Master Thesis

Utrecht University

Using pattern set mining on Weigh-in-Motion data.

Yolan Maxim van der Horst

July 2020

Master Thesis

Submitted in partial fulfilment of the requirements for the degree
Master of Science in the Subject of Artificial Intelligence.

Using pattern set mining on Weigh-in-Motion data.

Yolan Maxim van der Horst

July 2020

First Examiner

prof. dr. Arno P.J.M. Siebes
Algorithmic Data Analysis Division
Department of Information and Computing Sciences
Utrecht University


Utrecht University

Second Examiner

dr. Ad J. Feelders
Algorithmic Data Analysis Division
Department of Information and Computing Sciences
Utrecht University


Utrecht University

Supervisor

dr. Noortje Groot
Innovation and Data Science Division
Human Environment and Transportation Inspectorate
Ministry of Infrastructure and Water Management


Inspectie Leefomgeving en Transport
Ministerie van Infrastructuur en Waterstaat

Yolan Maxim van der Horst
Using pattern set mining on Weigh-in-Motion data.
July 2020
Reviewers: prof. dr. Arno P.J.M. Siebes and dr. Ad J. Feelders
Supervisor: dr. Noortje Groot

Utrecht University
Algorithmic Data Analysis Division
Department of Information and Computing Sciences
Faculty of Science
Princetonplein 5
3584 CC Utrecht

Ministry of Infrastructure and Water Management
Innovation and Data Science Division
Department of Information and Programming
Human Environment and Transportation Inspectorate
Graadt van Roggenweg 500
3531 AH Utrecht

Abstract

This research aimed to answer to which extend pattern set mining can prove to be useful for the Innovation and Datalab (ID-lab) of the Dutch Human Environment and Transportation Inspectorate (ILT). To study this, the unsupervised pattern set mining algorithm SLIM was used. SLIM finds interesting patterns in the data by searching for those patterns that can optimally compress the data. The patterns in question were travel time patterns in Weigh-in-Motion data. This data is relevant for ILT because one of its primary tasks is to inspect commercial transportation vehicles. Weigh-in-Motion systems collect data about vehicles in multiple locations along the Dutch highways. By analysing this data, more can be learned about the particularities of the transportation industry, which is helpful for inspections.

To find these patterns, three approaches were taken. First, using a simplified dataset grouped by license plates, SLIM was compared with well-known clustering technique K-Means. SLIM turned out to be more suitable, as it provided more distinctive results. Second, the dataset was made more complex by addressing the simplifications from the first approach, mainly going from binary to numeric variables. These had to be discretised, so they could be used by SLIM. SLIM still performed well but the resulting patterns were difficult to visualise. Finally, the data was grouped on businesses by adding more variables. The results were modelled as networks to visualise the patterns. This turned out to be a visualisation in which interesting patterns could be found with ease, especially with the filtering tools provided by Gephi.

In conclusion, pattern set mining could prove to be very useful for ID-lab. The new visualisation method using networks made discovering interesting patterns easier and could also be used to find other connections in the data. Interesting travel time patterns were extracted from the data, yet interpretation of the discretised continuous data proved to be challenging.

Acknowledgements

I want to thank my official supervisors Arno Siebes and Noortje Groot, and my unofficial supervisors Paul Merkx, Jasper van Vliet and Gerrit-Jan de Bruin for all the insightful comments, ideas and feedback you have given me. I would also like to thank my second examiner Ad Feelders for his patience and willingness to plan things on short notice.

I acknowledge Margje Schuur and Piet van Hoffen for providing the data and would like to thank them and everyone else at ID-lab for the wonderful time, even during the lockdown.

I would like to thank my friends and family for always supporting me during my studies and finally my gratitude goes out to Sophie van Berkel who assisted me with the final proofreading.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The Human Environment and Transportation Inspectorate (2019) (Dutch: ILT) identifies four goals concerning the reduction of economic and physical damage caused by commercial transportation:

1. Reduce damage to infrastructure, caused by overloaded vehicles.
2. Reduce economical and physical damage caused by and to over-exhausted drivers.
3. Reduce unfair market competition within the transportation industry, caused by illegal operations and schemes avoiding labour laws and regulations amongst others.
4. Gather more information about the commercial transportation industry in general.

These goals cannot feasibly be reached by increasing work force or having more inspections, as the commercial transportation sector is simply too large. As such, algorithmic data analysis and artificial intelligence solutions are needed to provide key insights about risk assessment and possible automation of parts of the enforcement. Providing these kinds of solutions is the goal of the innovation- and datalab (ID-lab) department within ILT.

ID-lab has access to data collected from Weigh-in-Motion (WiM) systems. These systems are active in a number of locations along the Dutch highway system, weighing transport vehicles whilst they are driving[1]. Collected data includes among others, weight, license plates of the cabin and trailer, and date and time of passage.

Currently the systems are used by the ILT to identify vehicles that could be too heavy. Inspectors then haul these vehicles, remeasure their weight and issue a fine if necessary. During the haul, the vehicle is also inspected on following the driving and resting regulations. The data is saved so it can be used for further analysis. One of the main aspiration is to create a model that can predict the risk of breaking the rules.

The task of creating such a model lies with ID-lab. The aim of this thesis, however, was not to create such a model. Instead this thesis

[1]For more information about how they do this, read Labarrere (2019).

aimed to find patterns in the WiM data, using pattern set mining. To be more specific, it aimed to find travel time patterns. The reasons for these choices are manifold.

First, all data is different. As such, effectiveness of methods can vary quite a bit. Next to ID-lab always looking to expand their knowledge about other methods, knowing more about this dataset in particular could help them in deciding which technique would be best suited for a model.

Second, it is unsupervised, which means it can be used on unlabelled data. Unsupervised techniques are often used to learn more about the data, which aligns with the first point. Also they can be used to find a labelling on the data, which a model could use.

Third, it is relatively easy to explain how pattern set mining gets its results. This is in line with the action plan set by Ministry of the Interior and Kingdom Relations (2018). This document outlines the aims of the Dutch government with regards to openness of algorithms, with the aim of being transparent in automated decision processes.

Fourth, travel times are often easily interpretable, as time is something that concerns us all, yet the complexity of their pattern can ramp up quickly, making it a challenging topic.

Finally, my own personal motivation, I was already familiar with pattern set mining and wanted to use it in a practical case.

Two research questions follow from this.

1. **To what extend can pattern set mining prove to be a useful method for ID-lab?**

2. **Can interesting travel time patterns be extracted from the WiM data using pattern set mining?**

Three approaches were taken to study these questions. After some preliminary terminology and background information in chapter 2, they will be discussed in chapters 3, 4 and 5. Each approach will start with explaining the data and methods, followed by the results and a conclusion and discussion on these results. Chapter 6 will draw a final conclusion for all three approaches. Finally, some general limitations and further research ideas will be included in Chapter 7.

Chapter 2

Preliminaries

This chapter explains some terminology and background information.

2.1 Institutions and Technologies

ILT The Human Environment and Transportation Inspectorate, which is a branch of the Dutch government concerned with supervising the policies towards environment and transportation. The rules they are tasked to enforce vary widely and include, but are not limited to:

- Transportation vehicles on land with regards to weight, licenses, dangerous cargo and tachograph correctness.
- Transportation vehicles on water with regards to sulphur emission, safety and fuel legality.
- Transportation businesses with regards to fraud and schemes to avoid labour laws.
- Transportation of waste over borders with regards to enforcing the strict European regulations.
- Transportation of fireworks and other dangerous goods with regards to safety.

ID-lab The innovation- and datalab of the ILT. It concerns itself with providing technological solutions to the ILT using data science and other innovations.

WiM Weigh-in-Motion technology. This is a type of measurement technique, whereby a vehicle's mass can be determined whilst the vehicle is in motion. The specific Weigh-in-Motion technique that concerns the data in this thesis, uses piezoelectric sensors.

KVK numbers and SBI codes Each business in the Netherlands has a uniquely identifying number called a KVK number. The "Kamer van Koophandel (KVK)" is the organisation where you have to register as a business and they assign the numbers. Businesses also have to specify at least one activity they are performing. These activities have to be picked from a list of predetermined activities and are called, "de Standaard BedrijfsIndeling (SBI)", the standard business layout. Each SBI activity has an identifying code.

2.2 Technique: Pattern Set Mining

Pattern set mining, finds its origins in field of data mining. It is unsupervised. In unsupervised learning the data is not labelled. Information is extracted from, often very large, sets of data without the need of a target variable.

2.2.1 Terminology

The following terms are used in the explanation of pattern set mining and are relevant for the rest of the thesis. Table 2.1 is used as an example to make the terms easier to follow.

Item An item is a variable in the data. Items can be included in an observation. In Table 2.1a the items are the variables and the Boolean value in each observation signifies whether they are included or not. Table 2.1b shows them directly in the observations.

Pattern A pattern, also called an **itemset**, is a group of items that describe a subset of the data. This description is often interpretable in regular language. In the example of a pattern is *{Toothbrush, Toothpaste}*. This can be described as "Customers who buy a toothbrush, also buy toothpaste.".

Cover The cover are those observations that are described by the pattern. For example the pattern *{Toothbrush, Toothpaste}*, covers not only the obvious observations 1, 5, 6, 7 and 8, but also observation 2 as the items in observation 2 form a superset of the pattern.

Support The support of a pattern expresses how much of the data supports the pattern. It is the ratio between the number of observations in the cover of the pattern and the total number of observations. In the example, the pattern *{Toothbrush, Toothpaste}* has a support of $6/9 = 0.67$. This means that around 67% of the data supports the pattern.

Id	Toothpaste	Toothbrush	Milk
1	True	True	False
2	True	True	True
3	True	False	False
4	True	False	False
5	True	True	False
6	True	True	False
7	True	True	False
8	True	True	False
9	True	False	False

(a) Original data

Id	Item 1	Item 2	Item 3
1	Toothpaste	Toothbrush	
2	Toothpaste	Toothbrush	Milk
3	Toothpaste		
4	Toothpaste		
5	Toothpaste	Toothbrush	
6	Toothpaste	Toothbrush	
7	Toothpaste	Toothbrush	
8	Toothpaste	Toothbrush	
9	Toothpaste		

(b) Data as patterns.

Table 2.1: Example of a dataset represented as patterns. In this example it concerns transactions from a supermarket.

Code-table A table containing a set of patterns and respective code. These codes can be used to encode the data by translating observations to codes using the patterns that occur in the observation.

Singleton pattern A pattern with only one item.

Standard code-table The code-table that contains all singleton patterns. This is used as a baseline to compare other code-tables.

Compression The pattern set algorithms use compression as a quality measure. By encoding the data with the standard code-table a baseline encoded size is obtained. Other code-tables can be compared by checking how much their encoded size differs. A compression of 20% means that the code-table made the data 20% smaller than the standard code-table did.

SLIM The pattern set mining algorithm that will be used in this thesis. It aims to find the code-table that gives the most compression, using only those patterns that have a support at least that of the minimum support parameter.

Cover ratio SLIM often covers only a part of the observations that can be covered with that pattern. The cover ratio is the ratio between the number of observations that a pattern covers and the number of observations that are in its cover in total. For example, the singleton pattern *{Toothpaste}* might only be used to cover observations 2, 3 and 9, whilst all observations are in its cover. This gives it a cover ratio of $3/9 = 0.33$.

2.2.2 Pattern Mining

Before pattern set mining can be explained, the parent field pattern mining needs to be addressed. The aim of pattern mining is to discover patterns in the data, where a pattern is a description of a subset of the data. These descriptions are often interpretable in normal language. For example, a pattern on a membership database might be "All male members older than 50 that regularly use one of their membership privileges.". This pattern is represented by restrictions on the variables in the data, e.g. $Gender = male$, $Age > 50$, etc.

Pattern Mining was first introduced by Agrawal et al. (1993), in the form of frequent itemset mining on transaction databases. In these databases observations are transactions and variables represent whether or not a product was bought in a given transaction. Itemsets are what they called patterns and they are sets of variables. One of their goals was to find all itemsets that occur in a frequent amount of observations, e.g. groups of products that are often bought together. They succeeded

in this goal with their Apriori algorithm, however it returned far too many frequent itemsets to be properly analysed by a domain expert. The set of all possible itemsets was the power set of the set of variables in their cases, and any of these could be frequent with regards to the observations.

Different methods for dealing with this explosion of patterns have been suggested. Pasquier et al. (1999) tried only mining closed frequent itemsets, Calders and Goethals (2007) did this for maximal frequent itemsets and Pei et al. (2004) and Bonchi et al. (2003) figured out how to add more data-related constraints on the itemsets that were allowed to be returned. However, instead of focusing on returning all patterns within certain criteria, another approach is to return a set of the most interesting patterns. First described by Bringmann and Zimmermann (2007), this is known as Pattern Set Mining.

2.2.3 Pattern Set Mining

The approaches for Pattern Set mining that are used in this thesis are based on the Minimum Description Length principle (MDL). Introduced by Rissanen (1978), it states that given the set of all possible models of the data, the model that compresses the data the most, is the best one. This is based on the intuition that compression makes use of repetitions in the data to represent it more efficiently.

The MDL principle has its roots in the field of Algorithmic Information Theory (AIT). Arguably, the most universal modelling tool is the programming language, as such the best model would be the shortest program to output the dataset. Invented by Kolmogorov (1968) one of the founders of AIT, the length of such a program is known as the Kolmogorov Complexity of the data. Kolmogorov proved that this complexity is uncomputable, and that for any practical purposes, only an upper bound can be given. He also proved that the specific choices for operating system and programming language create a constant overhead and as such are of no interest when approximating the complexity.

Krimp The MDL principle was first applied within Pattern Set mining by Vreeken, Leeuwen, et al. (2011). They created a code-table, a set of patterns and their respective codes. The patterns in this code-table were used to cover observations in the data until the whole data was covered. By encoding the patterns from the code-table on cover usage, e.g. patterns that are more often used to cover observations get a shorter code, the data could be optimally compressed. Their goal was to find the code-table that creates the smallest total size of the compressed data and the code-table combined.

This is akin to finding Kolmogorov Complexity, only instead of all possible models, only a subset is used in the form of code-tables. This relation is helpful as it provides a very useful notion. The actual algorithm used for encoding the data is irrelevant to the result, as this is

part of the constant overhead. This allowed them to used Shannon entropy, invented by Shannon (1948). It allows for optimal prefix coding, which makes sure that the data uses the shortest codes possible, whilst still creating a unique encoding.

Even though using code-tables drastically limits the set of possible models, it is still an NP-hard problem. The set of all possible code-tables is the power set of all possible patterns, which itself is the power set of all variables. Now not all of those code-tables are allowed, as they are required to cover the entire data. Still even for 10 variables with binary values there are more possible code-tables than can be checked before the end of the universe[1].

Instead of checking all code-tables, some smart heuristics were used. First, only patterns that occur frequently in the data are useful for compression. So by only considering patterns with a given minimum support, the search space is drastically reduced. Secondly, by starting with a code-table that only contains patterns of the smallest complexity, e.g. all patterns with a set cardinality of 1, they could iteratively increase the complexity of considered patterns. Finally, considering patterns in a standard order[2] and covering the data in another standard order[3], allows for optimisation and makes sure the outcome is deterministic. At each step of the algorithm, the patterns are considered in this order and the first pattern that improves complexity is added to the code-table. Also patterns in the code-table that no longer contribute enough are pruned. The end result is the KRIMP[4] algorithm.

KRIMP significantly reduced the number of patterns returned for any given support threshold. It also found patterns that matter, as KRIMP-based classification resulted on par with state-of-the-art classifier of that time. However, as always with a first application, there are some downsides. KRIMP needs to be provided a set of frequent patterns, so naive pattern mining is necessary before applying KRIMP. This takes a lot of time and memory, as all the patterns need to be saved somewhere. Furthermore, it considers patterns in a standard order, which means it might miss combinations of patterns that would otherwise have been very beneficial. Finally, KRIMP does a lot of unnecessary work, it needs to reject most of the proposed candidate patterns.

[1] The are $2^{2^{10}} \approx 1.8e308$ possible code-tables. Even if only a fraction of those are allowed it would be unfeasible. If a supercomputer could check 1 million code-tables every microsecond, it would barely have checked $1e34$ code-tables in the, at most, 100 trillion years that scientists predict our universe has left. See Adams and Laughlin (1997) for more information about the lifetime of the universe.

