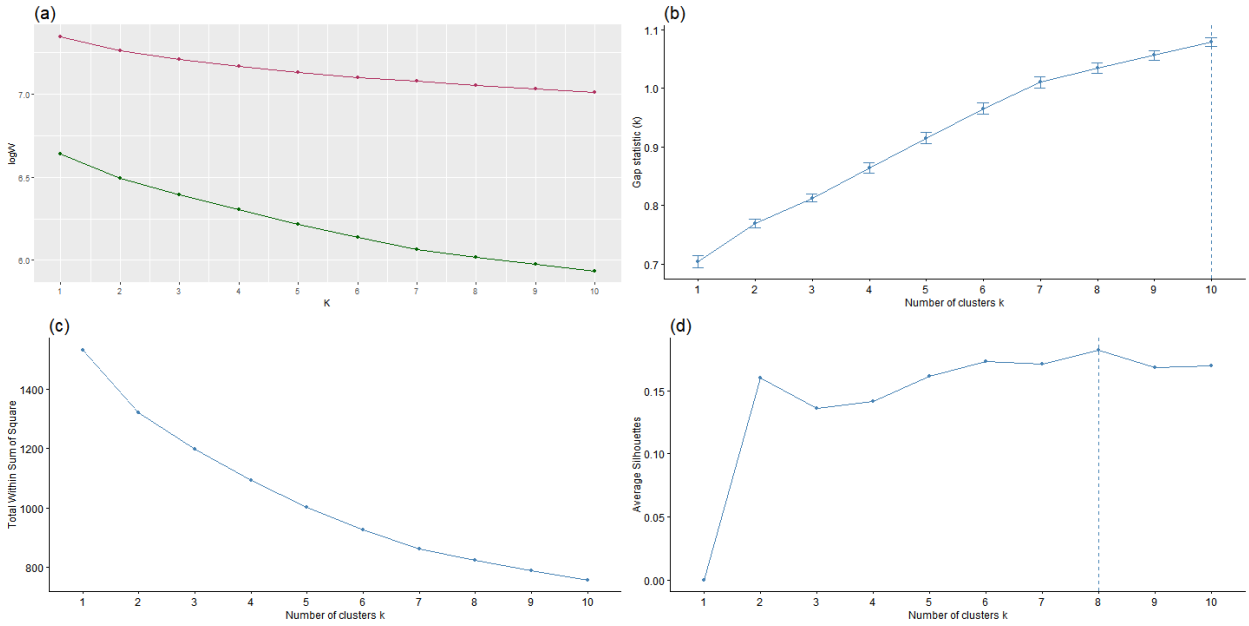


Figure 61: Gap Statistic, Elbow method and Average Silhouette method plot for the 3M_SAMP2 dataset

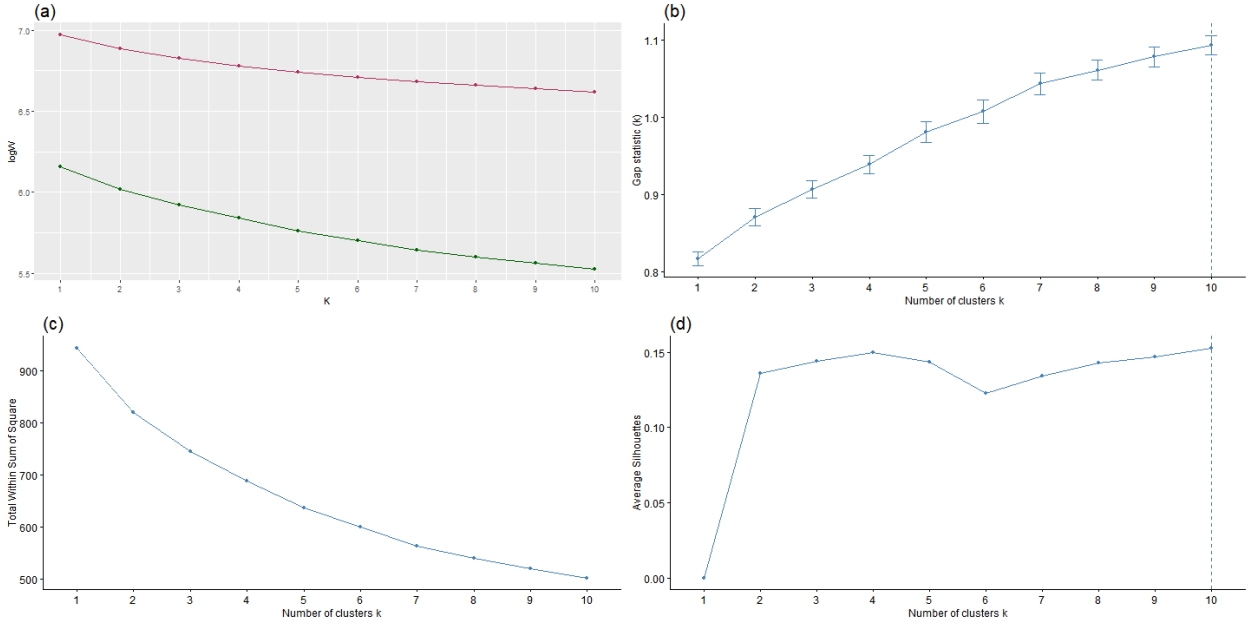


Figure 62: Gap Statistic, Elbow method and Average Silhouette method plot for the 3M_SAMP3 dataset

