“IT IS NOT THE QUANTITY OF MOTIVATION THAT MATTERS”

A STATISTICAL COMPARISON BETWEEN K-MEANS CLUSTER ANALYSIS AND LATENT PROFILE ANALYSIS

Joachim Waterschoot
Student number: 01201353

Supervisor(s): Prof. Dr. Jan De Neve, Prof. Dr. Wim Beyers

A dissertation submitted to Ghent University in partial fulfilment of the requirements for the degree of Master of Science in Statistical Data Analysis

Academic year: 2019 – 2020

GHENT UNIVERSITY

The author and promoter give permission to consult this master dissertation and to copy it or parts of it for personal use. Every other use falls under the restrictions of the copyright, in particular concerning the obligation to mention explicitly the source when using results of this master dissertation.

Gent, September 4, 2020

The promotor,	The author,
Prof. Dr. Jan De Neve	Joachim Waterschoot

Acknowledgements

The writing of this master thesis has involved a process of considering, exploring, discovering, experiencing and pursuing. Three years ago, I *considered* to start the Master in Statistical Data Analysis at Ghent University for the reasons of enhancing my statistical skills and insights, which is also one of the central activities in my PhD project at the Faculty of Psychology and Educational Sciences. By the Master courses, I started to *explore* all statistical possibilities both in a professional way and a personal way. Challenging my own capacities and *discovering* a set of new skills has been a life-changing *experience*. I hope this master thesis could be the cheery on the top, representing my capacities of coding, analyzing, visualizing interpreting and reporting statistical results. Furthermore, this thesis represents a feedback to the main reason why I started the Master course of Statistical Data Analysis, namely to participate the statistical and, thereby, conceptual development of the psychological literature. By focusing on a common-used but highly debated statistical method in the psychological literature, namely cluster analysis, the current report aims to provide a comprehensive view on cluster methods and how to use them.

I would like to compare the process in which this Master thesis has been written with my participation of the 100 km Dodentocht ('Walk of Death' in English) in 2017, an annual walking event in Flanders playing a central role in this master thesis. Notably, this has not been an individual project. Therefore, I would like to mention and express my gratitude to several people.

First of all, I received the chance of working on this specific topic because of my promotor prof. dr. Jan De Neve and my co-promotor prof. dr. Wim Beyers. From the first moment, they were supportive and enthusiastic in working on this project with me. To the last moments, they provided me informative and constructive feedback, this in favor of finishing an interesting journey.

Second, the process during both the Master course and during the writing of this master thesis has been supported by several significant others. First and foremost, Laura, who supported, took care and stimulated me at the most positive and most negative moments, and somewhere in-between. Next, I would like to thank my family, especially my parents Tony and Chris, my brother Mathias and my sister Marlies who encouraged me to participate and to pursue the Master course, my little goddaughter Marte for being a special source of positive energy since September 2019 and my close friends Ewout, Jelle, Mattia and Steven for providing me the sufficient and appropriate distractions and support.

Third, my PhD promotor prof. dr. Maarten Vansteenkiste and co-promotor prof. dr. Bart Soenens for demonstrating the reasons why I was doing this on a daily base. Based on an identified type of motivation, both has been and still are inspiring and

challenging me in my personal, investigatory and scientific development. Never, they mentioned issues in times of studying exams or working on statistical projects, which was a fundamental point of support for me. Together with them, I am grateful having my colleagues around me who were unconditionally supportive at all times.

Last but not least, the organization of the 100 km Dodentocht and all participants of the Dodentocht study. Their joy, enthusiasm and perseverance transformed this study into an experience to never forget.

Table of Contents

List of Tables

List of Figures

Abstract

Recently, a new methodological approach has been emerging in the literature of motivation psychology. By a person-oriented approach, individuals are studied by a combination of variables representing a 'typical feature', rather than studied by a separation of variables. For instance in the literature of the Self-Determination Theory, authors have been using different types of clustering methods to investigate *good*, *bad*, *amotivated*, *high* and *low* motivational profiles. However the literature is growing exponentially, the debate among which method to use is still ongoing as many authors have used different clustering techniques with different results. Therefore, the current statistical report aimed to focus on the two most-commonly used techniques in the literature, namely the K-Means clustering and the Latent Profile Analysis.