[2] Standard Candidate Order: $\downarrow$ support, $\downarrow$ complexity, $\uparrow$ lexicography

[3] Standard Cover Order: $\downarrow$ complexity, $\downarrow$ support, $\uparrow$ lexicography

[4] **krimp** means shrink in Dutch, a subtle nod to the compression element.

Slim Smets and Vreeken (2012) set out to improve upon KRIMP by addressing these downsides. They realised that when one starts with the code-table of the simplest complexity, more complex patterns can be created from pairwise combinations. Doing that has the following advantages:

- It eliminates the need to mine frequent patterns before applying the algorithm.
- Patterns are considered more than once, as all pairwise combinations are considered every iteration.
- The number of patterns considered per iteration is reasonable, as code-tables are often exponentially smaller than the set of frequent patterns.
- The patterns that are considered in this approach are likely to improve compression, as they are aimed at replacing the pair from which they are created, making the consideration less wasteful.

However if the compression has to be calculated for every considered pattern, even this more optimised solution still takes weeks to run. So the final improvement is a heuristic that estimates the compression gain of a candidate pattern. The candidates are considered in order of their estimated gain. The first considered candidate which improves the actual compression is added to the code-table. Like with KRIMP, pruning is also applied to the code-table. The resulting algorithm is called SLIM[5].

SLIM is orders of magnitude faster than KRIMP both in time and memory. It only requires one input parameter, the minimum support. For the purposes of this research, it is the better choice. However KRIMP can be applied to different forms of data, for which it is currently unknown if SLIM can be used. Examples of this include streams of data as shown by Van Leeuwen and Siebes (2008) and as an imputation technique as shown by Vreeken and Siebes (2008). These fall outside the scope of this thesis, but once more highlight that the effectivity of a technique depends on the data.

For the purposes of this research SLIM is applicable. KRIMP is mainly included here to give some background on the origin of the technique and because explaining SLIM without first explaining KRIMP is less understandable.

[5] **slim** means smart in Dutch, as it aims to be smarter than KRIMP.

2.3 Other techniques

There are some other methods that are used in this research and require a bit of background information.

First, another method of unsupervised learning, K-Means clustering. K-Means clustering, popularised by MacQueen (1967), aims to divide the data into k clusters. It does this by creating k points in the data space and assigning the observations to their closest points. With each new assignment the mean of the assigned observations is taken to be the new point, as such the k points move around and can be considered the means of the clusters. Eventually the means become stable and the result is essentially a Voronoi-diagram created by these means.

Second, some network science methods were used to visualise part of the result. A network is a collection of nodes and edges. A node is an individual actor and an edge connects two nodes together. Edges can be directed, like an arrow from one node to the other, or undirected, like a line between the nodes. Edges can also have a weight, for example the number of connections between two nodes.

2.4 Software

The main software that was used is RStudio, version 1.2.5019 by RStudio Team (2019), with the following packages.

- R, version 3.6.0 by R Core Team (2013)
- arules, version 1.6-5 by Hahsler et al. (2005)
- data.table, version 1.12.8 by Dowle and Srinivasan (2019)
- tidyverse, version 1.2.1 by Wickham et al. (2019)
- pbmcapply, version 1.5.0 by Kuang et al. (2019)
- fastDummies, version 1.6.1 by Kaplan (2020)

Most of the work was done in RStudio using the above packages, but some parts used other software. An executable for the SLIM algorithm was created by Gandhi et al. (2015). For this research an interface was written to use it in R. The R code of this interface can be found in Appendix A. Finally, Gephi, version 0.9.2 by Bastian et al. (2009), was used to create and visualise the networks.

Chapter 3

First Approach: Proof of concept

The first approach was mainly a proof of concept. Travel time patterns where represented in a very simplified way. The goal was finding simple, yet interesting patterns and comparing pattern set mining with a relatively well known method.

3.1 Data

For the initial analysis, all recorded passes from January 2018 up until November 2019 were taken from the WiM data. This dataset consists of over 22 million observations.

From each observation 2 variables were used. UTCPASSAGEDATUM, which contains the date and time the vehicle passed the site and KENTEKENVOORENC, which contains an encoded version of the license plate on the front of the vehicle. The decoded version was also available, but not used out of data sensitivity concerns. It is relevant to mention that it was the front license plate as trucks can have different license plates for the front (cabin) and back (trailer).

	KENTEKENVOORENC	Monday_0-3	Monday_3-6	Monday_6-9	Monday_9-12	Monday_12-15	Monday_15-18	Monday_18-21	Monday_21-0	Tuesday_0-3	Tue 6
1	MBIKHFLMGGOHNNLGHLNIBLJ...	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FA
2	HHEHMADAJODAOGIPBFFFLCC...	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FA
3	NJPOGGIPDANHBMDCBAMOAL...	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FA
4	KFGCAKOBCKDCAHKEHDOKEOJ...	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FA
5	JGGNKCHBOHKCMFMLBEINJBO...	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FA
6	IAJJPLJLOMHCFPNBFIIPLNLAFD...	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FA
7	MKBAJFNGLPKGOJCMFOMHLJ...	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FA
8	IMOAFJDFNMPECLGPJKGODCH...	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	TR
9	GJJODMDPMGNBNKGFEGNIJILI...	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FA
10	NJAONBMPECLPNNIBMLELNCD...	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FA
11	IJOIEHCBEIKODLOLIKBCOBLBB	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FA

Figure 3.1: The first few records of the complete dataset for the first approach. It contained $1,594,586$ observations and 56 variables, excluding the KENTEKENVOORENC identifying variable.

3.1.1 Creating travel times

To create travel times, the date was converted into its respective day of the week and the time was grouped into 3 hour windows. So **06-03-2018 16:41:53.5** would become **Tuesday between 15:00 and 18:00**. The 3 hour windows where chosen because of divisibility, exactly 8 per day, and to manage complexity.

A new dataset was created with license plates as observation identifiers. For each day of the week and time window combination, hereafter referred to as **time-slot**, it was recorded if that license plate had been present in the original data. Figure 3.1 shows what this dataset looks like. Each variable represents a time-slot, with binary values stating whether or not the license plate occurred in that time-slot. The choice for binary values is to facilitate the pattern representation, like seen in the example in chapter 2. It is evident that smaller windows than 3 hours would lead to more variables and thus more complexity. In total this dataset has nearly 1.6 million observations.

3.1.2 Sampling

The number of observations in combination with the number of variables, makes the dataset too big to use properly. Sampling was used to address this. Riondato and Upfal (2014) proposed sampling techniques specifically for pattern mining. They described a set of functions to determine the sample size given a specific class of problems within pattern mining. One of them is for Top-K itemset mining. This is comparable to pattern set mining. The best pattern set with k patterns is similar to top k itemsets given the constraints that they should describe the data well. So the following function for calculating the sample size is used.

$$\frac{8}{\epsilon^2}(d + log\frac{1}{\delta}) \tag{3.1}$$

With the following variables:

- ϵ The accuracy error, for any itemsets it holds that their frequency in the sample differs by at most ϵ compared to their actual frequency.
- δ The failure probability, the chance that itemset that are frequent in the original data are not frequent in the sample.
- d The VC-dimension of the dataset or an approximation thereof called the d-index. This approximation can quickly be calculated by finding the maximum number d for which there exists at least d itemsets of length d.

Sampling was performed with parameters $d = 51$, $\epsilon = 0.05$, and $\delta = 0.00001$. The big difference between δ and ϵ is because it is very important that no frequent itemsets are missed. Any itemsets that are not frequent in the sample will be skipped entirely by SLIM, even if

they are frequent in the original data. It is less important that their frequency in the sample matches completely with the original, as they will still be considered by SLIM regardless.

The resulting sample size reduced the number of observations from approximately 1.6 million to approximately 200 thousand.

3.2 Experiments

3.2.1 Pattern set mining

SLIM was applied to the sample set three times. First with a minimum support of 0.01, then 0.005 and finally 0.001. Bigger values than 0.01 were tried but very few patterns had a support of more than 1% of the data. Smaller values than 0.001 were also tried, but resulted in code-tables with too many patterns to analyse. So these values were chosen, with 0.005 as the middle ground between the two.

3.2.2 Clustering

As the problem was made relatively simple in this first approach, it is useful to evaluate the data with another method. By doing so, the results can be compared and something can be said about the validity of continuing with pattern set mining when making the problem more complex. For this purpose, the sample set was clustered using K-Means.

K-Means requires one input parameter, namely k the number of clusters. A common way to find a good number for k is the so called elbow method. The entails running K-Means with a range of values for k and plotting their within-clusters sum of squares. Figure 3.2 shows such a plot for k between 1 and 20. The plot shows a kind of arm where the inflection point on the curve, the elbow, is be the best value for k.

In this case the elbow is at $k = 2$, however this is a very low value of k. Low values of k might not be enough to differentiate multiple groups within the data. So another value for k was also used. There is no other elbow in the plot so the largest value was chosen, $k = 20$.

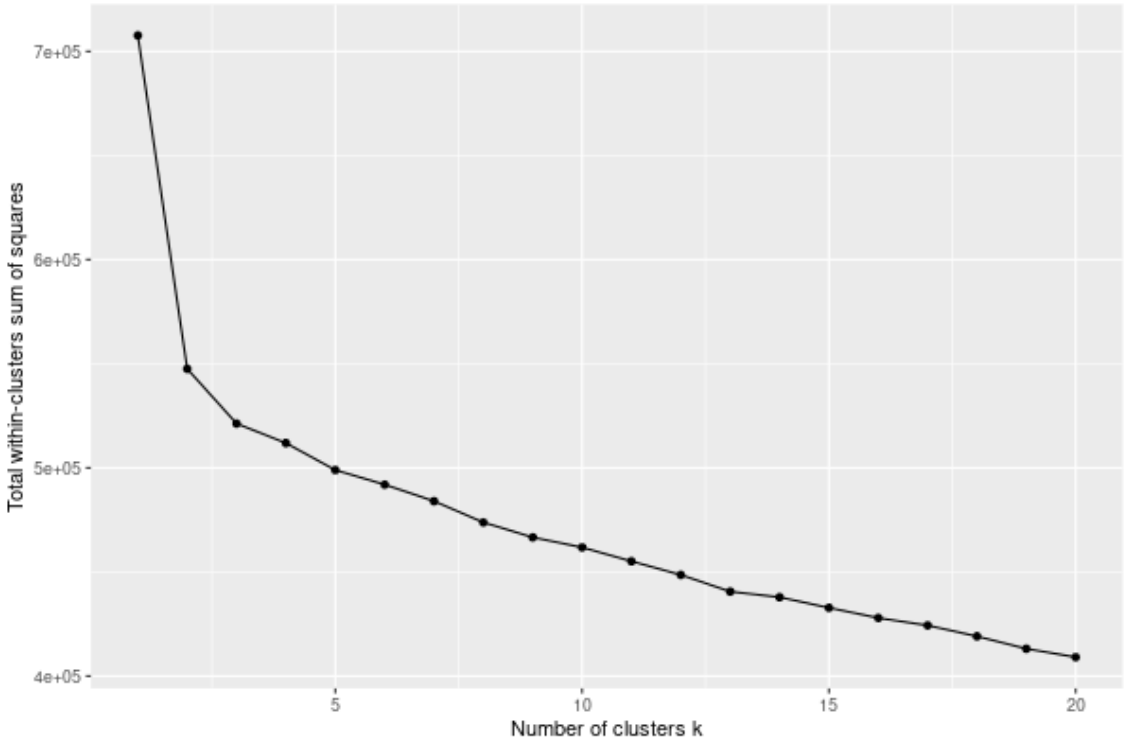


Figure 3.2: Elbow plot for K-Means, generated using the sampled dataset.

3.3 Results

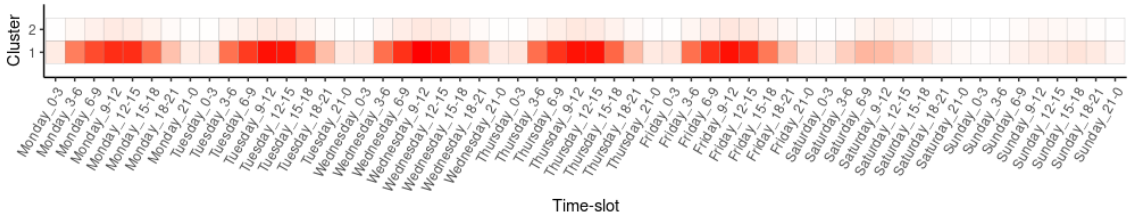
It was a challenge to find a good way to visualise the found results. For this approach it was decided to use a visualisation method that could be applied to the results of both SLIM and K-Means. Both patterns and clusters can be visualised as a heatmap over the variables.

3.3.1 K-Means

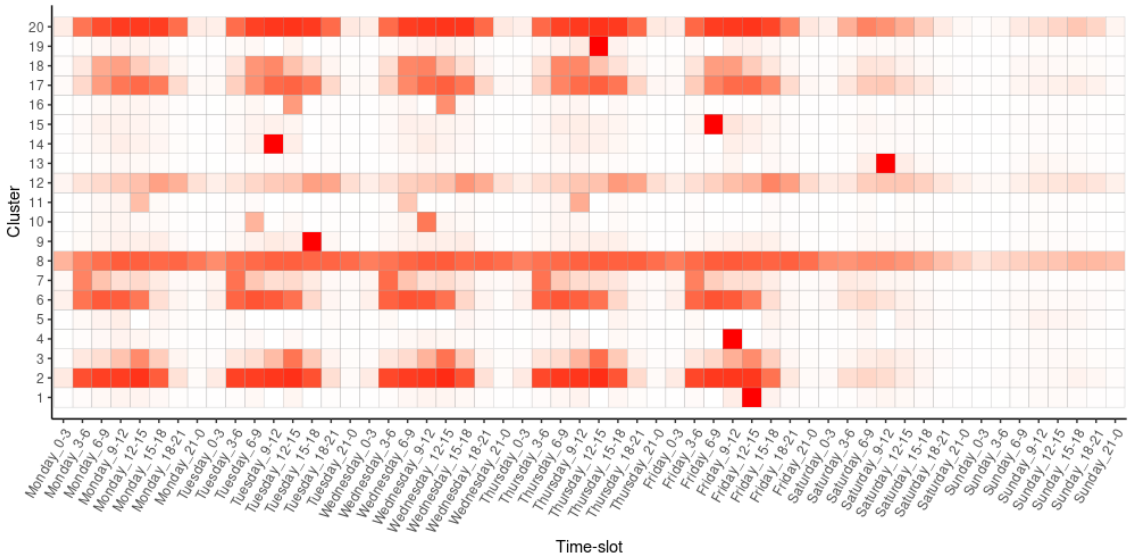
Starting with the K-Means results visualised in Figure 3.3. For each cluster centre on the y-axis, its value for each variable on the x-axis was plotted. The more red a square, the more that variable is represented in that cluster centre.

For $k = 2$ in Figure 3.3a, the first cluster represents nearly all the data. It is clear that most of the driving is done during daytime on the weekdays, with a little bit less driving on the weekends and even less during the nights. However this is all in the same cluster, so this is all the information that can be gained from this.

For $k = 20$ in Figure 3.3b, most of the clusters represent daytime weekday driving. especially clusters 2, 6, 8, and 20. Some clusters are very sparse with only a single important variable. A possible explanation is that the data could contain many observations with only one variable present.



(a) $k = 2$



(b) $k = 20$

Figure 3.3: Cluster centre values for each of the time-slots. White is low, red is high. Result of applying K-Means to the sampled dataset.