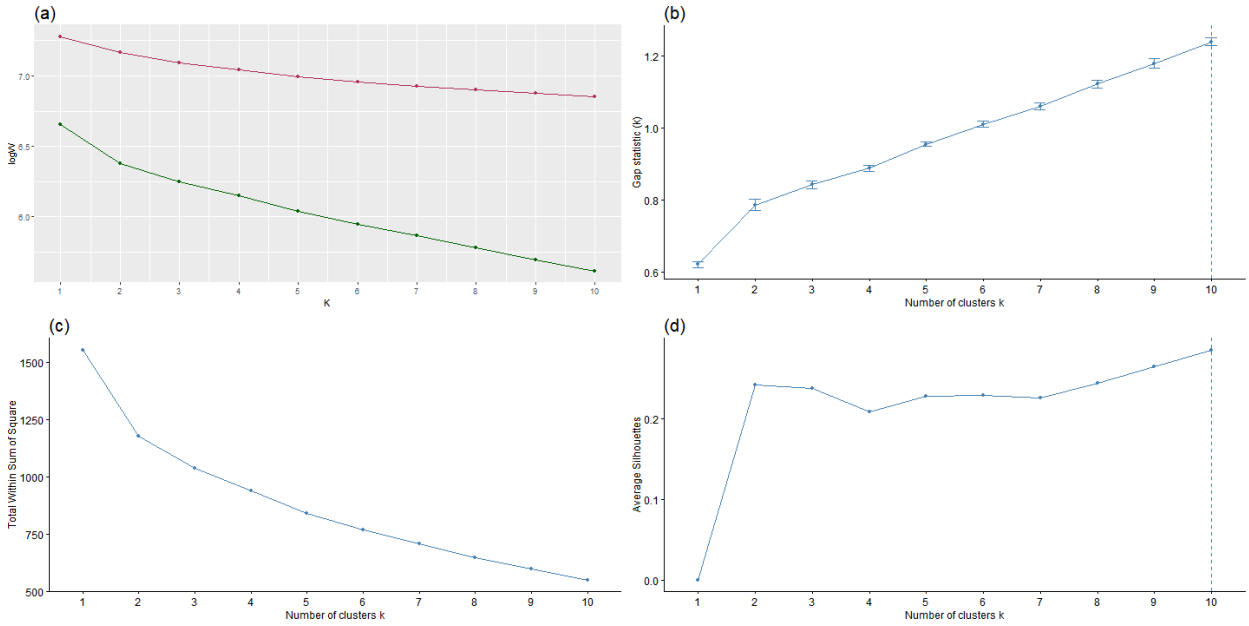


Figure 63: Gap Statistic, Elbow method and Average Silhouette method plot for the 6M_SAMP1 dataset

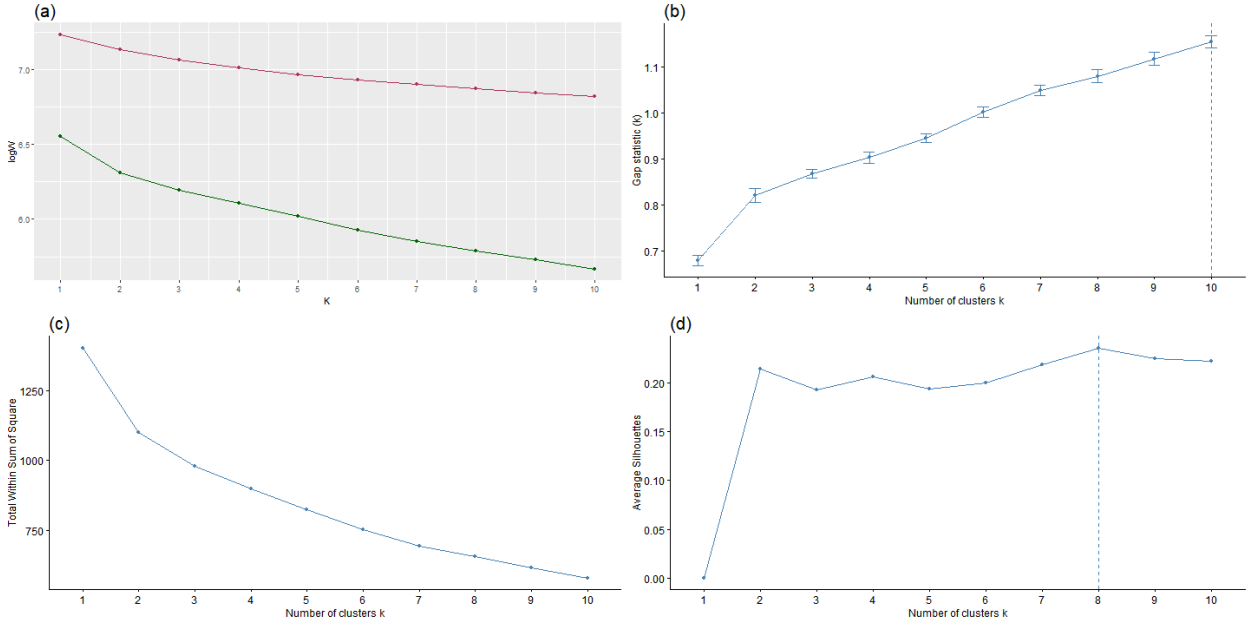


Figure 64: Gap Statistic, Elbow method and Average Silhouette method plot for the 6M_SAMP2 dataset

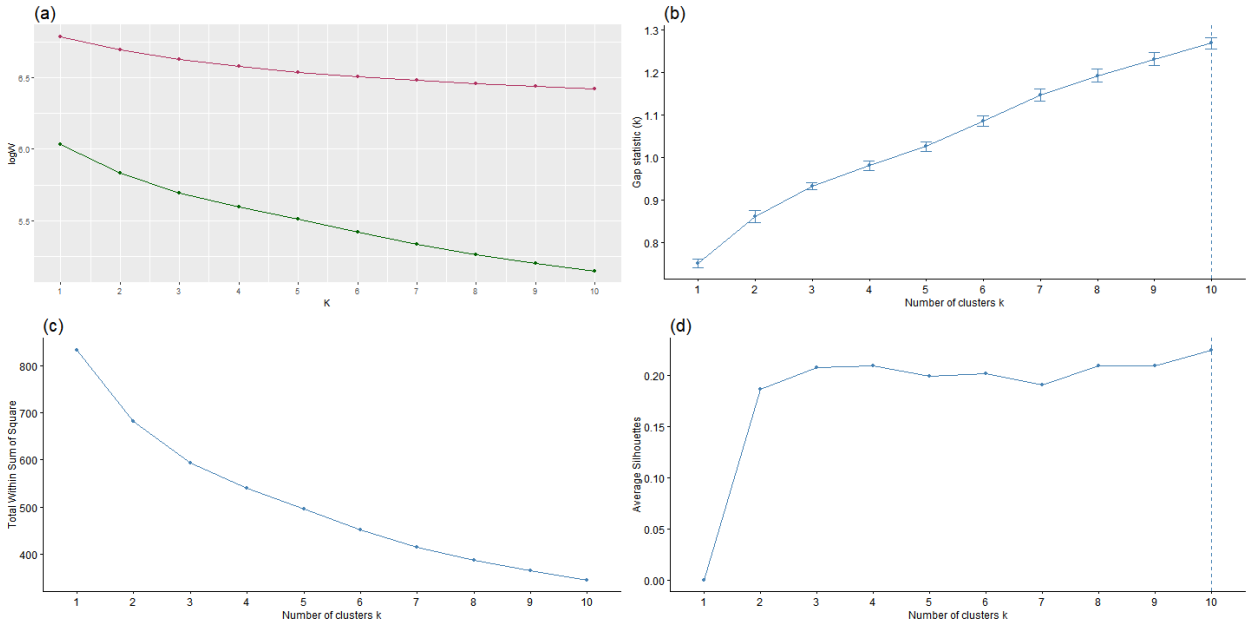


Figure 65: Gap Statistic, Elbow method and Average Silhouette method plot for the 6M_SAMP3 dataset