Using a sample of 1078 walkers participating the 100 km Dodentocht (59% female; $M_{age} = 43.27$, range: $18 - 77$), we addressed the topics of internal consistency, outliers detection and outliers handling, distribution skewness, descriptives and standardization in Chapter 1. Subsequentely, we discussed the K-Means clustering (Chapter 2) and the Latent Profile Analysis (Chapter 3) procedures. In each chapter, we briefly introduce and explains the theoretical idea behind the clustering method, followed by the appropriate procedures regarding the validation of clustering, the determination of choosing the optimal number of clusters and the study on cluster characteristics. Based on this, we concluded in both K-Mean clustering and Latent Profile analysis three clusters referring to a *Good motivation* profile, a *Bad motivation* profile and a *Low motivation* profile. Also, both clusters show good correspondence, one of the comparison topics that is addressed in Chapter 4. This final chapter contains a comparative discussion of both cluster methods mentioning the strengths and weaknesses and multiple topics characteristing each cluster method. In doing this, we used both an analytical and a conceptual approach, this with the goal to provide a guideline with fundamental considerations.

All this is accompanied by direct results of R codes. Moreover, a tutorial with full R codes and guidelines has been provided in Appendix C. By doing this, we hope to inspire, to guide and to inform future researchers in performing clustering techniques based on knowledge-based and statistical reflections.

Introduction

In the field of motivational psychology, the question *'why people do what they do'* has been a central focus of multiple theories for decades. One major theory dealing with the motivation and psychological development of people is the *Self-Determination Theory* (SDT; Deci & Ryan, 1985; Ryan & Deci, 2017). The theory serves a fundamentally theoretical and empirical base to study people's motivation since the eighties. In contrast to other famous theories of motivation, like Maslow's pyramid (1943) or Skinner's rewarding and punishing (1938), the SDT has a growing theoretical framework with practical applications to multiple domains like sports, parenting, education and even healthcare. This is important in the development of the theory, as its popularity is due to both the top-down (from theory to practice) and bottom-up (from practice to theory) movements. Central to the theory is the distinction between *controlled* and *autonomous* motivation, arguing for more types of motivation than the classic distinction between *extrinsic* and *intrinsic* motivation.

Multiple types of motivation: a Dimensional Approach

From the early 80's, SDT argues for multiple types of extrinsic motivation that differ in the degree to which one's motivation is *internalized* or 'owned' by the person. This means that the more the source of the behavior is 'outside' a person, the more the motivation is externally regulated and, thereby, the less it is internalized. For example, eating an apple for the reason it is healthy or it is tasteful is a subtype of motivation that is more internalized, rather than eating an apple to achieve a parental reward, from which the source of the behavior is outside the person's values.

In total, the theory proposes three types of extrinsic motivation, related to each other in the degree these are internalized. For instance, when someone performs behavior for the reason a reward will be achieved when something is done or for the reason someone else is expecting something is done, the motivation has an *external* regulation. When someone is pressuring him-/herself to do something, the type of motivation knows a stronger ownership by the person. This is called an *introjected* regulation. This is the case when, for instance, a student is learning for school to achieve high grades because he/she will feel guilty / ashamed / disappointed when the achieved grades are not high enough. This type of motivation is mostly related to feelings of high internal pressure and perfectionism (Chang, 2016). Both types can be labelled as *controlled* motivation, because they know the least levels of internalization/ownership.

The last subtype of extrinsic motivation within SDT is called *identified* regulation. When a person has found a match with the (externally-provided) reasons to do

something and think it is important / personally valuable / personally useful to do, the reasons to perform the behavior are 'identified'. For example, a child might eat an apple because he thinks it is healthy to do, he has identified himself with the reasons to eat, rather for the reasons it is obligated by his parents (i.e. external regulation) or to avoid feelings of guilt (i.e. introjected regulation). The most qualitative type of motivation is *intrinsic* motivation, in which the reasons to perform behavior are not internalized because they are within the person in a natural way. The most common outcomes of intrinsic motivation are the feelings of pleasure, interest and joy. People who are intrinsically motivated are enthusiastic to perform the behavior and feel responsible for the choices they make. Both types of motivation can be called *autonomous* motivation, because they know a high level of internalization and personal ownership.

At last, there is one type of motivation that ends up outside the internalization continuum. As this continuum describes different types of one's motivation, *amotivation* describes the type where the motivation is lost. At that point, the person's behavior is not regulated anymore because all reasons to perform the behavior are gone.