3.3.2 SLIM

Running SLIM gave the following results:

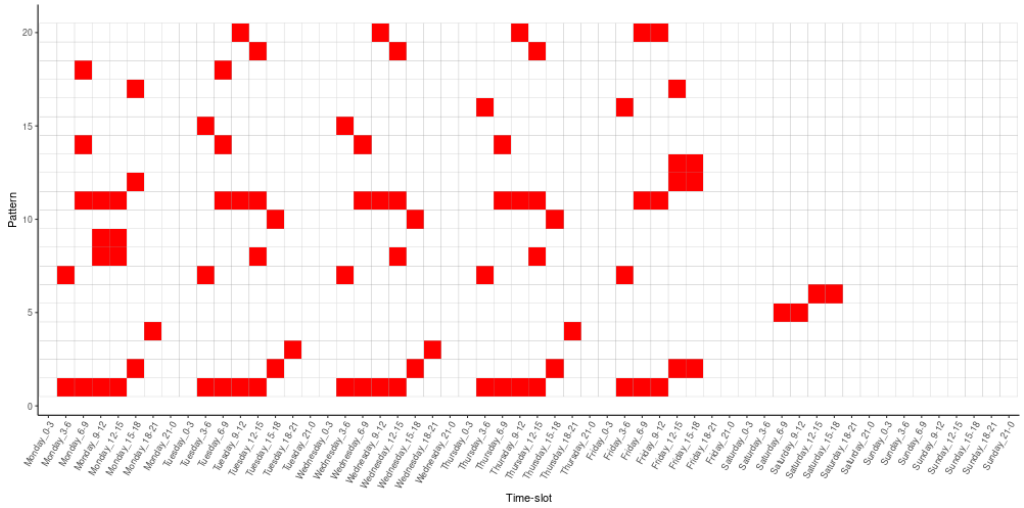
- SLIM with minimum support of 0.01 yielded 37 longer patterns and the 56 singleton patterns with a compression of about 20%.
- SLIM with minimum support of 0.005 yielded 72 longer patterns and the 56 singleton patterns with a compression of about 22%.
- SLIM with minimum support of 0.001 yielded 235 longer patterns and the 56 singleton patterns with a compression of about 27%.

Figure 3.4 shows the top 20 patterns when ordered on descending cover ratio. Higher cover ratio means that more of what can be covered by that patterns is actually being covered by it. As such it is a measure of the importance of a pattern.

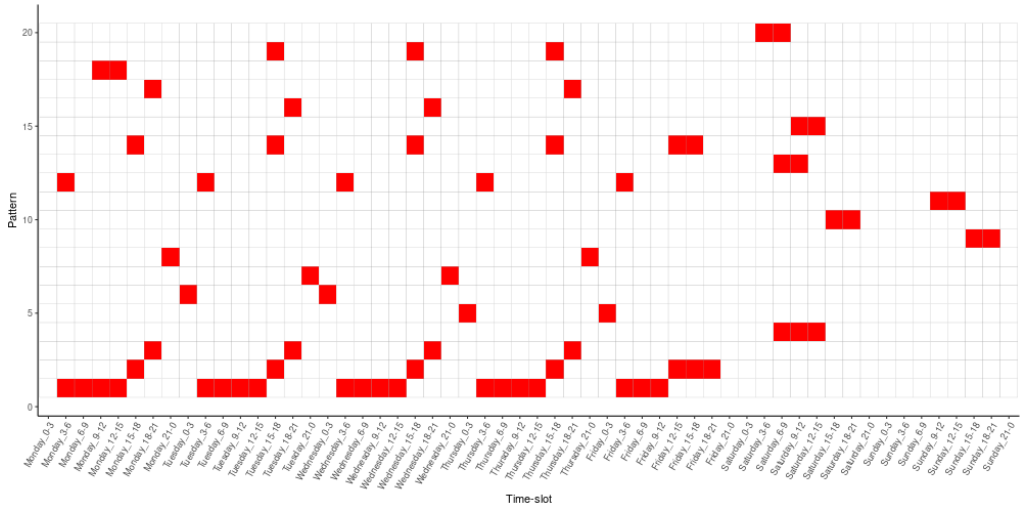
Setting the minimum support lower does not give better results by default. Figure 3.4c shows this. The low minimum support of 0.001 made it so the long daytime weekday pattern that is the first pattern in the previous figures was no longer at the top. It was replaced by even longer patterns that don't represent much of the data, but do have a very high cover ratio.

The SLIM results also show something about the amount of singleton observations. Not in the figures, but in the compression they achieve. It is quite low compression even at a low minimum support. This is likely, because singleton observations cannot be compressed. We also see this in Appendix B, where the complete heatmaps are given, this time ordered on highest cover occurrence count. The singleton patterns have the highest cover occurrence by far.

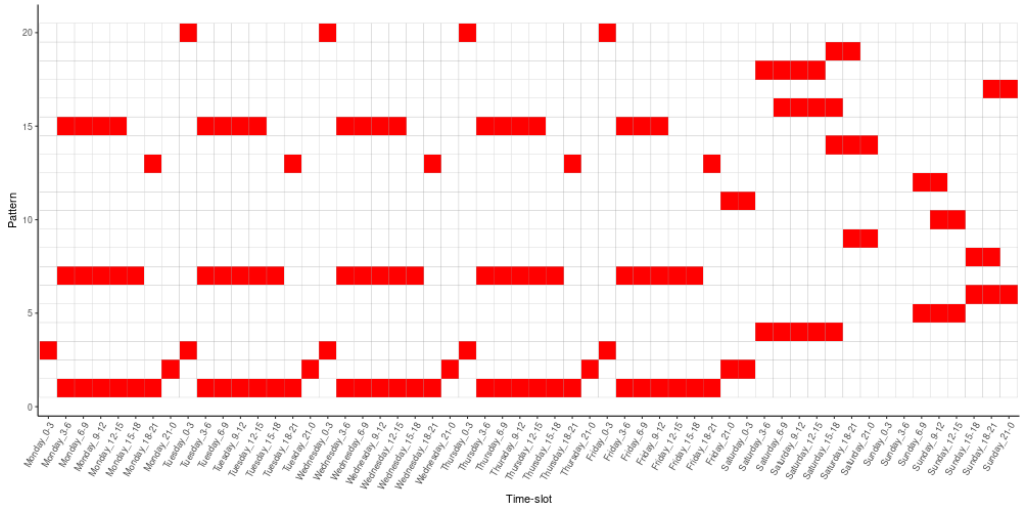
Many of the patterns concerns daytime weekdays. In Figure 3.4a, the night and weekend time-slots are almost entirely absent. When the minimum support was set lower in Figure 3.4b, the night and weekend patterns start to show up. This follows the intuition that trucks drive mainly during work hours and far less at night or on weekends. With the relatively high minimum support of 0.01 they couldn't be considered because they didn't represent enough of the data.



(a) minimum support = 0.01



(b) minimum support = 0.005



(c) minimum support = 0.001

Figure 3.4: Top 20 non-singleton patterns based on cover ratio. A red square means that that time-slot is included in that pattern. Results of running SLIM in the first approach.

3.4 Conclusion

The results that were found support some intuitions about the transportation industry. Most vehicles drive during the daytime on workdays and there seemed to be separate groups that drive during the night or on weekends.

SLIM and K-Means have some similarities. Both highlight the daytime workday drivers and both seemed to indicate that the data may contain many singleton observations.

SLIM differed with K-Means in terms of representing the night and weekend time-slots. In K-Means they were very slightly represented within some of the more filled out clusters, instead of in their own clusters. Where as in SLIM they got their own patterns.

SLIM and K-Means could be compared quite well. The discrete values from SLIM painted a clearer picture compared to the continuous value of the K-Means cluster centres. SLIM was also more informative with its separate clusters for relatively anomalous data, the nights and weekend. So pattern set mining is the better option to handle this problem.

There was a benefit in going from a minimum support of 0.01 to 0.005, adding more interesting patterns to the results. However, with 0.001 there was such a significant increase in the number of patterns, that analysis became a lot more complex without a clear benefit to the results. As such both too high and too low minimum support are unwanted. A minimum support of 0.005 is the right option for this dataset.

In conclusion, interesting patterns could be found in a simplified dataset and SLIM was more suited for this purpose than K-Means.

3.5 Discussion

A few issues remained in this approach, some of which will be addressed in the next approaches. First of all, the results suggested that the data contained many singleton observations. As these do not contain any patterns deemed interesting, they need to be removed. This will be addressed in the second approach.

Second, the data was simplified a lot with some arbitrary choices. The 3 hour windows and the binary representation both throw away much information. As such they will also be addressed in the second approach.

Third, there is the possibility that K-Means could have given a more informative result if more clusters where used, as the total within-clusters sum of squares was still descending. However adding more clusters to K-Means gets computationally quite expensive, with no guarantee that it will improve the results. As K-Means is not the focus of this research, this issue is not addressed and remains as a topic for further research.

Finally, the data is grouped by license plates. It might be better to add more data and group in some other way. Possibly by type of cargo or by owner of the vehicle, so the differences between those groups could be found and described in a more interpretable way. "Some license plates drive mainly on Friday evening." Is a lot less interpretable then "Vehicles of type x drive mainly on Friday evening.". A method of doing this will be addressed in the third approach.

Chapter 4

Second Approach: Adding more complexity

The second approach shifted the focus fully to pattern set mining and aimed to address some issues that were posted in the first approach. These issues were the removal of singleton observations and the abritrary choice of time-slots. The goal was to find interesting patterns in a dataset that is more complex.

4.1 Data

The source data is the same as in the first approach. So the WiM data from January 2018 up until November 2019, with only the UTCPASSAGEDATUM and KENTEKENVOORENC variables for each observation. The high number of singleton observations was the first thing that was addressed. Every license plate that occurred only once was removed. This reduced the number of observations by around 9 million, from approximately 22 million to 13 million.

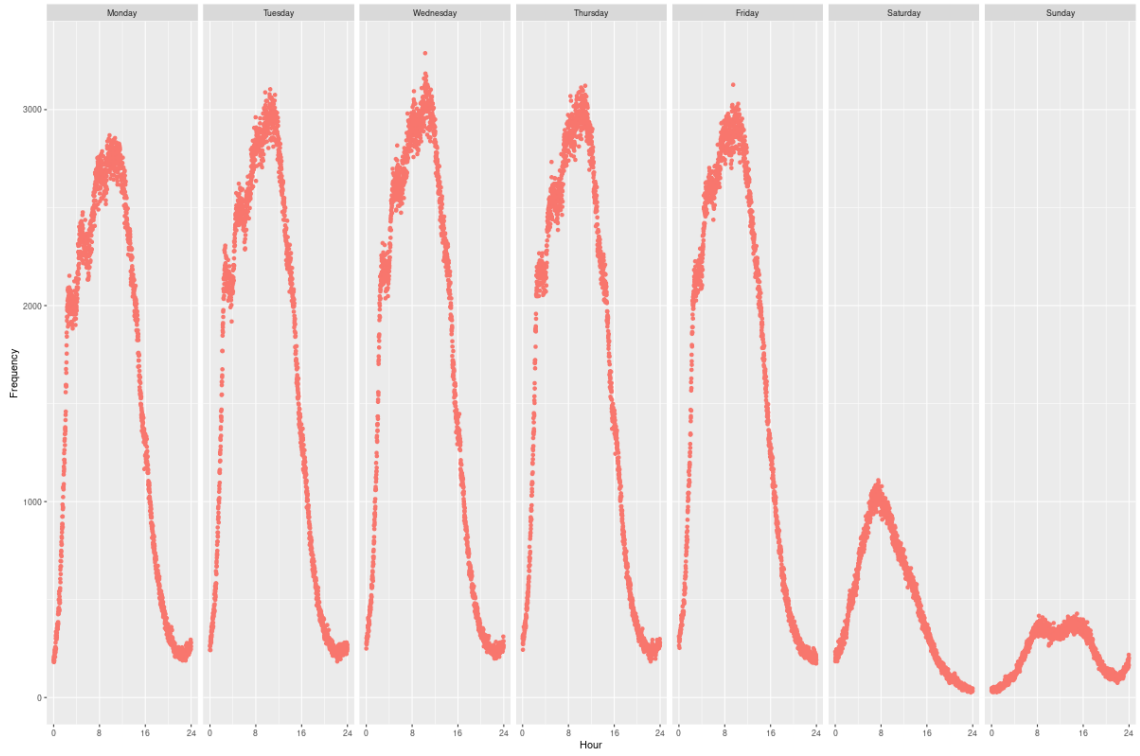


Figure 4.1: Number of observations for each minute of the week.

4.1.1 Improved time-slots

Instead of arbitrarily dividing the week into days and and the time into 3 hour windows like in the first approach, it seemed better to create time-slots based on observation density. Figure 4.1 shows the number of observations for each minute of the week. It can clearly be seen that most vehicles drive during the daytime and on weekdays. There are also small peak and valleys during the day. The ideal time-slots should be short during busy periods to divide these peaks. They should also be longer during quiet periods so those time-slots contain enough observations to not be buried.

K-Means clustering was used to approximate these ideal time-slots. By clustering the cumulative sum of the observation density distribution with $k = 60$. Computationally this is not a problem, because it only concerns a single variable. The k value was not chosen arbitrary. It was close to the 56 variables from the first approach and it kept the complexity manageable when addressing the upcoming issue.

The resulted clustering can be found in Figure 4.2. Most of the peaks were separated and the weekend needed less clusters than the weekdays. This was no surprise as it contained less observations and not as many small peaks. Other values for k, all of them between 40 and 100, were also tried, however $k = 60$ gave the best result. These clusters were used to divided the data into 60 time-slots, from which 2 dataset were created. The exact windows that these time-slots encompass can be found in Appendix C.

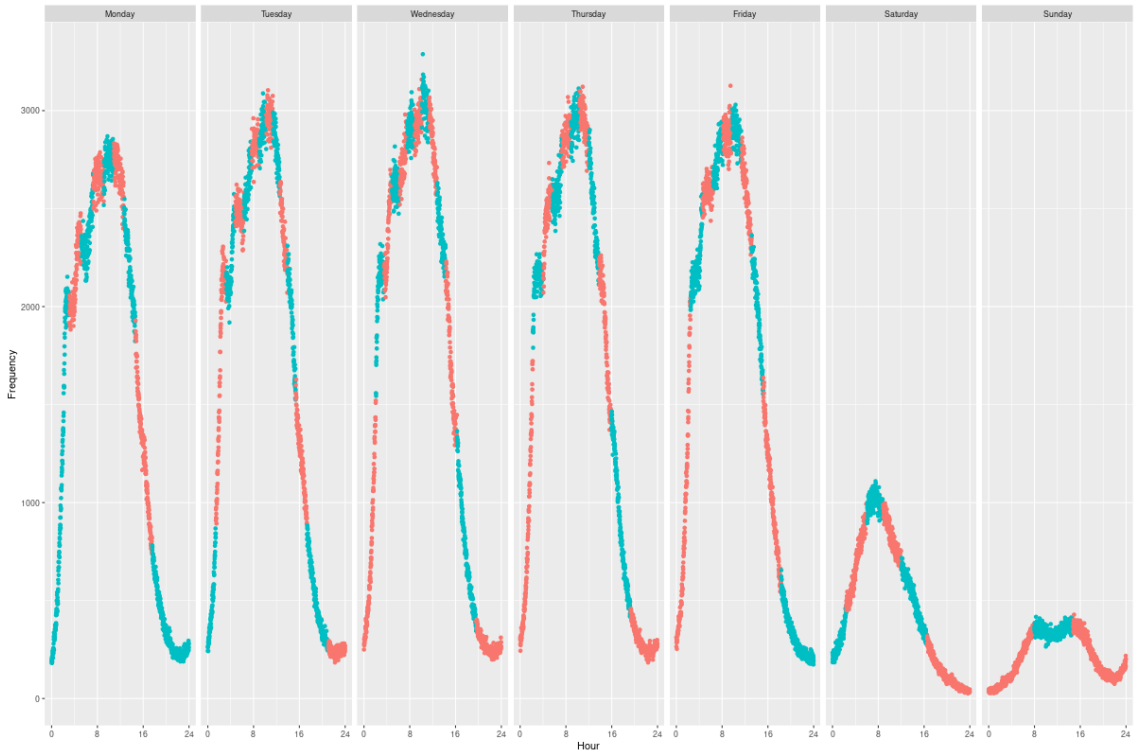


Figure 4.2: Time-slot clustering on the number of observations for each minute of the week. Generated using the WiM dataset without singleton observations. The two colours are used to signify where a new time-slot starts.

	KENTEKENVOORENC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	AAKOJKMJJOJLAFGKGKJFICOPBMLGKOFNHKAPCGGP	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	TRUE	FALS
2	ABNLOBCHHCBFGHNKBOAMCAJBHNFIOBAIHKHIDJOP	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALS
3	ADFIDPAIFPLFDEGFJPDHLFJCJBPIIFIELNACKHGK	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRU
4	ADJMFENDLNJOCLGOILDCLMGJDDMFIMDKDOFBNDGN	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	FALSE	FALS
5	AGLOMEDPIKJLPAADEAGAAAKLICKDEBJFKGNICBJD	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALS
6	AJFPKPNHAFMBNAENDMAKIILIHCBANJAAKIIDMJGI	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALS
7	AJJGIOMOIPHNONBNOHKECBMDHGLIJFMMMNFEHAFI	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE	TRUE	FALS
8	AKHCPFIOHNMOIKIEKGEPFDBHJFFLAPKGPFCIKOHI	TRUE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	FALSE	FALSE	FALSE	FALS
9	ALBJNKGGODPLFDMIOOEMJCDDEFJGAJNIFMKCENFB	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALS
10	AMDIBDNLLJAENGDOMCNDHDMFOUFHBGLBGDIEBCC	TRUE	TRUE	FALSE	TRUE	FALSE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	FALS
11	ANAAAOJDMJOFMOEHHJAPCKNMELAIFJOPMGBKBOHE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRU

Figure 4.3: The first few records of the **binary dataset** for the second approach. It contained $792,197$ observations and 60 variables, excluding the KENTEKENVOORENC identifying variable.