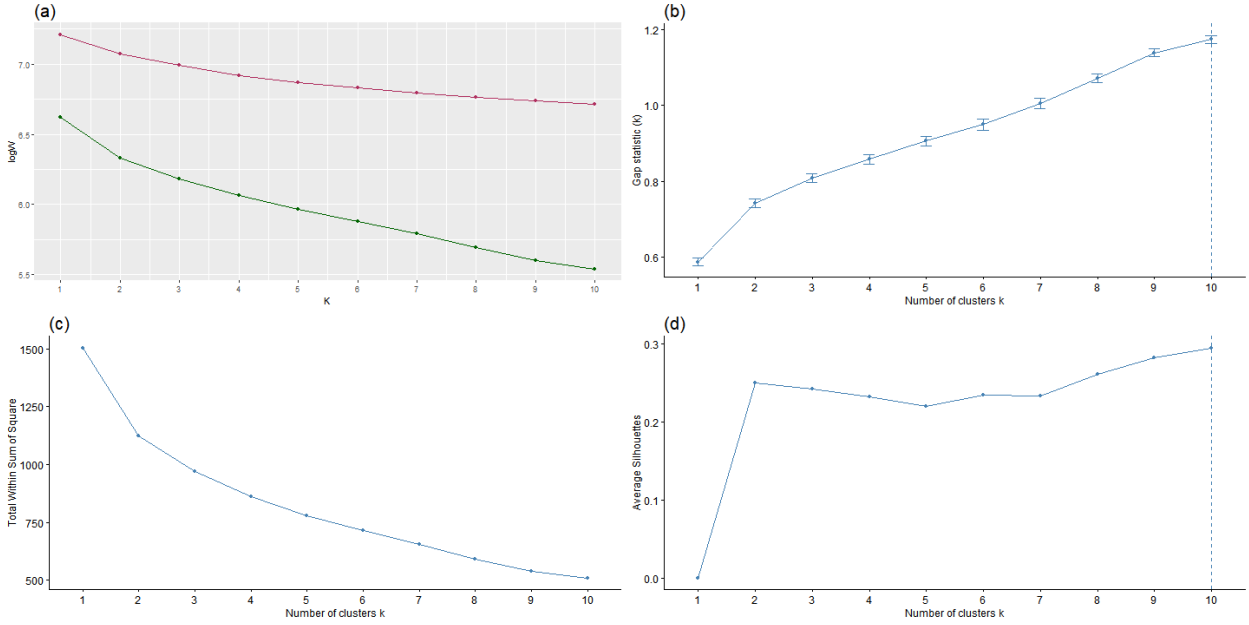


Figure 66: Gap Statistic, Elbow method and Average Silhouette method plot for the 12M_SAMP1 dataset

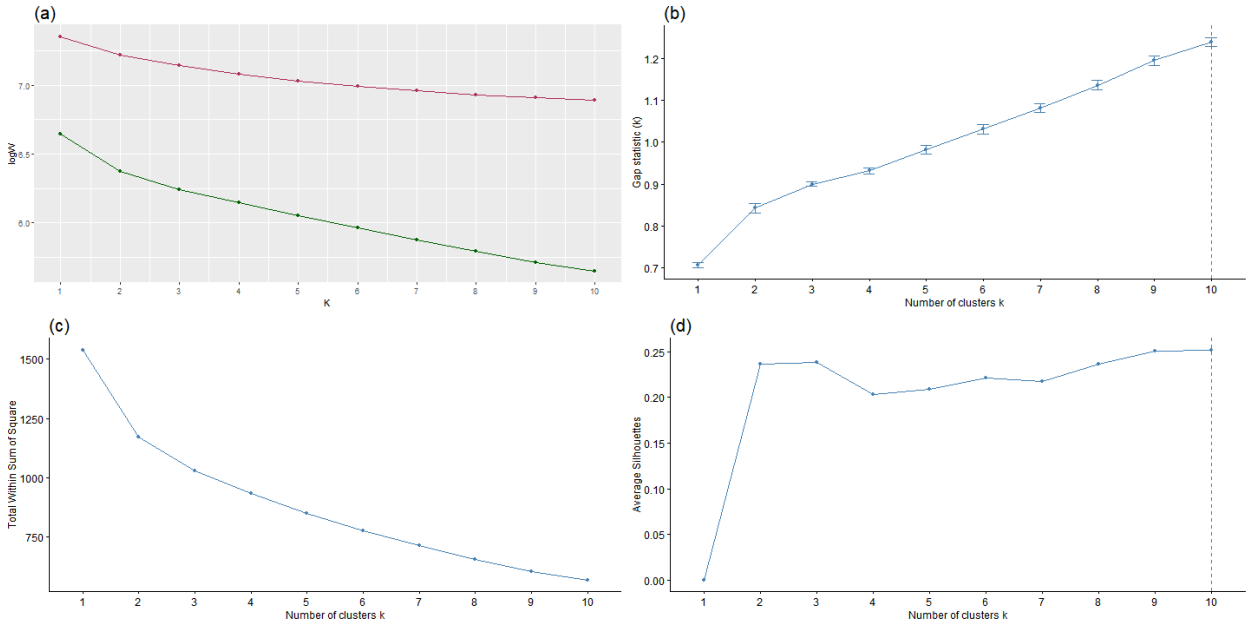


Figure 67: Gap Statistic, Elbow method and Average Silhouette method plot for the 12M_SAMP2 dataset