As SDT represents an empirical-based theory, several questionnaires has been developed to identify and to quantify one's motivation. By doing this, researchers are able to both measure the *quantity* (i.e. level of motivation) and the *quality* (i.e. type of motivation) of one's motivation. One of the most used questionnaires is developed by Ryan and Connell (1989), measuring all types of motivation. In the following years, the same questionnaire has been successfully used in multiple studies across domains like work (e.g. Deci et al., 2017) and school (e.g. Waterschoot et al., 2019). In 2009, Vansteenkiste, Soenens, Sierens, Luyckx and Lens attempted to study both aspects of motivation in one analysis. Here, they aimed to find groups of people that could be distinguished based on both their level (i.e. quantity) and their type (i.e. quality) of motivation, this by using K-Means cluster analysis. By doing this, the SDT-literature joins the modern person-oriented approach in psychology in which the individual is studied by analyzing multiple variables in a parallel way instead of separately (Bergman & Wangby, 1997). This allows for a perspective on the individual as a 'functioning totality', by which the statistical techniques uncover 'common types' or 'typical patterns' in a population. For instance, an individual is classified to the typical pattern of being 'highly' or 'strongly' motivated, rather than scoring high on both autonomous and controlled motivation.

"the Quality of Motivation matters": A modern Person-Centered Approach

For the first time, Vansteenkiste et al. (2009) applied a person-centered approach where the separate motivational dimensions are combined into clusters or 'profiles' of motivational scores. These profiles should gain the practical insight into combinations of motivation and support the theoretical idea of SDT wherein the quality of motivation matters more than the quantity of motivation. In the upcoming years, more papers started to use this idea, like the study of Haerens et al. (2010). As they provide extended theoretical predictions, five different clusters were found based on the types of *autonomous* (combination of *intrinsic* and *identified* motivation), *controlled*

motivation (combination of *external* and *introject* motivation) and *amotivation*. One cluster contained the combination of high scores on all types of motivation, showing a *high quantity* of motivation, where another cluster included low scores on all types of motivation, named as the *low quantity* profile. In addition, two other clusters showed evidence for the role of the quality of motivation, with one cluster including high scores for controlled motivation and low scores for autonomous motivation (i.e. the *bad quality* cluster) and a group with scores in reverse (i.e. the *good quality* cluster). The fifth clusters contained high scores for *amotivation* (see Figure 1 for an overview of these motivational clusters). In the following years, the person-centered approach of motivational profiles has been used by multiple papers in the domain of sports (e.g. Bechter et al., 2018; Wang et al., 2016), work (e.g., Howard et al., 2016; Moran et al., 2012) and education (e.g. Liu et al., 2014). In general, these papers aimed to find the same number of profiles as been hypothesized by Vansteenkiste et al. (2009).

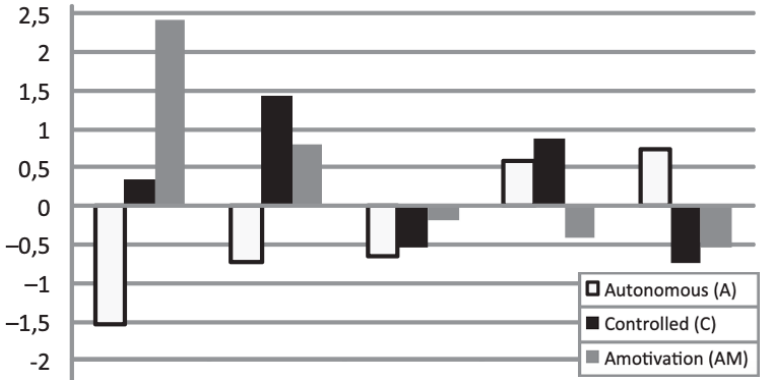


Figure 1: Standardized scores as a function of motivational profiles in SDT (Haerens et al. 2010)

However, it can be observed that the increasing number of both scientific publications and conference presentations and posters including cluster analysis showed fundamental differences in the type of cluster analysis, one part using K-Means Cluster analyses (MacQueen, 1957) and one part using Latent Profile Analysis (Lazarsfeld & Henry, 1968). As the literature of SDT is still increasing with 36173 publications in 2018, 39872 in 2019 (+ 9.3 %) and already 22768 publications in 2020 (Google Scholar, 16/08/2020), the aim of the current report is to provide a statistical-based overview of the two most popular cluster analyses: K-Means cluster analysis (Chapter 2) and Latent Profile Analysis (Chapter 3). Based on the literature, we would expect to find 5 motivational profiles, equivalent to the results of Haerens et al. (2010). By doing this, we hope to demonstrate important differences and considerations within each analysis, this to raises analytical consciousness for all future researchers in social sciences. To encourage this, a tutorial is written using R-coding and the most important considerations regarding cluster analysis (see Appendix C).