	KENTEKENVOORENC	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	AAKOJKMJJOJLAFGKGKJFICOPBMLGKOFNHKAPCGGP	6	2	4	1	0	0	0	0	8	0	1	3	0	0	2	0
2	ABNLOBCHHCBFGHNKBOAMCAJBHNFIOBAIHKHIDJOP	2	1	3	1	3	1	0	0	1	1	1	1	0	0	0	0
3	ADFIDPAIFPLFDEGFJPDHLFJCJBPIIFIELNACKHGK	14	6	4	1	1	0	3	1	7	4	2	2	3	0	1	1
4	ADJMFENDLNJOCLGOILDCLMGJDDMFIMDKDOFBNDGN	19	4	1	1	1	1	1	6	25	5	4	1	0	2	0	0
5	AGLOMEDPIKJLPAADEAGAAAKLICKDEBJFKGNICBJD	2	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
6	AJFPKPNHAFMBNAENDMAKIILIHCBANJAAKIIDMJGI	3	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
7	AJJGIOMOIPHNONBNOHKECBMDHGLIJFMMMNFEHAFI	4	2	3	0	1	2	0	2	0	1	1	3	0	0	1	0
8	AKHCPFIOHNMOIKIEKGEPFDBHJFFLAPKGPFCIKOHI	11	0	0	1	1	0	0	0	10	5	0	2	0	0	0	0
9	ALBJNKGGODPLFDMIOOEMJCDDEFJGAJNIFMKCENFB	1	0	0	0	0	1	0	1	9	0	0	0	0	0	0	0
10	AMDIBDNLLJAENGDOMCNDHDMFOIJFHBGLBGDIEBCC	1	1	0	2	0	1	1	0	1	3	1	2	0	2	3	0
11	ANAAAOIDMJOFMOEHHIAPCKNMELAIFJOPMCBKBOHE	1	0	0	2	0	0	0	0	2	0	0	0	1	0	0	1

Figure 4.4: The first few records of the **tally dataset** for the second approach. It contained $792,197$ observations and 60 variables, excluding the KENTEKENVOORENC identifying variable.

The first dataset uses the same binary format as the first approach, just with the new time-slots. This is the **binary dataset**. An example of it can be found in Figure 4.3. The removal of singleton observations made it so the size of this dataset is only around 800 thousand observations. This is about half the size of dataset used in the first approach.

The second dataset addresses another issue. Instead of just checking whether or not a license plate has been seen in a certain time-slot, it counts how many times it has been seen. This is the **tally dataset**. Figure 4.4 shows an example. It retains much more information, but comes with its own problem. SLIM can only handle discrete data and occurrence counts are continuous. Thus, the data had to be discretised before it could be used.

4.1.2 Discretisation of occurrence counts

When discretising data there is always a loss of information. However this is not a bad thing as the resulting categories can be a lot more descriptive than just numbers. In this case, the aim was to divide the data into 5 categories. These categories were *Never, Rarely, Sometimes, Often and Very Often*. This specific number of categories was chosen for 2 reasons. It is high enough to properly differentiate between the categories, as evident by the distinctive names and it is low enough to keep complexity manageable.

Like with the time-slots, clustering was used to find good ranges for the categories. Doing clustering per time-slot is not an option as this obscures the relations between time-slots. Relations between time-slots, in the form of patterns, are the aim of this research. So the time-slots need to be clustered all at once.

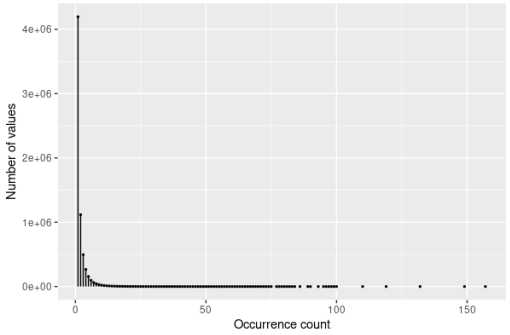
The **tally dataset** contained a big variety of occurrence counts. Many values of 0, as almost always vehicles are not present in all time-slots. So 0 was assigned the "Never" category and was left out of the clustering phase. There were also some vehicles that occurred very often. If clustering had been applied to this data, interesting relations would have been lost. For example, if a vehicle would have a high count in the first time-slot, and 10 times as much in the second time-slot, both time-slots would get the "Very Often" category. It would be better if the values would be considered relative to each other. So the first time-slot would get "Rarely" and the second would get "Very Often" in comparison.

Another reason why relative values would be better, is that given some very high values, all others would fall in the "Rarely" category. This is clearly visible in Figure 4.5a. Most of the occurrence counts relatively low. To address this, normalisation was applied to each observation, to bring all values between 0 and 1. The following normalisation function was used to ensure a diverse spread between 0 and 1.

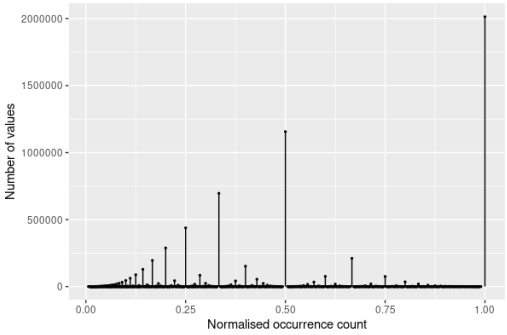
$$normalised\ observation = \frac{observation - min(observation)}{max(observation) - min(observation)} \tag{4.1}$$

Figure 4.5b shows the normalised distribution. It has much more spread than the original and thus is better to cluster.

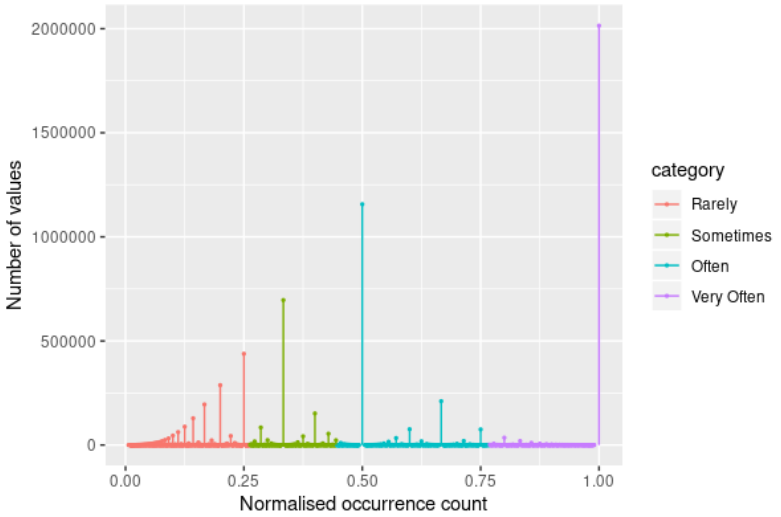
The normalised values were clustered using the `discretize` function from the `arules` package. This function runs K-Means internally and translates it back to the categories. The discretisation that resulted from this, visualised in Figure 4.5c. The time-slots, now with categorical values, were dummified so each time-slot and category combination became a binary variable. An example of the final **tally dataset** can be found in Figure 4.6. To come back on the number of time-slots that was chosen. It had to be kept at a reasonable amount because the discretisation and dummification resulted in a fivefold increase.



(a) Original



(b) Normalised



(c) Normalised and clustered.

Figure 4.5: Density plots for the different occurrence counts in the **tally dataset**.

	KENTEKENVOORENC	1_Never	1_Rarely	1_Sometimes	1_Often	1_Very.Often	2_Never	2_Rarely	2_Sometin
1	AAKOJKMJJOJLAFGKGKJFICOPBMLGKOFNHKAPCGGP	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
2	ABNLOBCHHCBFGHNKBOAMCAJBHNFIOBAIHKHIDJOP	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE
3	ADFIDPAIFPLFDEGFJPDHLFJCJBPIIFIELNACKHGK	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE
4	ADJMFENDLNJOCLGOILDCLMGJDDMFIMDKDOFBNDGN	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE
5	AGLOMEDPIKJLPAADEAGAAAKLICKDEBJFKGNICBJD	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE
6	AJFPKPNHAFMBNAENDMAKIILIHCBANJAAKIIDMJGI	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE
7	AJJGIOMOIPHNONBNOHKECBMDHGLIJFMMMNFEHAFI	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE
8	AKHCPFIOHNMOIKIEKGEPFDBHJFFLAPKGPFCIKOHI	FALSE	FALSE	FALSE	TRUE	FALSE	TRUE	FALSE	FALSE
9	ALBJNKGGODPLFDMIOOEMJCDDEFJGAJNIFMKCENFB	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
10	AMDIBDNLLJAENGDOMCNDHDMFOIJFHBGLBGDIEBCC	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE

Figure 4.6: The first few records of the descritised **tally dataset** for the second approach. It contained $792,197$ observations and 300 variables, excluding the KENTEKENVOORENC identifying variable.

4.1.3 Sampling

Sampling was still necessary even though the number of observations was cut in half. It was performed on the **binary dataset** with parameters $d = 59$, $\epsilon = 0.05$, and $\delta = 0.00001$. The resulting sample contained 225 thousand observations. It was also performed on the **tally dataset** with parameters $d = 60$, $\epsilon = 0.05$, and $\delta = 0.00001$. These parameters were chosen for the same reason as in the first approach. The resulting sample contained 228 thousand observations. This low d-index, even though the dataset has 300 variables, is due to the sparsity that dummification introduced. Only one of the categories is true for each time-slot after all.

4.2 Experiments

SLIM was applied to the sampled binary dataset twice. Once with a minimum support of 0.01 and once with 0.005.

SLIM was applied to the sampled tally dataset with a minimum support of 0.01. The prevalence of the "Never" category, however, was so large that it smothered all other categories. As such it was removed and represented in absence, "Never" is when all other categories have a false value. This is akin to the data of the first approach, where "not seen in a time-slot" did not have its own variable, but instead was just the false value for "seen in a time-slot".

On this reduced dataset, with 240 variables, SLIM was applied twice. Once with a minimum support of 0.01 and once with 0.005.

These values for minimum support were chosen for the same reason as in the first approach, however 0.001 was left out, as it resulted in too many patterns to analyse properly.

4.3 Results

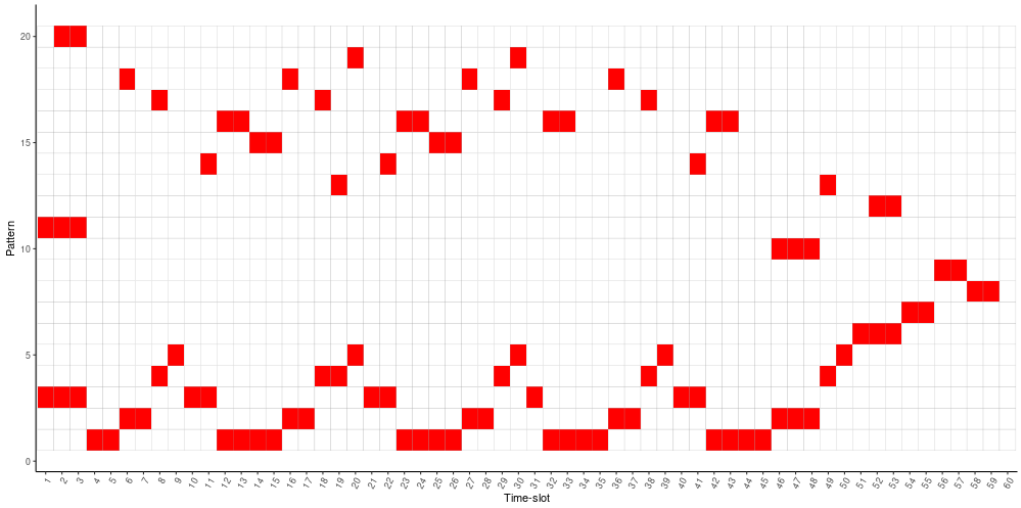
Running SLIM gave the following results:

- SLIM on the sampled binary dataset with minimum support of 0.01 yielded 109 longer patterns and the 60 singleton patterns with a compression of about 26%.
- SLIM on the sampled binary dataset with minimum support of 0.005 yielded 161 longer patterns and the 60 singleton patterns with a compression of about 28%.
- SLIM on the reduced and sampled tally dataset with minimum support of 0.01 yielded 29 longer patterns and the 240 singleton patterns with a compression of about 5%.
- SLIM on the reduced and sampled tally dataset with minimum support of 0.005 yielded 133 longer patterns and the 240 singleton patterns with a compression of about 10%.

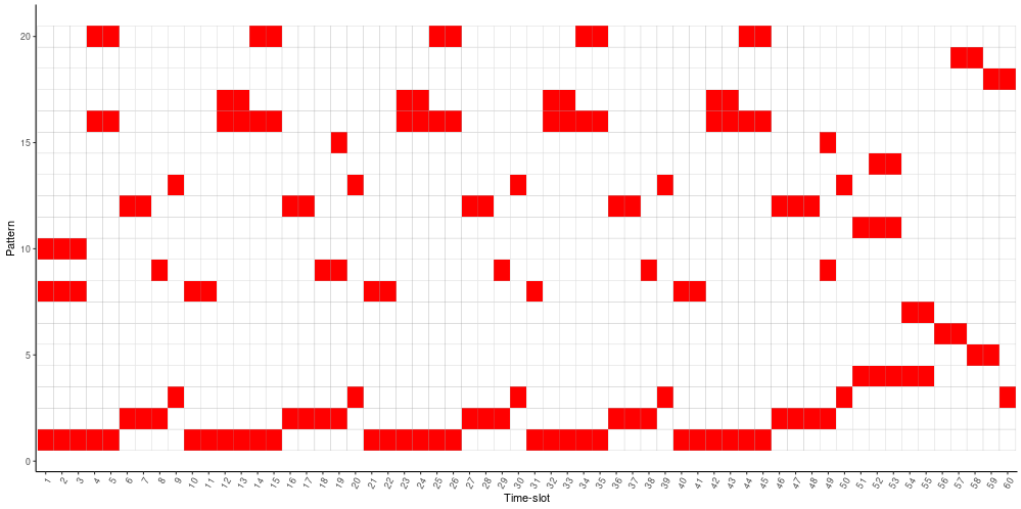
4.3.1 Binary dataset

The results of the binary dataset were comparable to the results of the first approach. This is no surprise as the input data is very similar, just with slightly different time-slots and without the singleton observations. The benefit of removing the singleton observations is immediately apparent. Now the minimum support of 0.01 is enough to get the night and weekend patterns and the compression has improved quite a bit.

The result of the improved time-slots were less obvious. Figure 4.7 shows the 20 patterns with the highest cover ratio, like Figure 3.4 did in the first approach. Compared to the first approach, the weekdays shared more similarities between each other. They followed almost the same structure, which was in accordance with the density plot of the observations. So the time-slots seemed to create more uniformity in the patterns.



(a) minimum support = 0.01



(b) minimum support = 0.005

Figure 4.7: Top 20 non-singleton patterns based on cover ratio. A red square means that that time-slot is included in that pattern. Results of running SLIM in the second approach on the binary dataset.

4.3.2 Tally dataset

The results of the tally dataset could not be visualised in the same manner as previous results. They simply had too many variables on the x-axis. That doesn't mean that nothing could be learned from these results. First of all, the compression is significantly lower than previous results.

Second, with a minimum support of 0.01, only 29 non-singleton patterns were found and all of them had a length of 2. A few of these are sequential time-slots in the same category, for example *{43_Rarely, 44_Rarely}*. These are likely caused by overlapping time-slots where, for this subgroup of the dataset, the time-slot windows could have been merged. The other patterns were seemingly random and difficult to analyse. One example was *{16_Rarely, 36_Rarely}*, which is Tuesday morning and Thursday early afternoon. Also most of the patterns were from the "Rarely" category and others were from the "Sometimes" category.