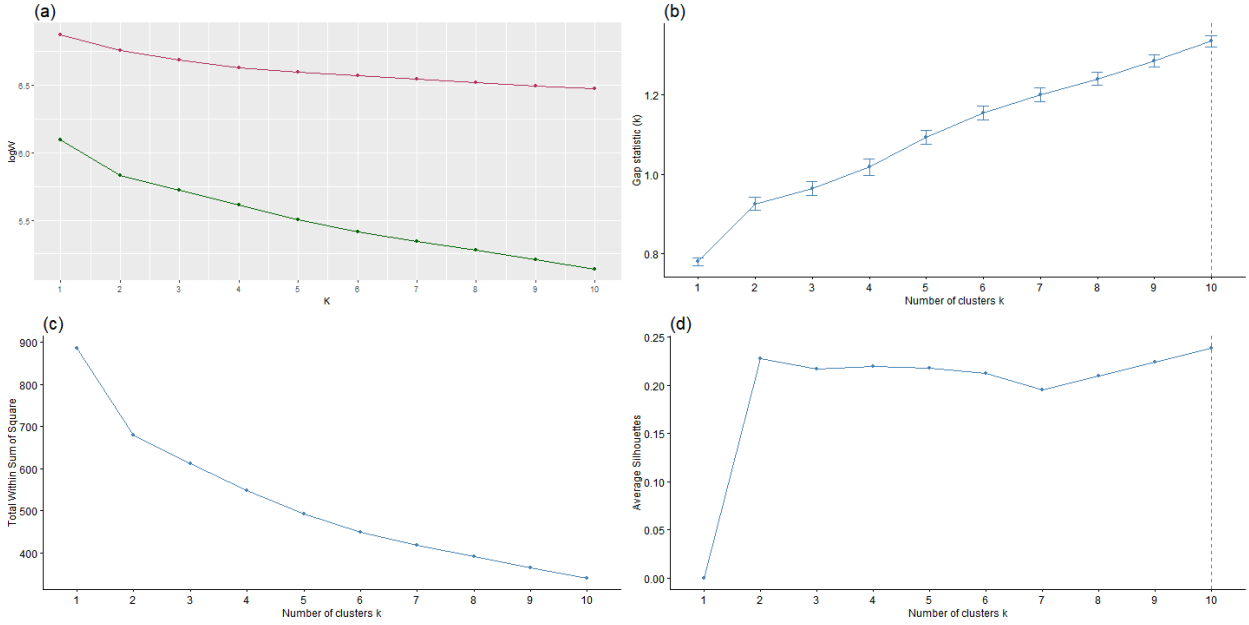


Figure 68: Gap Statistic, Elbow method and Average Silhouette method plot for the 12M_SAMP3 dataset

Appendix I

From Figure 69 to Figure 78 the following cluster traits were extracted.

Cluster 1:

- Clients make very little use of Debit Orders, NAEDOs and Stop Orders
- Clients rely less on cash withdrawals and cash usage
- Clients frequently buy Airtime
- Clients have lower inflow values
- Cluster has the least Quality Banking clients

This cluster has many similarities to cluster 1 of 3M_SAMP1.

Cluster 2:

- Clients have lower inflow values and a lower number of inflow transactions
- Clients least frequently buy electricity and airtime
- Clients least frequently use the Banking Application or Card Not Present transactions

These clients have similarities with cluster 5 of 3M_SAMP1. These clients may possibly not be very financially active or be constrained by low income.

Cluster 3:

- Clients make the most use of Debit Orders, but have the least Debit Order Insufficient Fund flags
- Cluster has the most Quality Banking clients
- Clients make use of Branch visits
- Clients have the least Internet Banking usage

The clients have similarities to cluster 4 of 3M_SAMP1.

Cluster 4:

- Clients have high value and frequent inflows
- Clients have a high frequency of performing App, INET, POS, CNP and Airtime transactions
- Clients have a very low USSD and cash usage
- Clients make the least use of branch transactions
- Cluster has the many Quality Banking clients

These client have similarities to cluster 2 of 3M_SAMP1.

Cluster 5:

- Clients make the most use of NAEDOs and very frequent use of Debit Orders
- Clients also have a high number DO Insufficient Funds flags and also DO Disputes
- Client make very little use of POS transactions

This has some traits from cluster 4 of 3M_SAMP1, but feels like a cluster with no true grouping. Like cluster 3 of 3M_SAMP1, it might contain all the clients that did not completely

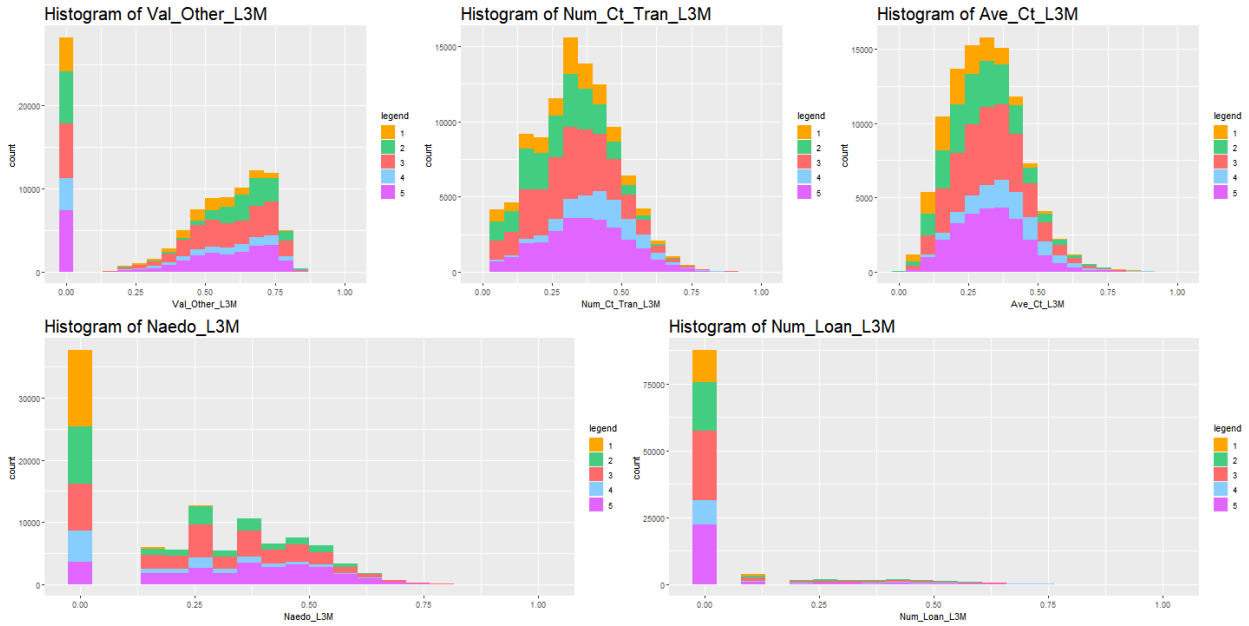


Figure 69: Histograms for the monetary values, transaction counts and monetary averages variables, colour-coded according to clusters for 3M_SAMP2

fit elsewhere.