Chapter 1

The `dotoset` dataset

In preparation of the following chapters that will focus on the K-Means Cluster analysis (Chapter 2) and Latent Profile Analysis (Chapter 3), Chapter 1 briefly represents the data collection procedure, the descriptive statistics and the outlier detection and the standardization procedures of a dataset. This is important, as it is essential for each researcher to known the content and the structure of the dataset. Also, it is important for accurate cluster analysis to consider the reliability, outliers and scales of the study variables. Therefore, we discuss the most fundamental steps to follow in preparation of cluster analysis. This will be done on the `dotoset` dataset as part of a large-scale study that took place in August, 2019 during the '100 km Dodentocht' (100 km Walk of Death in English) in Flanders, Belgium.

1.1 Methods

1.1.1 Participants and Procedure

Participants were registered walkers for a long-distance walking event in Flanders, Belgium, named the '`dodentocht`' (in Dutch, referring to the name of the `dotoset` dataset). It is an annual event in which 13 000 walkers (as a limit) have to finish 100 kilometers within 24 hours after the start. The event is located around the village Bornem in Flanders and receives an increase of international attention each year. During the 50th edition of the event in 2019, 12 675 walkers were registered for the event, from which 1078 (8.5%) participated the current study (59% female; M_{age} = 43.27, range: 18 – 77). The study was promoted both digitally by email after registration and verbally at several 'registration days' preceding the walking tour. As walkers needed some material (e.g. chip, t-shirt, etc.) before starting their walking tour, these days provided us as researchers the opportunity to promote the research verbally. When one was interested to participate the study, it was asked to complete a short digital form including their mail address. It was emphasized that these contact data only would be used for study-related goals. Three days before the start of the walking tour, all interested walkers (n = 1620) received an email including an online questionnaire - made in Qualtrics - regarding their motivation to participate the walking event. The questionnaire was available both in Dutch and

English. On average, it took approximately 9 minutes to complete. At the start of the walking event, on August the 9th at 9 pm, the questionnaire was closed. The study was in line with all ethical protocols of the Ghent University.

This data collection was part of a larger study project, named 'the Dodentocht-study', in which 1078 walkers were followed before, during and after the walking tour. More information about the full study and future plans can be found on the website www.ugent.be/dodentocht.

1.1.2 Measurements

The data that will be used in the current report is derived from the first questionnaire, conducted between three and one days before the start of the walking event. All items had to be rated on a Likert scale going from 1 (*"Totally disagree"*) to 5 (*"Totally agree"*). In this online questionnaire, participants were asked to report their name, age and birth date (as second check) in case they agreed the informed consent. It was emphasized that this data would be handled confidentially and would be used to combine all conducted questionnaires within participants. As demographical information, we asked participants' age and gender (1 = boy, 2 = girl). As we will focus on the idea and techniques of cluster analysis, these background variables will not be included in further analyses of this report.

Motivation

Motivation was assessed in regard to a particular personal goal that participants had described qualitatively.[1] For this goal, we measured people's motivation using the *Self-Regulatory Style Questionnaire* (SRQ; e.g. Assor, Vansteenkiste, & Kaplan, 2009). The questionnaire contained 18 items assessing 3 types of motivation: autonomous (16 items, e.g. *"because I think it is joyful to pursue this goal."*; $\alpha = .84$, see table 1.1), controlled (16 items, e.g. *"because I would feel like a failure when I do not pursue this goal"*; $\alpha = .91$, see table 1.2) and amotivation (4 items, e.g. *"I wonder why I would pursue this goal"*; $\alpha = .85$, see table 1.3). The number of items differ between types, because all subtypes of motivational regulation were conducted in the original questionnaire. More specifically, items for intrinsic and identified regulation were merged for autonomous motivation and introjected and external regulation for controlled motivation.