Third, with a minimum support of 0.005 the number of patterns increased drastically. From 29 to 133. The length of the patterns, however, did not increase that much. There were 8 patterns of length 4, 12 of length 3 and all others had a length of 2. There were some interesting patterns. For example *{12_Rarely, 23_Rarely, 32_Rarely, 42_Rarely}*, which is Monday evening/Tuesday night, Wednesday early morning, Thursday early morning, Friday early morning. This is a group of very early morning drivers, Possibly trucks that deliver stock for stores. Despite some anomalies, most of the results were quite difficult to interpret, because they were very short patterns. Again most of the patterns contained only the "Rarely" category.

Finally, to check if the minimum support might have been too high, some lower values were also tried. The results were way more patterns, but only small increases in compression and still almost all patterns with a length of 2.

4.4 Conclusion

The binary dataset gave an improvement compared to the first approach. Both the removal of singleton observations, as well as the improved time-slots added to this. A minimum support of 0.01 resulted in 26% compression compared to 20% and a minimum support of 0.005 resulted in 28% compression compared to 22%. Also the patterns that were found make sense intuitively, just like before.

The increased complexity of the tally dataset, together with the inability to create a proper visualisation, made it difficult to analyse. The patterns that were found did make some sense. Most of them were subsets of the patterns that were found in the binary dataset. This suggests that the simplification made in the binary dataset leads to longer patterns. It does this by grouping multiple patterns together when simplifying.

In conclusion the goal of finding interesting patterns in a more complex dataset has partially been reached. Removing arbitrary simplifications gave significant improvements, but adding a lot of complexity gave rise to a difficult visualisation issue.

4.5 Discussion

Most issues from the first approach were addressed in the second approach. However, one remains and one new one was added. The lack of visualisation when complexity increases is a big issue. It makes the analysis very difficult and it also makes it hard to present the result in a clear way. A possible way it could be addressed also addresses the remaining issue of the grouping variable from the first approach. By adding more variables or grouping on different variables, other visualisations could be made. In the third approach, both of these are applied.

Chapter 5

Third Approach: New data and network models

The third approach added more data and grouped on different data. The goal was to research if this could provide better visualisation and more interesting results.

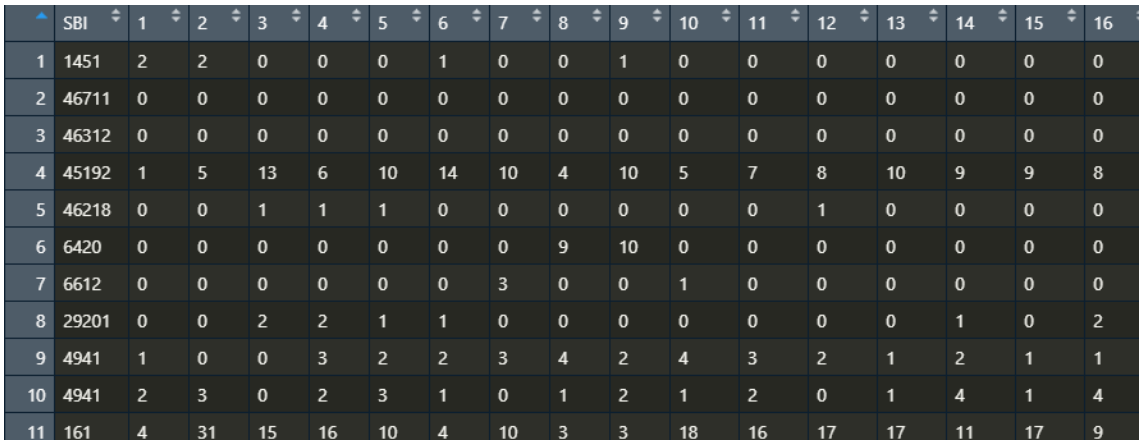
5.1 Data

It started out with the non-descritised tally dataset from the second approach, refer to Figure 4.4 for an example. The KENTEKENVOORENC variable, representing the encoded license plate on the front of the vehicle, was replaced by its decoded counterpart. This way it could be coupled with new data. The decoded license plates were only used for coupling.

ID-lab had a dataset containing information about businesses registered in the Netherlands. It contained both the KVK numbers as well as their respective SBI codes. It was created on 17-05-2019.

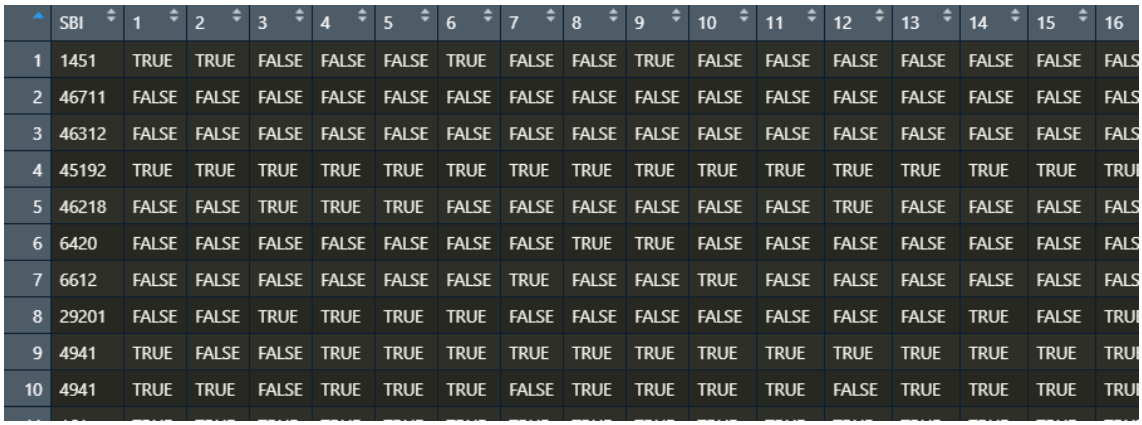
They also had a dataset which connects license plates to KVK numbers. This was provided by the RDW, the Netherlands Vehicle Authority, on 03-07-2019. The tally dataset with the decoded license plates was joined with this dataset. Grouping was then done on KVK numbers instead of license plates. Observations with the same KVK number were merged by summing them together. Afterwards the SBI codes were added from the KVK number. For KVK number with more than one SBI code, the first code of the main activity, the HOOFDACT variable, was used. An example of the resulting dataset can be found in Figure 5.1. Encoding the KVK numbers, like the license plates, was unnecessary as they served as the identifying grouping variable and wouldn't be considered by the model in any case.

From this dataset 2 input dataset for SLIM were created. One with simplified binary values, like the binary dataset from the second approach. An example of which is shown in Figure 5.2. The other with descritisation like the tally dataset, as shown in Figure 5.3. As the number of observations was relatively low, sampling was not needed.



	SBI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1451	2	2	0	0	0	1	0	0	1	0	0	0	0	0	0	0
2	46711	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	46312	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	45192	1	5	13	6	10	14	10	4	10	5	7	8	10	9	9	8
5	46218	0	0	1	1	1	0	0	0	0	0	0	1	0	0	0	0
6	6420	0	0	0	0	0	0	0	9	10	0	0	0	0	0	0	0
7	6612	0	0	0	0	0	0	3	0	0	1	0	0	0	0	0	0
8	29201	0	0	2	2	1	1	0	0	0	0	0	0	0	1	0	2
9	4941	1	0	0	3	2	2	3	4	2	4	3	2	1	2	1	1
10	4941	2	3	0	2	3	1	0	1	2	1	2	0	1	4	1	4
11	161	4	31	15	16	10	4	10	3	3	18	16	17	17	11	17	9

Figure 5.1: The first few records of the dataset for the third approach. It contained 17295 observations and 61 variables. The KVK identifying variable was excluding as it could be considered sensitive information.



	SBI	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1451	TRUE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALS
2	46711	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALS
3	46312	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALS
4	45192	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRU
5	46218	FALSE	FALSE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALS
6	6420	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALS
7	6612	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALS
8	29201	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	TRU
9	4941	TRUE	FALSE	FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRU
10	4941	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	TRUE	TRU

Figure 5.2: The first few records of the binary dataset for the third approach. It contained 17295 observations and 61 variables. The KVK identifying variable was excluding as it could be considered sensitive information.

Both the tally and binary dataset were chosen, despite the relatively poor performance of the tally dataset in the second approach. The reason for this was that it was that a new visualisation method might prove to be especially beneficial for the tally dataset, as that was very hard to visualise in the previous approach.

	SBI	1_Rarely	1_Sometimes	1_Often	1_Very.Often	2_Rarely	2_Sometimes	2_Often	2_Very.Often	3_Rarely
1	1451	FALSE	TRUE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
2	46711	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
3	46312	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
4	45192	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE	FALSE
5	46218	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
6	6420	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
7	6612	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
8	29201	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
9	4941	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
10	4941	FALSE	TRUE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE	FALSE
11	161	TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	TRUE	FALSE

Figure 5.3: The first few records of the tally dataset for the third approach. It contained 17295 observations and 241 variables. The KVK identifying variable was excluding as it could be considered sensitive information.

5.2 Experiments

For both datasets the same experiment was executed. First, subsets of the data were created by splitting on SBI code. Only subsets that contained more than 1% of the total number of observations were considered. Then, SLIM was applied to each of these subsets with a minimum support of 0.01.

The 1% of the total data was chosen to indirectly mandate a minimum support of 0.01 on the total dataset. As the number of observations in these datasets was quite low compared to the previous approaches, 0.01 was chosen to keep the absolute minimum support, the number of actual observations needed to reach minimum support, to a reasonable value of around 170.

The 0.01 minimum support of the subsets on which SLIM was actually executed, is admittedly chosen without good reason, instead merely because it was the default value given to the parameter in the code. Due to time constraints, other values could not be tested.

The results were converted to a network where each SBI code has an directed edge towards all the patterns that occurred in their respective code-table. These networks were created and visualised in Gephi.

	SBI	size	description
1	4941	3484	Goederenvervoer over de weg (geen verhuizingen)
2	6420	2280	Financi<eb>le holdings
3	45112	503	Handel in en reparatie van personenauto's en lichte bedrijfs...
4	70102	429	Holdings (geen financi<eb>le)
5	161	301	Dienstverlening voor de akker- en/of tuinbouw
6	45192	268	Handel in en reparatie van zwaardere bedrijfsauto's (geen i...
7	4622	255	Groothandel in bloemen en planten
8	77399	244	Verhuur en lease van overige machines en werktuigen en va...
9	52291	230	Expediteurs, cargadoors, bevrachters en andere tussenperso...
10	7732	198	Verhuur en lease van machines en installaties voor de bouw
11	4120	175	Algemene burgerlijke en utiliteitsbouw

Figure 5.4: The 11 SBI codes that had a large enough subset size.

5.3 Results

The resulting networks both contain the exact same SBI codes. This is logical, as the amount of observations and the SBI variable are the same in both datasets. There are 11 SBI codes which passed the threshold of being in at least 1% of the observations. They, together with their subset size and a Dutch description, can be found in Figure 5.4.

By far the biggest subsets were SBI codes 4941 and 6420, they represent "Freight transport by road" and "Financial Holdings" respectively. Both were not surprising. The first is the broadest category a business can pick when operating some kind of commercial transport vehicle. The second is an umbrella term for parent companies, under which multiple child businesses could be constructed. The vehicles can be registered with the parent company, while they may be used by the child business. Many businesses work with a financial holding to minimise risks to their assets.

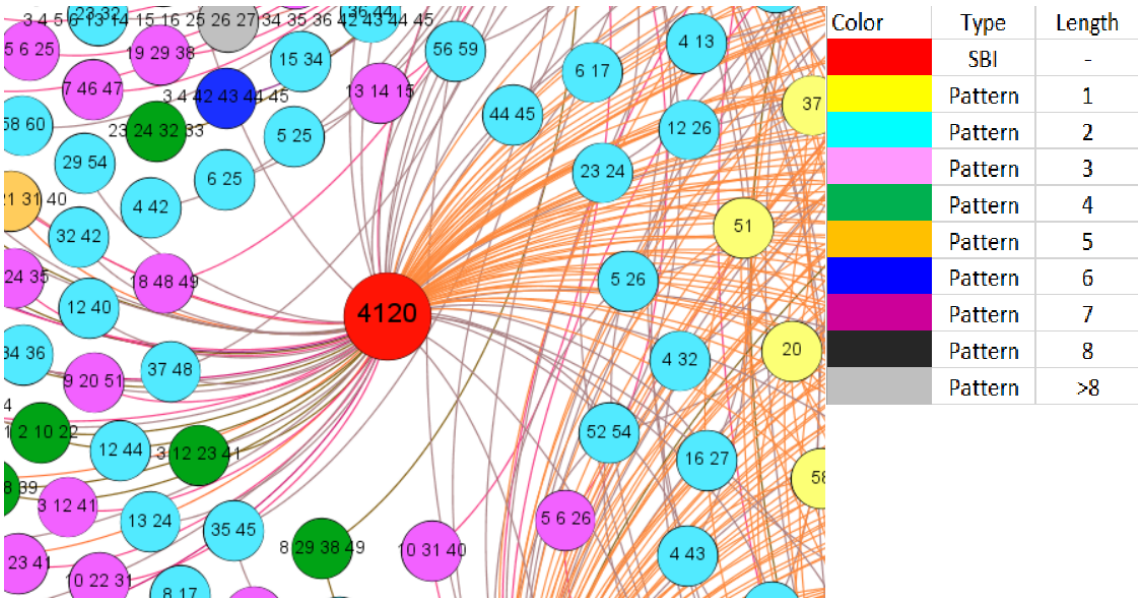


Figure 5.5: Example of how to interpret the network. Legend shows the colours of the different types of nodes.

Figure 5.5 gives an example of how to interpret the networks. The network has two distinct kinds of nodes. SBI codes, the big red nodes, and patterns, all other nodes. Patterns are coloured based on length. Singleton patterns are yellow, while patterns with a length of 2 are light-blue. Patterns up to a length of 8 have distinct colours, all patterns longer than 8 are grey. This enables one to distinguish patterns in a visual way. Nodes are also labelled with their respective SBI code or pattern. This labelling is removed in larger pictures for the sake of visibility.

Edges always go from an SBI node to a pattern node. If an edge runs between an SBI code and a pattern, that signifies that that pattern is included in the code-table created by subsetting the data on that SBI code. Node size is dependent on the number of edges a node has, which is why the SBI nodes are bigger.

5.3.1 Binary dataset

The full network for the binary dataset is visualised in Figure 5.6. It uses the same colouring as the example. This network is very pretty, but also a bit overbearing. Many edges go to the singleton patterns and many patterns are only connected to a single SBI code. Luckily Gephi provides filtering tools for networks. Figure 5.7 shows the network without singleton patterns and only with pattern nodes that connect to 2 or more SBI nodes.

With these filters finding interesting patterns was easier. Patterns with a high length immediately stand out. For example, the highest dark green node in the centre has the pattern *{51, 52, 53, 54}*, which is Saturday early morning until noon. This pattern is connected to 4 SBI codes, 4941, 6420, 4622 and 77399. Those first two have been explained already. They are umbrella groups so they have edges to lots of patterns. However it is a little surprising to see them on Saturday morning as that is a quiet period according to Figure 4.1 from the second approach.

The other two are "Wholesale of flowers and plants" and "Rental of machines, tools and other goods" respectively. That first one is not surprising given a little bit of domain knowledge. Growers of flowers and plants take the harvest of Friday to the flower auction on Saturday morning. The second one may also have such a logical explanation, however it may also be unknown information. It is hard to say without domain knowledge.

Another interesting pattern is the orange node very close to the previously discussed node. It has the pattern *1, 10, 21, 31, 40*, which are weekday night, with the exception of time-slot 10. This mainly night drivers pattern is connected to 4 SBI codes, 4941, 6420, 70102 and 7732. The first two are the previously discussed umbrella codes. The other two are "Non-financial Holdings" and "Rental of construction machines and installations". It makes sense that large construction machines, like cranes, are transported by night. They can be very slow and some even need their own convoy of signalling vehicles.