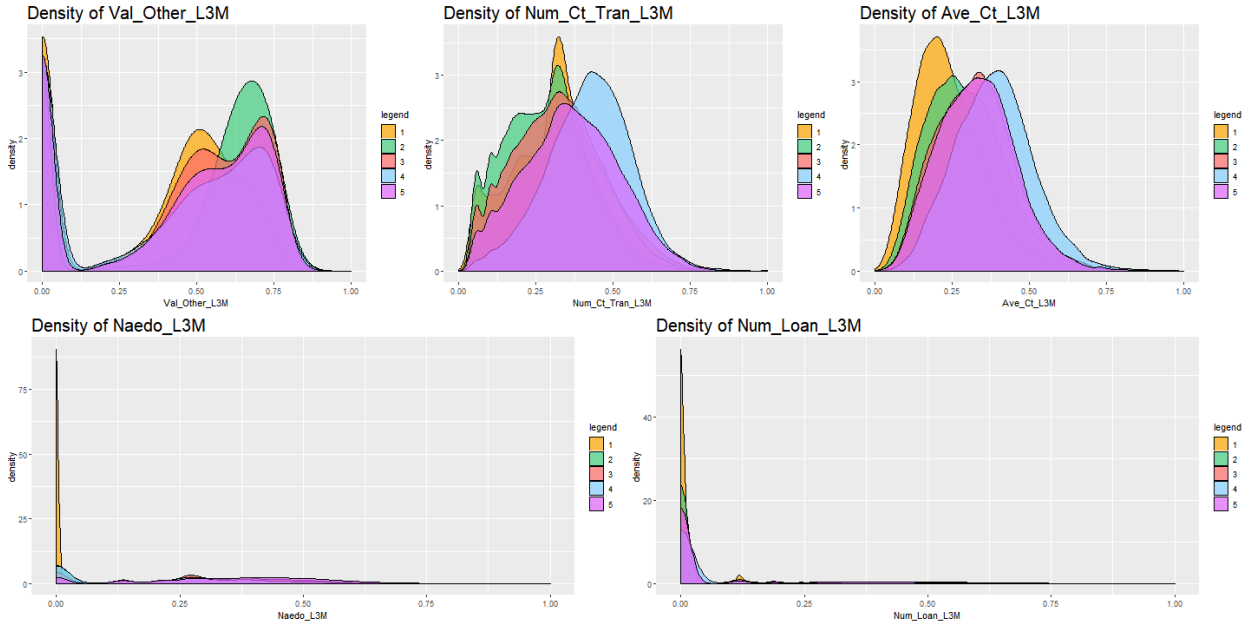


Figure 70: Density plots for the monetary values, transaction counts and monetary averages variables, colour-coded according to clusters for 3M_SAMP2

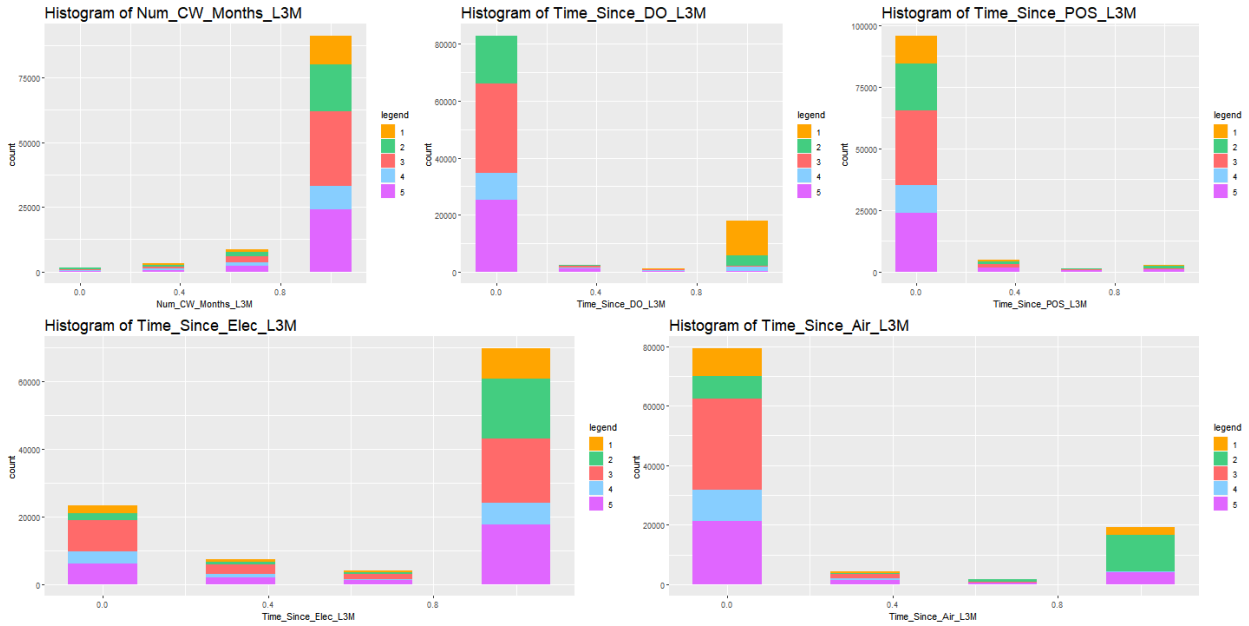


Figure 71: Histograms of the variables containing information on monthly frequency of the different channel's usage, colour-coded according to clusters for 3M_SAMP2

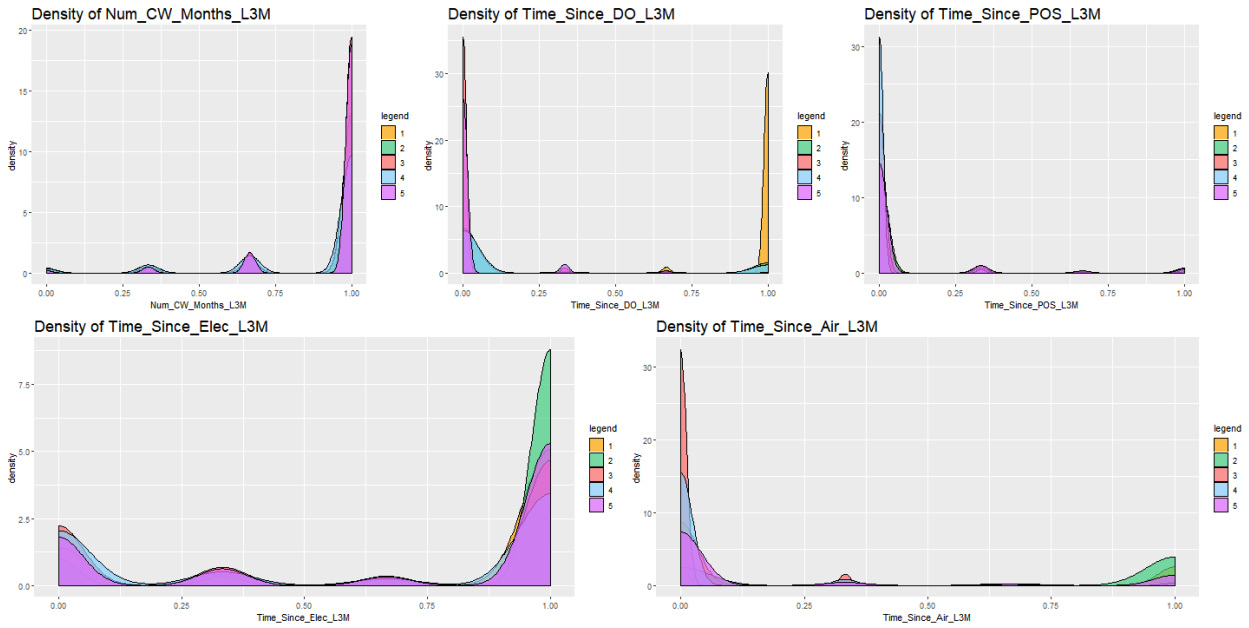


Figure 72: Density plots of the variables containing information on monthly frequency of the different channel's usage, colour-coded according to clusters for 3M_SAMP2

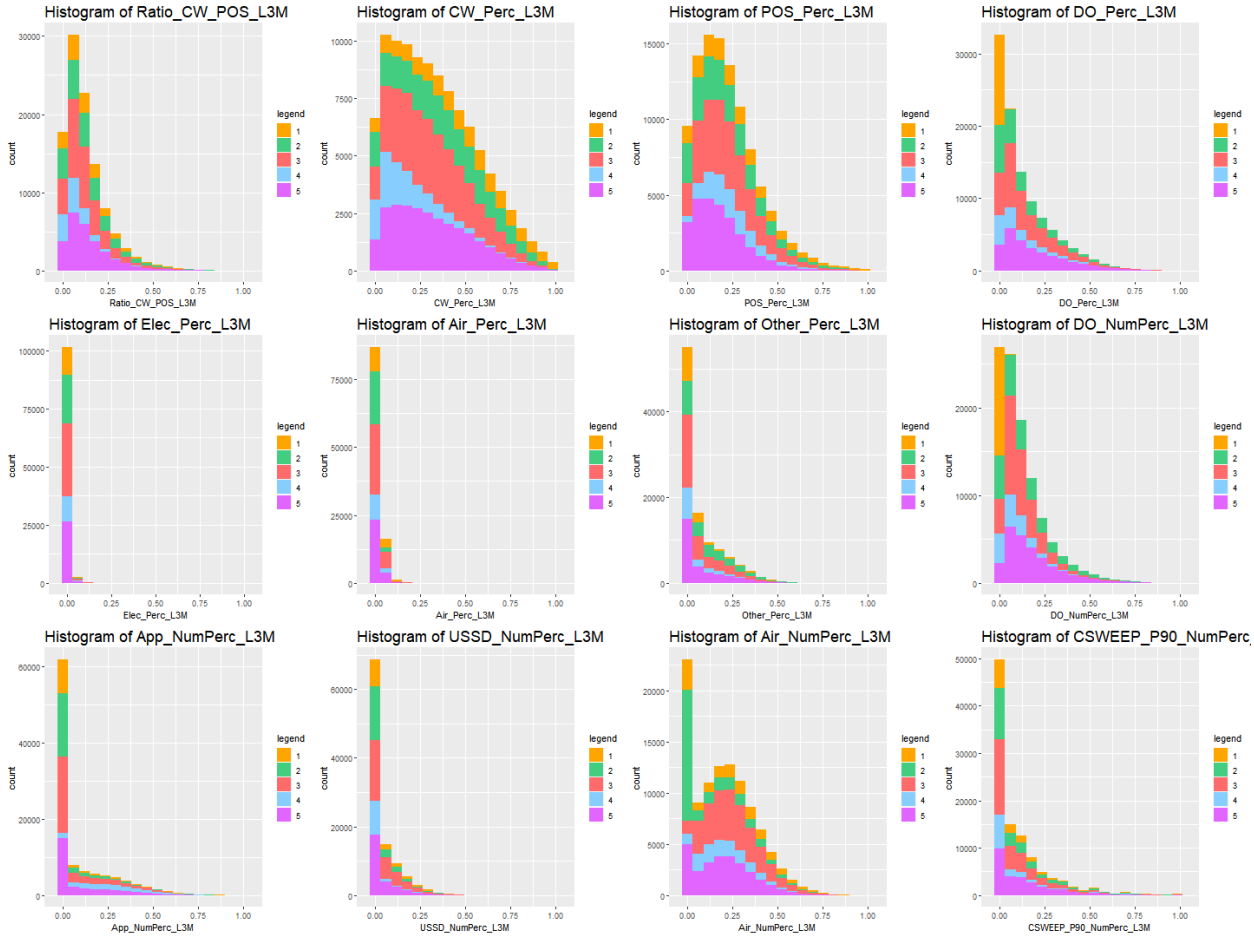


Figure 73: Histograms of percentage and ratio variables, colour-coded according to clusters for 3M_SAMP2