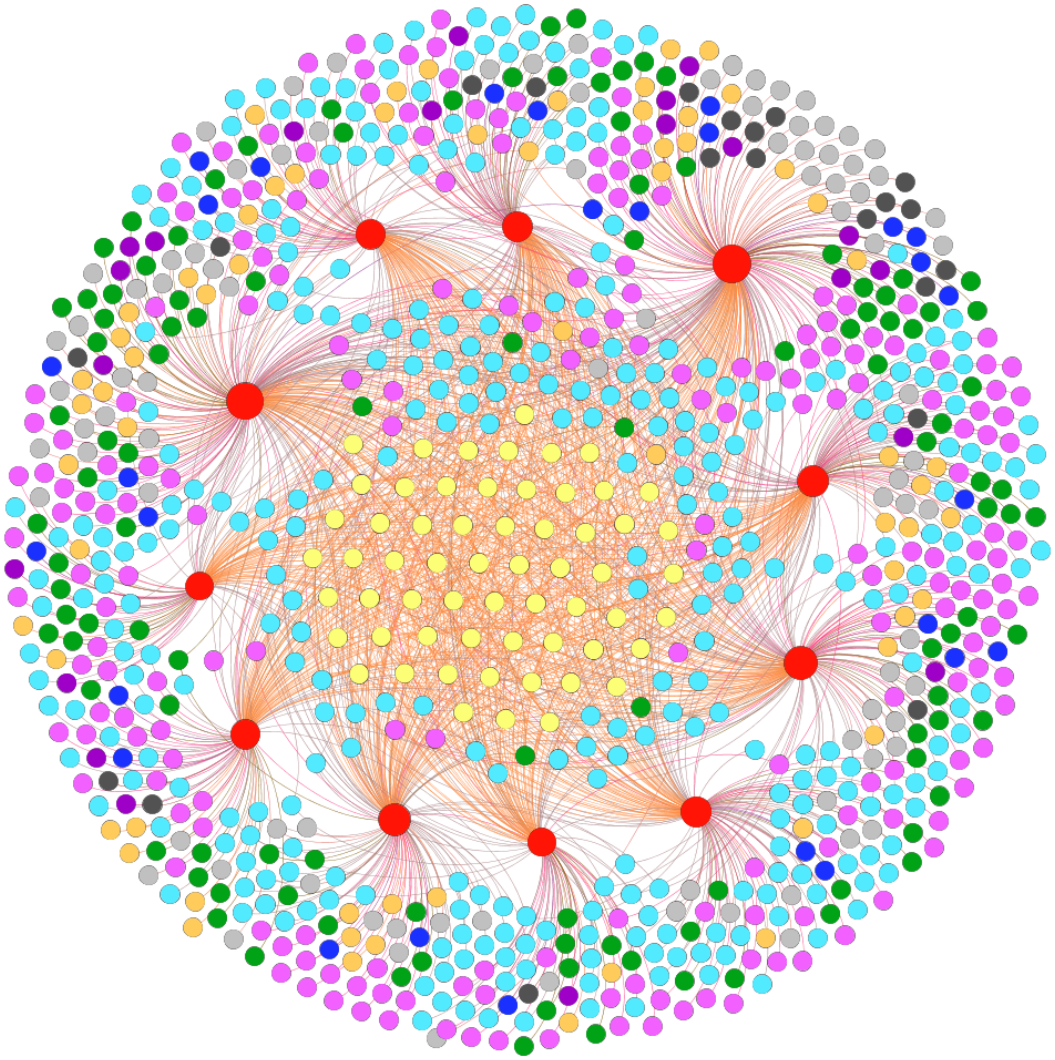


Figure 5.6: Full network for the binary dataset in the third approach.

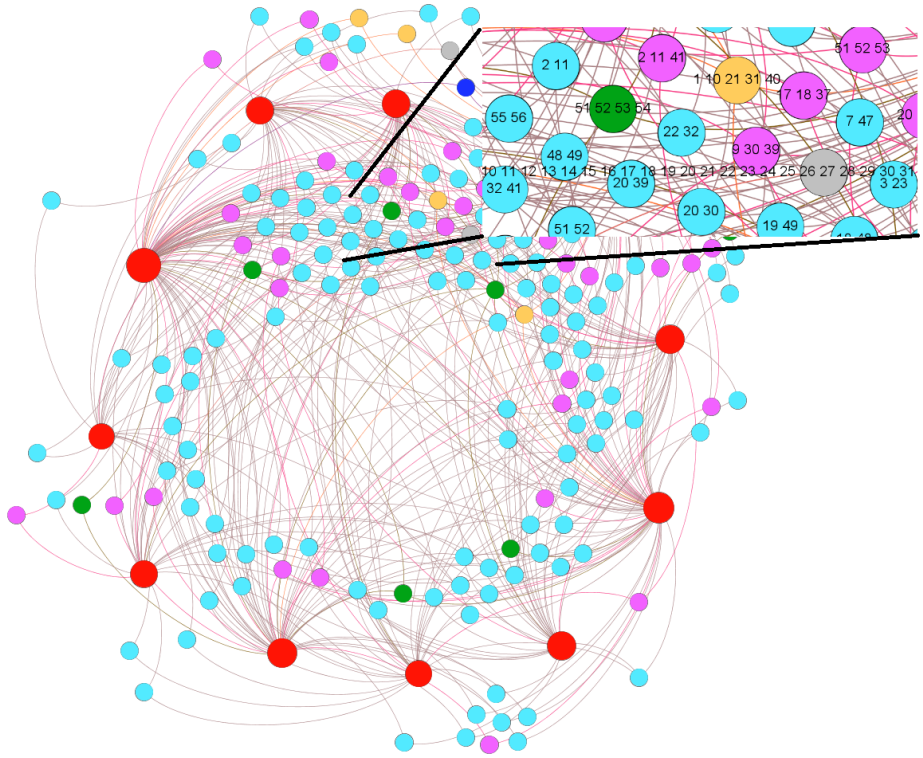


Figure 5.7: Network without singleton patterns and patterns only connected to one SBI code. Includes a zoomed in portion with labelling. Generated on the binary dataset in the third approach.

Apart from more interpretable patterns there are also some extremely long patterns. The grey node very close to the previously discussed nodes has such a pattern. It contains all time-slots from 1 until 49, which is Monday early morning until Friday mid-afternoon. To better analyse the pattern, the original input data was filtered with it. It was found that there were a substantial number of observations that contained this pattern, around 1750 observations, which is 10% of the total.

Most of these where to be expected, businesses like transportation companies and car rental services. Their vehicles are so often on the road that they are very likely to be in many time-slots. Others were more surprising, like an interior construction company and a trader in personal property. Knowledge from domain experts is needed to conclude if there is a logical explanation for companies in those sectors having such divers travel times.

5.3.2 Tally dataset

The full network for the tally dataset is visualised in Figure 5.8. It uses the same colouring as the example. Like with the previous network, it is very pretty, but also a bit overbearing. Many edges go to the singleton patterns and many patterns are only connected to a single SBI code. Figure 5.7 shows the network without singleton patterns and only with pattern nodes that connect to 2 or more SBI nodes.

Finding long patterns with multiple connections is as easy as in the previous network, but interpreting them is quite a bit harder. For example take the purple node in the zoomed in portion of Figure 5.9. It has the pattern *9_Rarely, 20_Rarely, 30_Rarely, 39_Rarely* and is connected to 6 SBI codes. These are 4149, 6420, 52291, 70102, 77399 and 45192, all of which are either general transport, holdings or rental businesses. The time-slots in which they rarely occur are Monday, Tuesday and Thursday afternoon and Thursday early morning. It is hard to say why this might be the case, possibly they have afternoon resting periods.

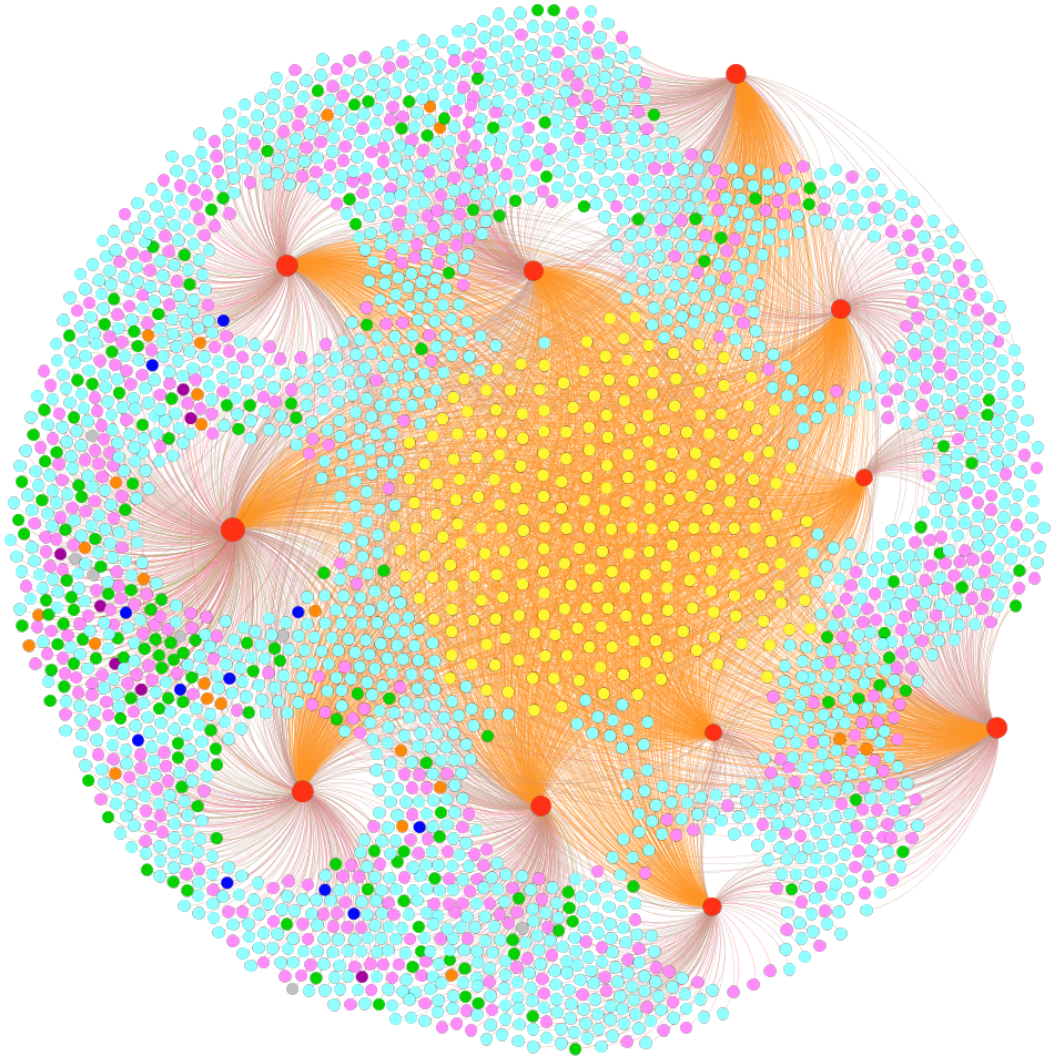


Figure 5.8: Full network for the tally dataset in the third approach.

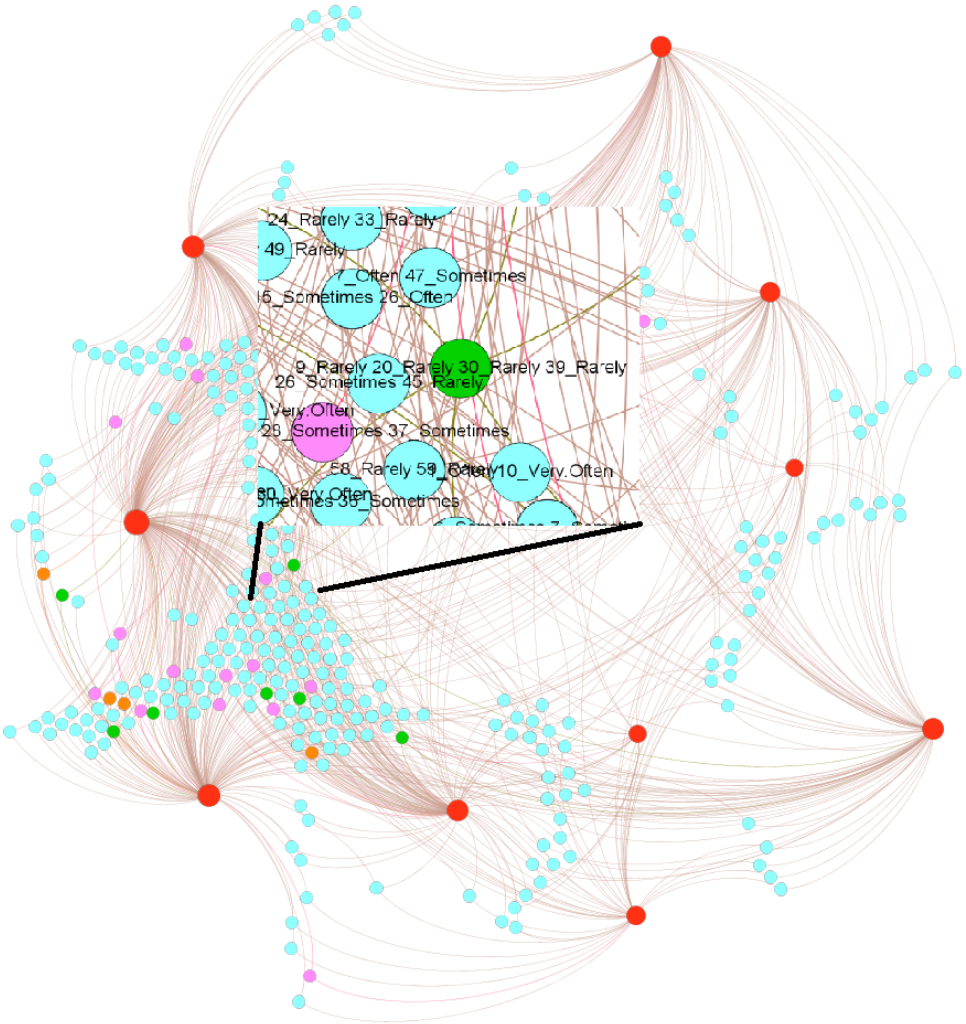


Figure 5.9: Network without singleton patterns and patterns only connected to one SBI code. Includes a zoomed in portion with labelling. Generated on the tally dataset in the third approach.

5.4 Conclusion

Networks turned out to be a very useful visualisation for pattern sets, especially with an extra grouping variable. Instead of analysing multiple different results, one for each SBI code, with a network all can be done at once, including relations between them. The filtering that can be done on networks provides a method to look at both the individual and connected results, without having to compare their heatmaps like what was done in the first approach.

Finding patterns in the binary set went well. As with the previous approaches, the simplicity of the time-slots, makes for patterns that are easy to interpret. The colouring of the nodes combined with the filtering tools made finding patterns a lot less tedious and if needed the relations with the original data could still be made. For this kind of data, it is a successful new method.

This is in contrast to the tally dataset. Interpreting the meaning behind patterns with the categories is still difficult. The relative relations between the categories proved hard to translate to meaning-full context. The discretisation both loses information and makes it harder to interpret the results. It also makes the network far larger. For this kind of data, pattern set mining, only being able to perform well with discrete data, is not the right choice.

5.5 Discussion

Although this approach addressed the issues raised by the previous approaches, it also created some new issues, mainly concerning the newly added data.

KVK numbers and SBI codes are only available for business, and thus vehicles, that are registered in the Netherlands. So using these requires throwing out all the foreign license plates. This might be an undesirable outcome. Ideally, a similar European data source would be used, containing license plate information on an international scale.

Another issue is that the SBI codes are chosen by the company themselves. Often when a company first registers, these codes are correct. Later one, when the company changes its activity, there is no requirement to update the registered SBI code. From that point on, the SBI code would be incorrect. Currently some organisations in the Netherlands are working on a replacement for SBI codes, based on information about the business collected via web scraping. If this is successful, it would be a better alternative.

Some of the SBI codes were also these very broad umbrella terms. If those would be left out, the results could differ quite a bit. However, those umbrella codes account for a lot of the observations, so the remaining dataset might have too few observations to really analyse using big data techniques.

Lastly, The minimum support parameters were not varied at all for this approach. Changing it for the subsets on which SLIM was run could have been beneficial, possibly to decrease the number of patterns for the tally dataset and making the results more interpretable. Lowering the support threshold for subset inclusion, would have revealed more SBI codes that could have been interesting. Varying these parameters is left to further research.

Chapter 6

Conclusion

Let's start the final conclusion with a recap of the research questions. These were the following.

1. **To what extend can pattern set mining prove to be a useful method for ID-lab?**
2. **Can interesting travel time patterns be extracted from the WiM data using pattern set mining?**

The extend to which pattern set mining can prove to be useful for ID-lab seems to be quite dependent on the data. It can give great results, which are easy to interpret, even the more complex results like the very long pattern from the third approach. Combined with a network representation to visualise the results it is a powerful method for finding interesting knowledge in the data. It also outperformed the relatively well-known clustering method K-Means.

There is also a downside, which is why it depends heaviliy on the data. If the data contains continuous variables, they need to be discretised. Proper discretisation is a research problem on its own. Even if they are discretised very well, it makes interpretation of the results quite a bit more difficult, as was shown in the results of the tally dataset.

In conclusion, pattern site mining can prove to be useful for ID-lab on discrete data. On discretised continuous data the results are quite difficult to interpret and another method that doesn't require discretisation is better.

The second question can be answered with yes, interesting travel time patterns can be extracted for the WiM data using pattern set mining. To expand on it a little bit, in the approaches some interesting patterns have already been discussed. Some of which could be previously unknown information for the ILT. This could be interesting to know for the ILT, as it could relate to inspections. For example, on of the patterns concluded that interior constructions drive during the entire workweek, including the nights. This might be suspicious behaviour to an inspector and as such could lead to new insights about offences.

Chapter 7

Discussion

This sections serves to discuss the limitations, other choices and further research paths that could be taken.

First of all, the notion of interestingness of a pattern is often used. In chapter 2, it is explained that the quality measure of SLIM is the compression. It is also stated that patterns that are more frequent can add more to the compression. As such for SLIM there is little difference between interesting and frequent. So SLIM won't find patterns that are not frequent, even if they might be interesting.

Second, to reiterate the conclusion of the second research question, many interesting patterns might still be found in the results. An extension of the third approach is an interesting area for further research. Like mentioned in the discussion of the third approach, there are a few options for improvement. The most approachable one is varying the parameters of this research.

Third, The choice was made to represent travel times as weekdays. This could have also been multiple weeks, months or even years. The data could even be represented as a continuous stream of incoming data, where each observation is supplied one after the other, instead of all at once. SLIM would not be suitable for such a format, but other pattern set mining techniques would be. As discussed in chapter 2, a version of the KRIMP, created by Van Leeuwen and Siebes (2008) is especially suitable for data streams.

Fourth, next to cover ratio and networks, some other methods of filtering patterns on criteria were tried. The first one was to actually cover the observations with the resulting code-table and try to create a network from this cover. The second was to calculate the gain in compression for each pattern by leaving it out and then determining the compression again. Both of these methods were too computationally expensive to execute properly. Given more research they could be more efficiently implemented, and would be actual feasible options.

A big limitation of this research were the possible risks of the data quality. The license plates are often read incorrectly by the WiM systems. This was partially addressed by removing the singleton observations, but no other checks on license plate validity were performed. Furthermore, the SBI codes lack control as stated in the third approach,

this is an area of further research. Given a subset of data for which it is assumed that the SBI codes are correct, the method in the third approach can be applied. Instead of finding interesting patterns, the resulting code tables can be used to classify the SBI codes of unseen observations.

Another core issue was also about data collection. The observations density plot found in Figure 4.1 can be created for each WiM site. This makes very clear that some sites do not function properly as they have a fraction of the observations compared to their counterparts on the other site of the road. This data was left in as it was only a fraction of the complete dataset, however it does have some impact on the result, even if it is just little bit.

An issue for ID-lab could be the difficulty of use. Currently the SLIM algorithm is only available as source code and without a graphical interface. It can only be run from the command line with some configuration files for setting parameters. This is enough for research, but not for common use. An interface was written in R for this research, however, it is rudimentary and still not very suitable for production. This issue might be addressed in one or two years. Currently there are plans to add SLIM and other pattern set mining algorithms to Pythons well-known scikit-learn package. As ID-lab has many people familiar with Python this would lower the threshold of using pattern set mining quite a bit.

Finally, on a completely different note. The WiM data contains both front and back license plates. They can differ from each other as explained in the first approach. An interesting area of research could be to create a network of front and back plates. There would likely be some separate clusters withing this network and pattern set mining could be used to describe those clusters.

Bibliography

Adams, Fred C and Gregory Laughlin (1997). "A dying universe: the long-term fate and evolutionof astrophysical objects". In: *Reviews of Modern Physics* 69.2, p. 337.

Agrawal, Rakesh et al. (1993). "Mining association rules between sets of items in large databases". In: *ACM SIGMOD Record* 22.2, pp. 207–216. DOI: 10.1145/170036.170072.

Bastian, Mathieu et al. (2009). "Gephi: An Open Source Software for Exploring and Manipulating Networks". In: URL: http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154.

Bonchi, Francesco et al. (2003). "Exante: Anticipated data reduction in constrained pattern mining". In: *Proceeding of the 7th European Conference on Principles of Data Mining and Knowledge Discovery*, pp. 59–70.

Bringmann, Bjorn and Albrecht Zimmermann (2007). "The Chosen Few: On Identifying Valuable Patterns". In: *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. DOI: 10.1109/icdm.2007.85.

Calders, Toon and Bart Goethals (2007). "Non-derivable itemset mining". In: *Data Mining and Knowledge Discovery* 14.2, pp. 171–206.

Dowle, Matt and Arun Srinivasan (2019). *data.table: Extension of 'data.frame'*. R package version 1.12.8. URL: https://CRAN.R-project.org/package=data.table.

Gandhi, Manan et al. (2015). *Slim source code & binaries for Windows and Linux (20th July 2015)*. URL: https://people.mmci.uni-saarland.de/~jilles/prj/slim/.

Hahsler, Michael et al. (2005). "arules - A Computational Environment for Mining Association Rules and Frequent Item Sets". In: *Journal of Statistical Software, Articles* 14.15, pp. 1–25. ISSN: 1548-7660. DOI: 10.18637/jss.v014.i15. URL: https://www.jstatsoft.org/v014/i15.

Human Environment and Transportation Inspectorate (Sept. 17, 2019). *Meerjarenplan 2020-2024*. Document describing the goals the Human Environment and Transportation Inspectorate wants to focus on in the years 2020–2024, Section 2, Program 5.

Kaplan, Jacob (2020). *fastDummies: Fast Creation of Dummy (Binary) Columns and Rows from Categorical Variables*. R package version 1.6.1. URL: https://CRAN.R-project.org/package=fastDummies.

Kolmogorov, A. N. (1968). "Three approaches to the quantitative definition of information*". In: *International Journal of Computer Mathematics* 2.1-4, pp. 157–168. DOI: 10.1080/00207166808803030.

Kuang, Kevin et al. (2019). *pbmcapply: Tracking the Progress of Mc*pply with Progress Bar*. R package version 1.5.0. URL: https://CRAN.R-project.org/package=pbmcapply.

Labarrere, J (2019). *What is Weigh-in-Motion and History*. URL: http://www.is-wim.org/index.php?nm=2&nsm=1&lg=en.

MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations". In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Berkeley, Calif.: University of California Press, pp. 281–297. URL: https://projecteuclid.org/euclid.bsmsp/1200512992.

Ministry of the Interior and Kingdom Relations (2018). *Actieplan Open Overheid 2018-2020*. URL: https://www.open-overheid.nl/wp-content/uploads/2018/11/digitale-versie-vouwfolder.pdf.

Pasquier, Nicolas et al. (1999). "Efficient mining of association rules using closed itemset lattices". In: *Journal of Information Systems* 24.1, pp. 25–46.

Pei, Jian et al. (2004). "Pushing convertible constraints in frequent itemset mining". In: *Data Mining and Knowledge Discovery* 8.3, pp. 227–252.

R Core Team (2013). *R: A Language and Environment for Statistical Computing*. ISBN 3-900051-07-0. R Foundation for Statistical Computing. Vienna, Austria. URL: http://www.R-project.org/.

Riondato, Matteo and Eli Upfal (2014). "Efficient Discovery of Association Rules and Frequent Itemsets through Sampling with Tight Performance Guarantees". In: *ACM Transactions on Knowledge Discovery from Data* 8.4, pp. 1–32. DOI: 10.1145/2629586.

Rissanen, J. (1978). "Modeling by shortest data description". In: *Automatica* 14.5, pp. 465–471. DOI: 10.1016/0005-1098(78)90005-5.

RStudio Team (2019). *RStudio: Integrated Development Environment for R*. RStudio, Inc. Boston, MA. URL: http://www.rstudio.com/.

Shannon, Claude E (1948). "A mathematical theory of communication". In: *Bell system technical journal* 27.3, pp. 379–423.

Smets, Koen and Jilles Vreeken (2012). "Slim: Directly Mining Descriptive Patterns". In: *Proceedings of the 2012 SIAM International Conference on Data Mining*. DOI: 10.1137/1.9781611972825.21.

Van Leeuwen, Matthijs and Arno Siebes (2008). "StreamKrimp: Detecting Change in Data Streams". In: *Proceedings of the 2008 European Conference of Machine Learning and Knowledge Discovery in Databases (ECML PKDD)* 1.2, pp. 765–774.

Vreeken, Jilles, Matthijs van Leeuwen, et al. (2011). "Krimp: mining itemsets that compress". In: *Data Mining and Knowledge Discovery* 23.1, pp. 169–214. DOI: 10.1007/s10618-010-0202-x.

Vreeken, Jilles and Arno Siebes (2008). "Filling in the Blanks – Krimp Minimisation for Missing Data." In: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM)* 1.1, pp. 1067–1072.

Wickham, Hadley et al. (2019). "Welcome to the tidyverse". In: *Journal of Open Source Software* 4.43, p. 1686. DOI: 10.21105/joss.01686.

Appendix A

Code of SLIM interface in R

The main function is *SLIM_for_binary()*, which returns an object containing the results of the experiment.

```
# Functions to provide an R Interface to run SLIM.

# Import require libraries.
require(data.table)
require(tidyverse)
require(arules)

# Update a config (.conf) file
# by replacing the value of a specific variable.
update_config <- function(file, variable, value){
  f <- read_lines(file)
  regexpattern <- paste0("^(", variable, " = ).*$")
  parameter <- paste(variable, value, sep = " = ")
  f <- str_replace(f, regexpattern, parameter)
  write_lines(f, file)
}

# Run steps for translating the .dat file
# and excuting the algoritm.
execute_Slim <- function(resultName, absMinSup,
                         resultPath, slimFolder){

  # Change working directory so the call to SLIM works,
  # reset regardless of execution success.
  oldwd <- getwd()

  tryCatch(
    {
      setwd(slimFolder)

      update_config("fic.user.conf", "datadir", resultPath)
      update_config("fic.user.conf", "expDir", resultPath)
```

```
      update_config("convertdb.conf", "dbName", resultName)
      system(paste("./fic", "convertdb"))

      update_config("analysedb.conf", "dbName", resultName)
      system(paste("./fic", "analysedb"))

      iscName <- paste0(resultName, "-gain-", absMinSup, "d")
      update_config("compress.conf", "iscName", iscName)
      system(paste("./fic", "compress"))
    },
    error=function(cond) {
      message(cond)
      return(NA)
    },
    warning=function(cond) {
      message(cond)
      return(NULL)
    },
    finally={
      #Reset working directory
      setwd(oldwd)
    }
  )
}

# Create a .dat file from the data,
# return a translation table of the variables.
write_dat <- function(data, resultName, resultPath){

    # Change variable names to numbers,
    # this is required by the algorithm.
    cols_new <- as.character(0:(ncol(data) - 1))

    # Create a table to save the original variables.
    translation_table <- colnames(data)
    names(translation_table) <- cols_new

    data %>%
      rename_at(vars(colnames(data)), ~cols_new) %>%
      as("transactions") %>%
      write(file=paste0(resultPath, resultName, ".dat"),
            format="basket", sep=" ", quote=FALSE)

    return(translation_table)
}
```

```
# Translate the results from the code-table
# back into the originals variables.
translate_codetable <- function(resultName, resultPath,
                    ctpath, translation_table, type){

  codetable <- paste0(resultPath, ctpath)
  cat("Translating codetable...")

  f <- read_file(paste0(resultPath, resultName,
                        ".db.analysis.txt"))
  matches <- str_match_all(f, "[0-9]+=>[0-9]+")[[1]]
  splits <- str_split(matches, "=>", simplify = TRUE)
  model_table <- as.integer(splits[,2])
  names(model_table) <- splits[,1]

  #Change the actual pattern in the code-table file
  # to create the result file.
  lines <- read_lines(codetable)
  #Remove first two lines as they don't need translation.
  lines <- lines[-1:-2]

  #Translate each subsequent line.
  for (i in 1:length(lines)){
    lines[i] <- translate_pattern(lines[i], model_table,
                                  translation_table)
  }

  #Add csv header to the beginning of the vector.
  lines <- c("Pattern, Cover Occurrence, Total Occurrence",
             lines)
  cat(paste0("Done!\nWriting to file ", type, ".csv ..."))
  write_lines(lines, paste0(resultPath, type, ".csv"))
  cat("Done\n")
}

#translate a single line of the codetable.
translate_pattern <- function(line, model_table,
                              translation_table){
  # All lines have this format: '0 5 8 29 (123,123)\n'
  # Only numbers before the parentheses are variables,
  # the ones within parentheses are cover
  # and total occurrence rates.
  split <- str_split(line, " \\(|\\)") %>% flatten()
   #['0 5 8 29', '123,123', ' ']
  head <- split[[1]] #'0 5 8 29'
  tail <- split[[2]] #'123,123'

  cts <- str_split(head, " ") %>% flatten()
```

```
   #['0','5','8','29']
  dbs <- map_int(cts, ~model_table[[.]]) #[2,34,0,1]
  dbs <- sort(dbs)
   #[0,1,2,34], sort them as integers so the order is correct.
  result <- map(dbs, ~translation_table[[as.character(.)]])
   #['MO0-3', 'MO3-6', 'MO6-9', 'FR6-9']
  result <- do.call(paste, result)
   #'MO0-3 MO3-6 MO6-9 FR6-9'

  paste(result, tail, sep=",")
   #'MO0-3 MO3-6 MO6-9 FR6-9,123,123'
}

# Create object to save the results of running SLIM.
SLIM_result <- setClass("SLIM result",
    slots=list(resultName="character",
        minimumSupport="numeric",
        codetable="data.frame", patternset="vector",
        standardtable="data.frame", absMinSup="numeric"))

# Run SLIM on a binary dataset with the given parameters.
# Note that the folders may need to be changed
# to run in another location.
SLIM_for_binary <- function(data, resultName,
                minimumSupport = 0.01,
                resultFolder = "2_Data/4_Models/Slim/",
                slimFolder = "1_Scripts/Slim/",
                absMinSup = NULL,
                projectPath = "/data/rstudio/WIM-Analyse/",
                overwrite = FALSE){

  #Step 1: Create folder and the .dat file of the data in.
  cat("------------------------------------ Setup:
  ------------------------------------\n")

  resultPath <- paste(projectPath, resultFolder,
                resultName, "/", sep="")

  if (dir.exists(resultPath)) {
    if (!overwrite) {
      stop("Interrupted: Result folder already exists
        and will not be overwritten.
           If you want to overwrite it,
           set 'overwrite = TRUE'.")
    } else {
      cat("Overwriting contents of folder...")
      unlink(resultPath, recursive = TRUE, force = TRUE)
      cat("Done\n")
```

```
  }
}
cat("Creating results folder...")
dir.create(resultPath)
cat("Done\n")

cat("Converting data to .dat file...")
translation_table <- write_dat(data,
              resultName, resultPath)
cat("Done\n")

#Step 2: Run SLIM on the .dat file

cat("-------------------------------------
Executing SLIM steps:
-------------------------------------\n")
#Calculate absolute minimum support if not given
if(is.null(absMinSup)){
  absMinSup <- round(minimumSupport * nrow(data))
}
execute_Slim(resultName, absMinSup,
              resultPath, slimFolder)


#Step 3: Translate SLIM results back to
# original variables.
cat("-------------------------------------
Translating codetables:
-------------------------------------\n")

#Find both code table files, there should be 2:
#The found optimal codetable of the format
# "ct-[resultname]-[absolute support]d-0-[##].ct".
#The standard codetable of the format
# "ct-[resultname]-[absolute support]d-0-0.ct
ct <- list.files(resultPath,
      ".*(ct-).*(-0-)[^0].*(.ct)$", recursive = TRUE)
if(length(ct) > 1){
  stop("Error: SLIM should only produce one codetable
  that doesn't end with '0-0.ct',
  please inspect the results.")
}
if(length(ct) == 0){
  stop("ERROR: Found no codetable!")
}

st <- list.files(resultPath,
      ".*(ct-).*(-0-0.ct)$", recursive = TRUE)
```

```
  if(length(st) > 1){
    stop("ERROR: SLIM should only produce one
    standard codetable that ends with 'o-o.ct',
    please inspect the results.")
  }
  if(length(st) == 0){
    stop("ERROR: Found no standard codetable!")
  }

  translate_codetable(resultName, resultPath,
                ct, translation_table, type="ct")
  translate_codetable(resultName, resultPath,
                st, translation_table, type="st")

  cat("-------------------------------------
  All Done!
  -------------------------------------\n")

  # Create SLIM result object.
  codetable <- fread(paste0(resultPath, "ct.csv"))
  patternset <- map(as.list(codetable$Pattern),
                    ~str_split(.x, " ", simplify = TRUE))
  standardtable <- fread(paste0(resultPath, "st.csv"))

  result <- SLIM_result(resultName=resultName,
        minimumSupport=minimumSupport,
        codetable=codetable, patternset=patternset,
        standardtable=standardtable, absMinSup=absMinSup)

  return(result)
}
```