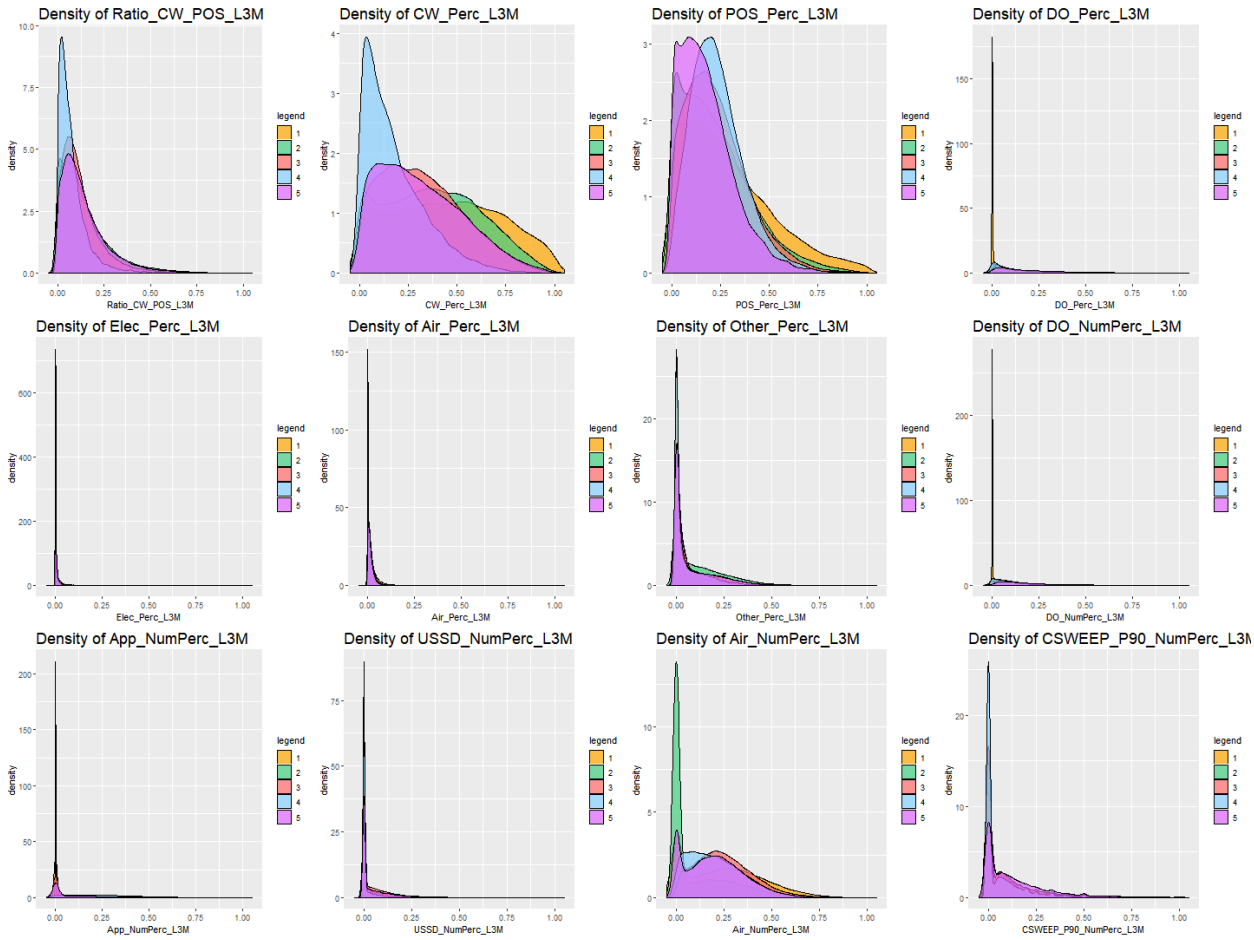


Figure 74: Density plots of percentage and ratio variables, colour-coded according to clusters for 3M_SAMP2

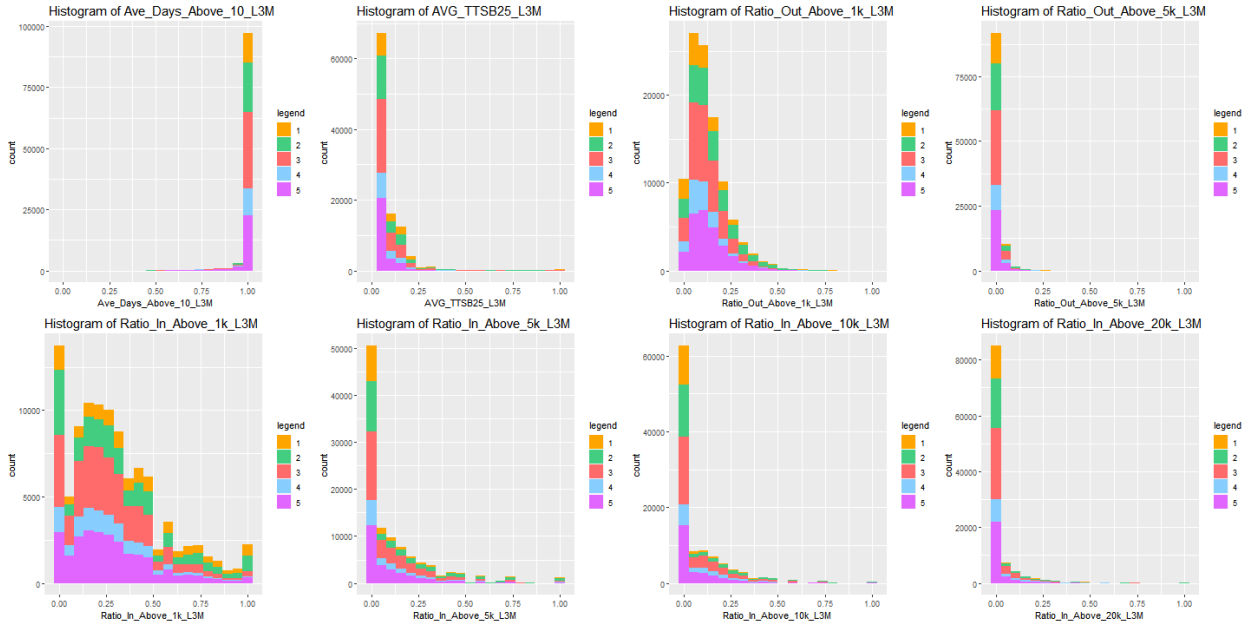


Figure 75: Histograms of the variables containing information on sizes of inflows and outflows, colour-coded according to clusters for 3M_SAMP2