Appendix B

SLIM results from Approach 1

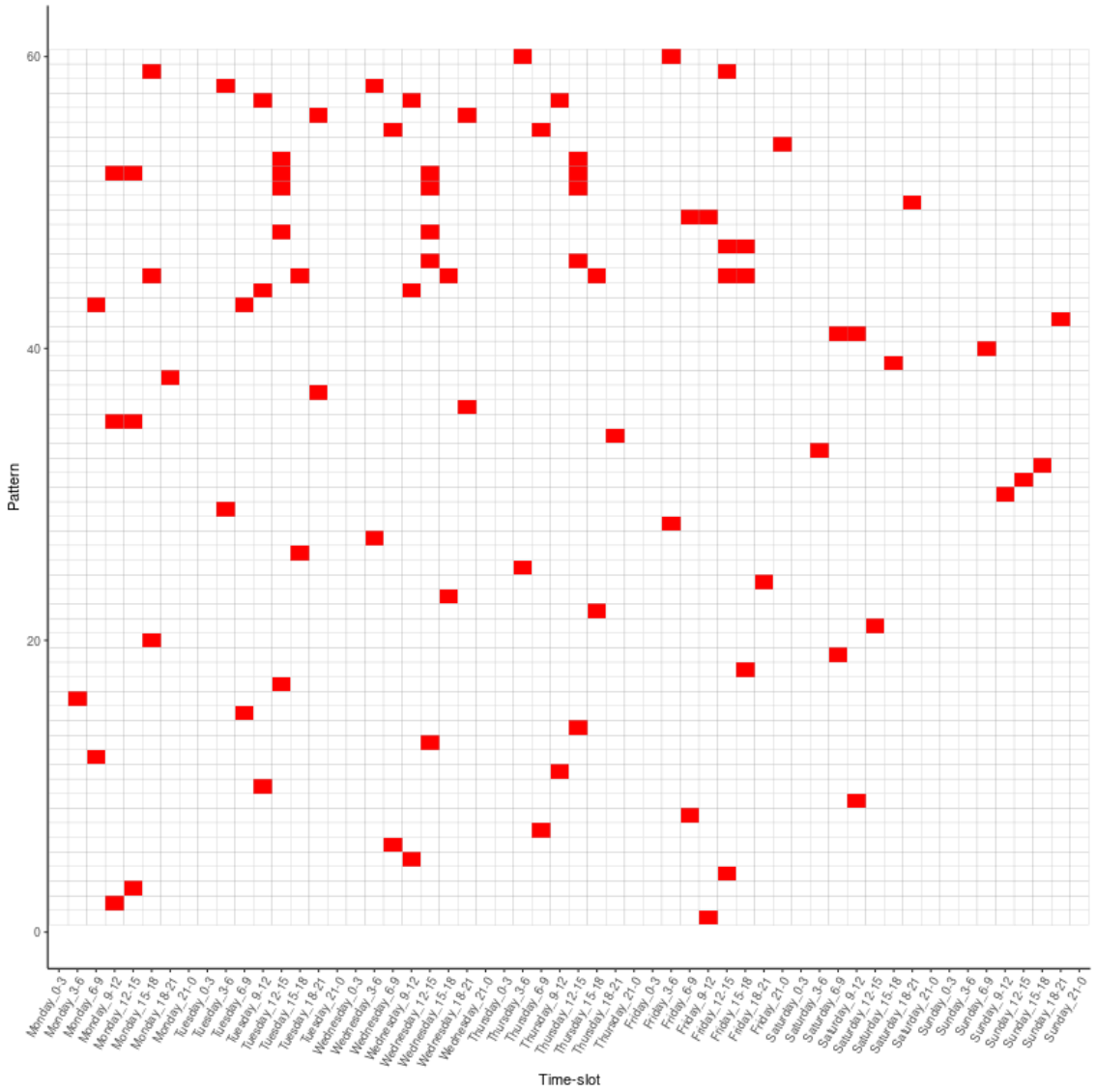


Figure B.1: Top 60 patterns ordered by cover occurrence. Result of running SLIM in the first approach with minimum support 0.01.

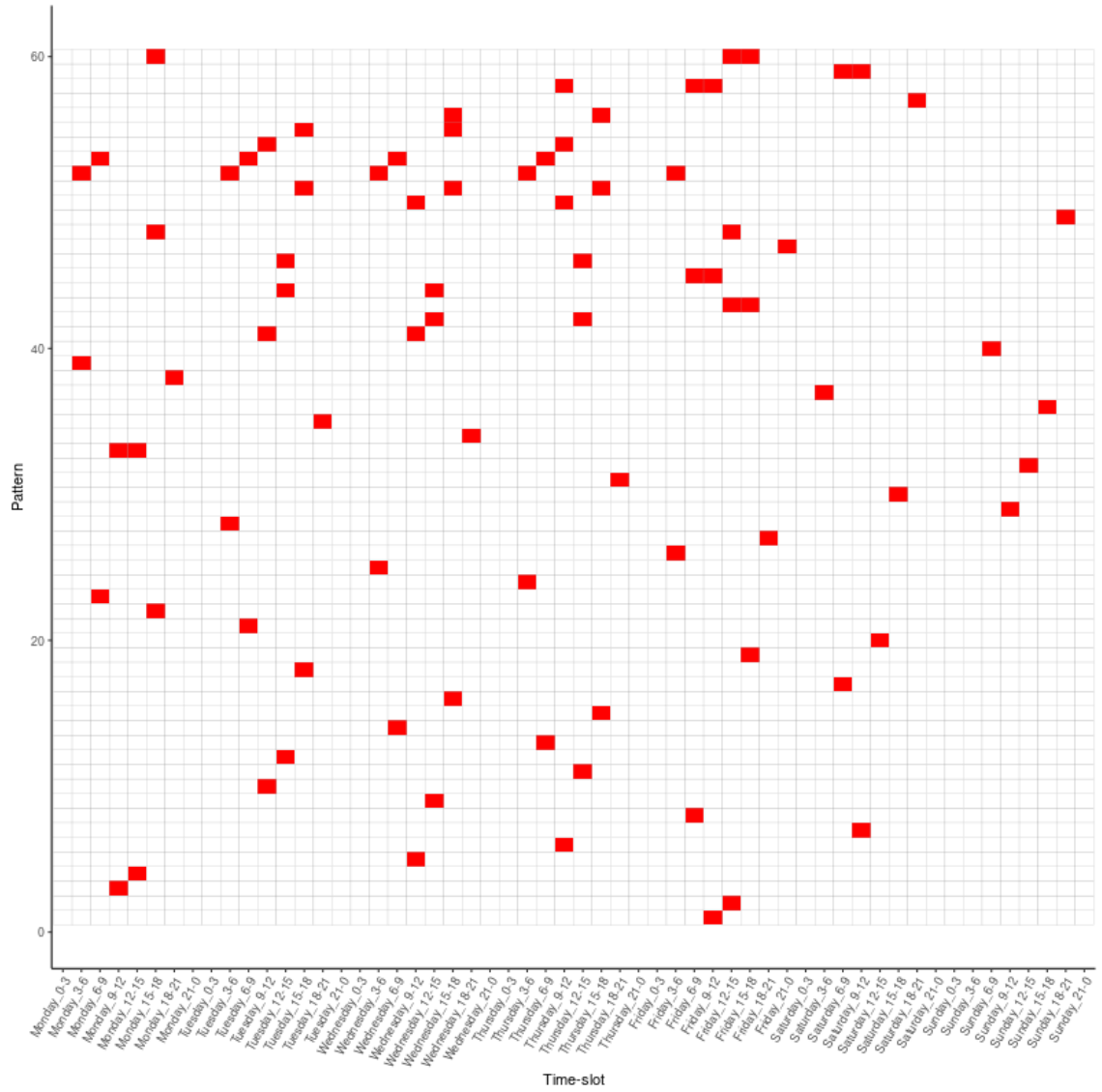


Figure B.2: Top 60 patterns ordered by cover occurrence. Result of running SLIM in the first approach with minimum support 0.005.

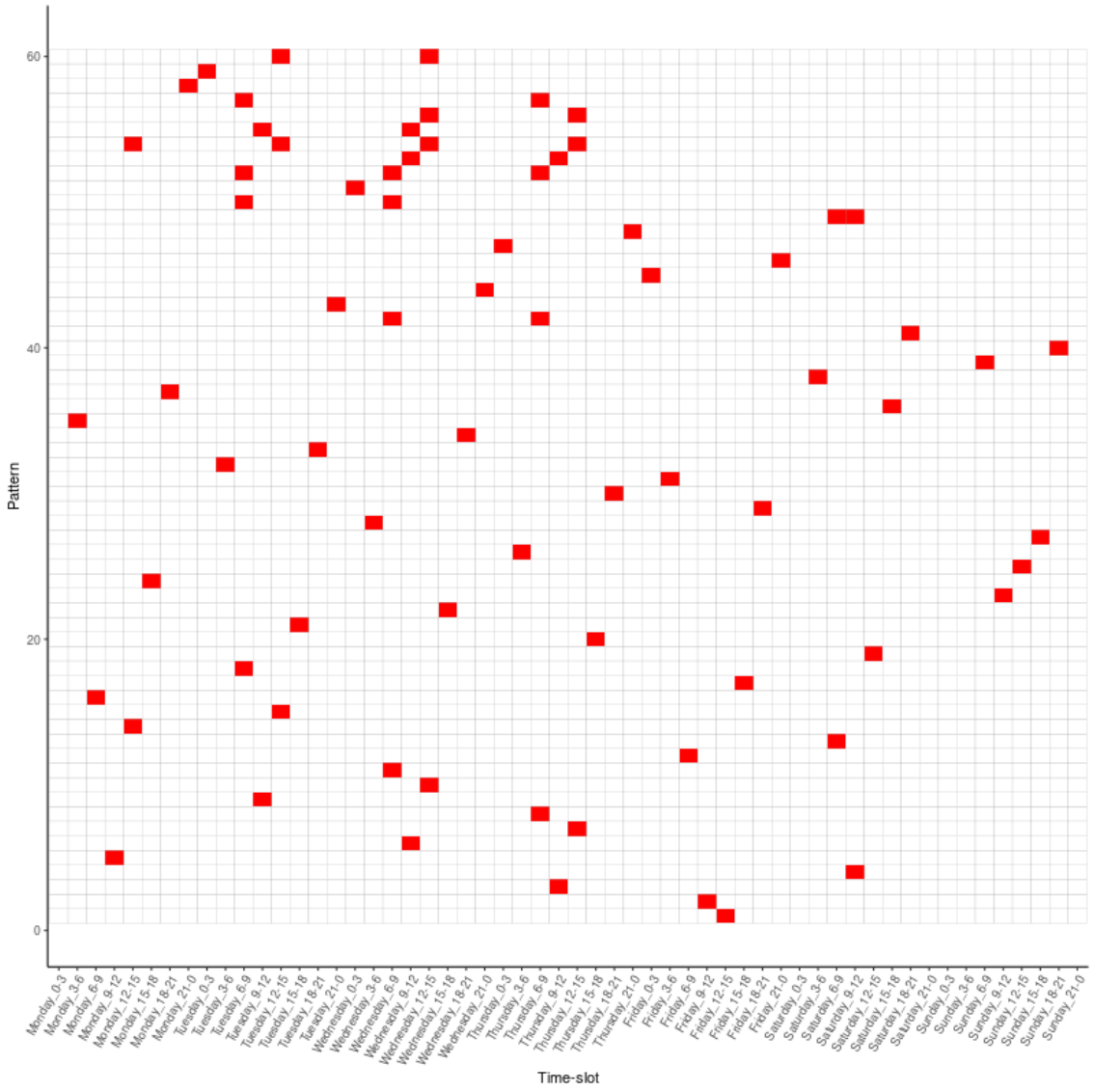


Figure B.3: Top 60 patterns ordered by cover occurrence. Result of running SLIM in the first approach with minimum support 0.001.

Appendix C

Time-slots from Approach 2

time-slot	start-time	end-time
1	Monday:00:00	Monday:02:45
2	Monday:02:45	Monday:04:30
3	Monday:04:30	Monday:06:02
4	Monday:06:02	Monday:07:30
5	Monday:07:30	Monday:08:52
6	Monday:08:52	Monday:10:11
7	Monday:10:11	Monday:11:29
8	Monday:11:29	Monday:12:51
9	Monday:12:51	Monday:14:25
10	Monday:14:25	Monday:16:28
11	Monday:16:28	Monday:20:10
12	Monday:20:10	Tuesday:02:24
13	Tuesday:02:24	Tuesday:04:23
14	Tuesday:04:23	Tuesday:06:08
15	Tuesday:06:08	Tuesday:07:51
16	Tuesday:07:51	Tuesday:09:30
17	Tuesday:09:30	Tuesday:11:07
18	Tuesday:11:07	Tuesday:12:49
19	Tuesday:12:49	Tuesday:14:51
20	Tuesday:14:51	Tuesday:18:00
21	Tuesday:18:00	Wednesday:02:22
22	Wednesday:02:22	Wednesday:04:33
23	Wednesday:04:33	Wednesday:06:26
24	Wednesday:06:26	Wednesday:08:12
25	Wednesday:08:12	Wednesday:09:54
26	Wednesday:09:54	Wednesday:11:32
27	Wednesday:11:32	Wednesday:13:15
28	Wednesday:13:15	Wednesday:15:12
29	Wednesday:15:12	Wednesday:18:13
30	Wednesday:18:13	Thursday:02:00

Table C.1: Continues on the next page.

31	Thursday:02:00	Thursday:03:49
32	Thursday:03:49	Thursday:05:28
33	Thursday:05:28	Thursday:07:06
34	Thursday:07:06	Thursday:08:35
35	Thursday:08:35	Thursday:10:01
36	Thursday:10:01	Thursday:11:24
37	Thursday:11:24	Thursday:12:50
38	Thursday:12:50	Thursday:14:29
39	Thursday:14:29	Thursday:16:33
40	Thursday:16:33	Thursday:20:16
41	Thursday:20:16	Friday:02:28
42	Friday:02:28	Friday:04:27
43	Friday:04:27	Friday:06:09
44	Friday:06:09	Friday:07:46
45	Friday:07:46	Friday:09:18
46	Friday:09:18	Friday:10:47
47	Friday:10:47	Friday:12:17
48	Friday:12:17	Friday:13:57
49	Friday:13:57	Friday:15:59
50	Friday:15:59	Friday:19:18
51	Friday:19:18	Saturday:02:45
52	Saturday:02:45	Saturday:06:08
53	Saturday:06:08	Saturday:08:48
54	Saturday:08:48	Saturday:11:50
55	Saturday:11:50	Saturday:15:48
56	Saturday:15:48	Sunday:04:38
57	Sunday:04:38	Sunday:09:00
58	Sunday:09:00	Sunday:13:10
59	Sunday:13:10	Sunday:16:55
60	Sunday:16:55	Monday:00:00

Table C.2: Exact time-slots found with clustering in the second approach. Start-time is included and end-time is excluded in each time-slot.