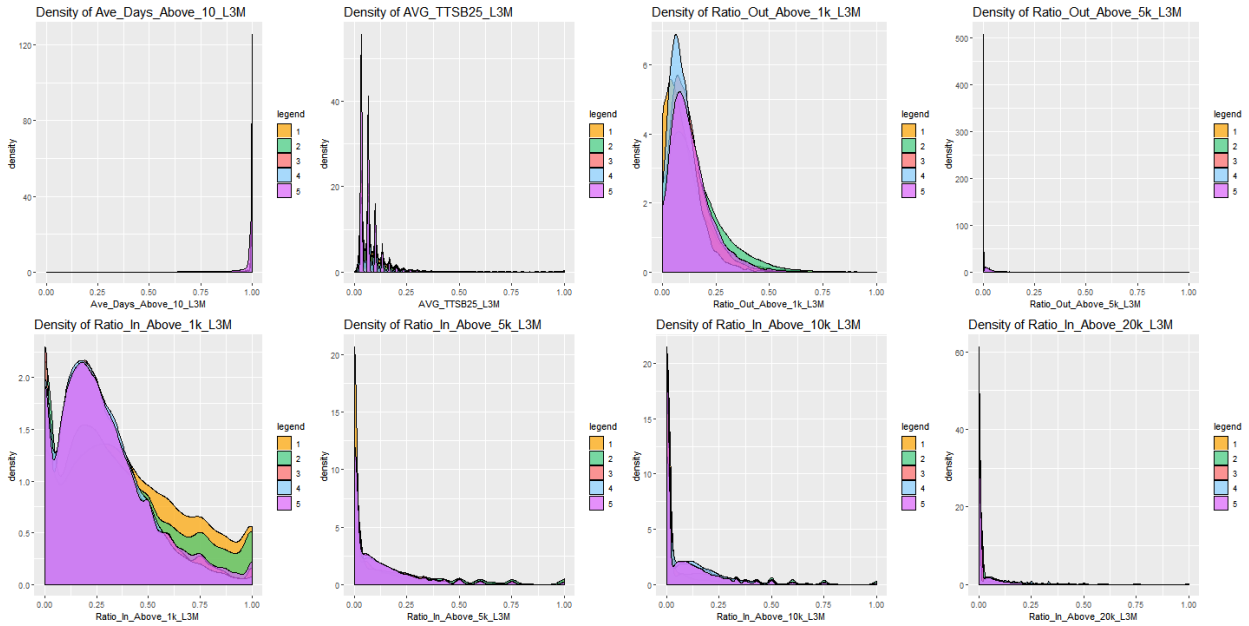


Figure 76: Density plots of the variables containing information on sizes of inflows and outflows, colour-coded according to clusters for 3M_SAMP2

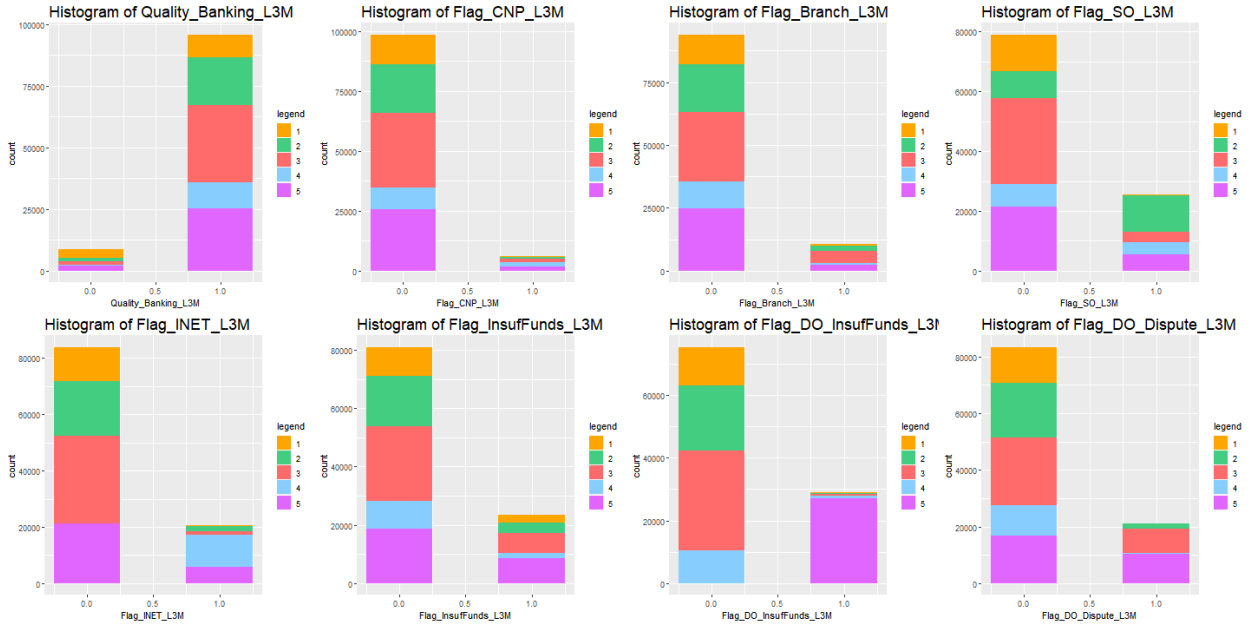


Figure 77: Histograms of indicator variables, colour-coded according to clusters for 3M_SAMP2

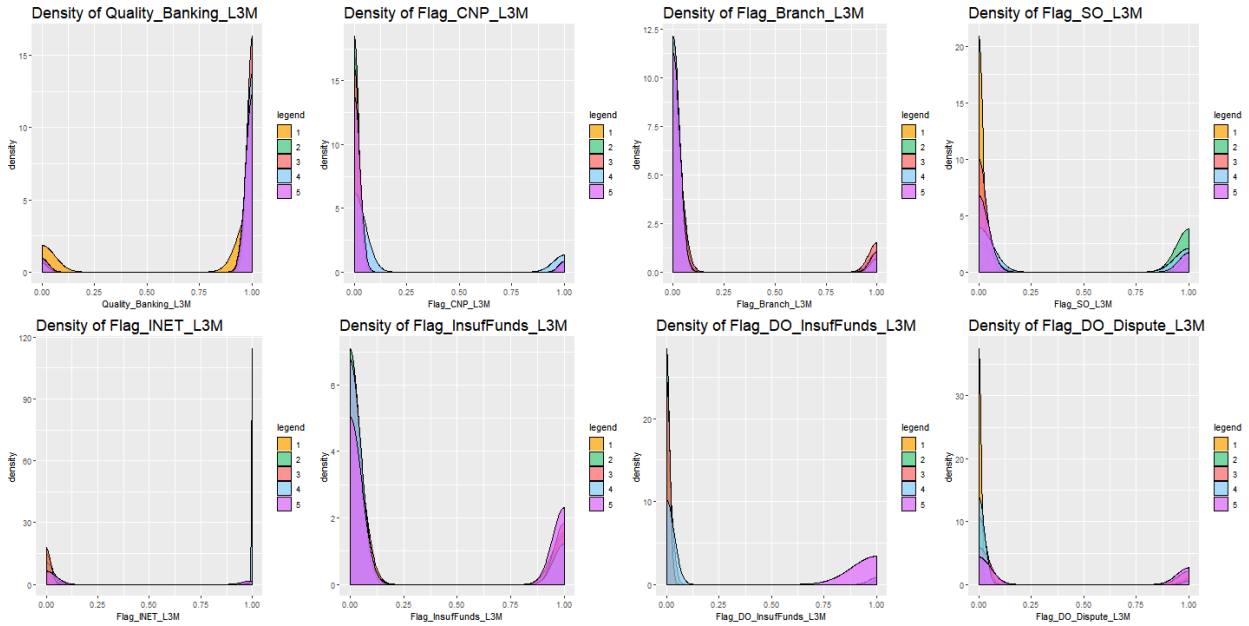


Figure 78: Density plots of indicator variables, colour-coded according to clusters for 3M_SAMP2