Before calculating the scale scores for each variable, the internal consistency was tested by the `sjt.itemanalysis` function of the `sjPlot` package (see table 1.1-1.3 for output). By doing this, we become scores for *Cronbach's alpha* (i.e. indicator for the internal consistency with acceptable scores higher than .70), *mean inter-item*

[1] Using qualitative analysis, these goals were coded into subcategories according to the Approach-Avoidance Achievement-Goal theory (Elliot, Murayama & Pekrun, 2011), Social goals (Allen, 2003) and SDT (Ryan & Deci, 2017). The first results of Waterschoot et al. (in preparation) did not show any significant differences between goal types in terms of motivation scores. Examples of the described goals were 'finishing', 'proving his/herself', 'honoring a close person' and 'enjoying the experience'

correlations (i.e. alternative indicator for the internal consistency with acceptable scores between .20 and .40), *item difficulty* (i.e. indicator for the difficulty of items with acceptable scores between .20 and .80) and *item discrimination* (i.e. indicator for how well items can detect differences between persons with acceptable scores from .20. The closer to 1, the better). In total, the questionnaire had high acceptable scores for the internal consistency with Cronbach's alphas between .84 and .91 and mean inter-item correlations between .26 and .59. Further, values for item difficulty range between .37 and .90 and item discriminations are higher than at least .37. Though, the mean inter-item correlation for amotivation might be higher than the acceptable value of .40. Also, the values for item difficulty for controlled motivation might be high as they reach .90, indicating that some items were too comparable towards each other, especially with regard to items measuring the motivation to achieve a reward. Based on all items for each variable, the mean score was calculated to become the variables *autonomous motivation*, *controlled motivation* and *amotivation.* If more than 80% of the items are missing, the scale score was defined as a missing value (NA), resulting in 30 NA's (2.8%) for autonomous motivation, 26 NA's (2.4%) for controlled motivation and 9 for amotivation (0.8%). These values were imputed using 'predictive mean matching', after testing the Little's MCAR test showing that NA's are Completely Missing at Random ($\chi^2 = 3.44, p = .08$).

Table 1.1: Item Analysis autonomous motivation

Row	Missings	Mean	SD	Skew	Item Difficulty	Item Discrimination	Alpha if deleted
mot1_1	0.83 %	4.49	0.69	-1.63	0.9	0.347	0.839
mot10_1	1.29 %	4.23	0.74	-1.21	0.85	0.361	0.839
mot1_2	5.24 %	4.39	0.74	-1.53	0.88	0.485	0.833
mot10_2	5.98 %	4.18	0.85	-1.3	0.84	0.43	0.836
mot2_1	1.38 %	4.31	0.79	-1.2	0.86	0.492	0.833
mot11_1	1.20 %	3.98	0.9	-0.92	0.8	0.525	0.83
mot2_2	5.52 %	4.16	0.89	-1.15	0.83	0.614	0.826
mot11_2	5.98 %	3.98	0.97	-1	0.8	0.632	0.824
mot3_1	0.83 %	4.18	0.82	-0.93	0.84	0.428	0.836
mot12_1	1.66 %	3.78	0.98	-0.7	0.76	0.434	0.836
mot3_2	5.52 %	4.19	0.86	-1.11	0.84	0.566	0.829
mot12_2	6.07 %	3.93	1.02	-0.95	0.79	0.496	0.832
mot4_1	1.20 %	4.39	0.83	-1.56	0.88	0.37	0.839
mot13_1	1.20 %	3.41	1.12	-0.48	0.68	0.366	0.841
mot4_2	5.52 %	4.04	1.02	-1.18	0.81	0.495	0.832
mot13_2	5.89 %	3.43	1.21	-0.53	0.69	0.379	0.841
Mean inter-item-correlation=0.258 · Cronbach's Alpha=0.843							

1.2 Results

In this section, we briefly overview the descriptive statistics, possible outliers, considerations regarding data transformations and standardization of the three motivation variables *autonomous motivation*, *controlled motivation* and *amotivation* as been measured in the previous section.

Table 1.2: Item Analysis controlled motivation

Row	Missings	Mean	SD	Skew	Item Difficulty	Item Discrimination	Alpha if deleted
mot5_1	0.92 %	3.27	1.2	-0.19	0.65	0.48	0.912
mot14_1	1.56 %	2.96	1.25	-0.06	0.59	0.56	0.91
mot5_2	5.89 %	2.96	1.25	0.01	0.59	0.549	0.91
mot14_2	5.89 %	2.91	1.27	-0.03	0.58	0.605	0.908
mot6_1	1.01 %	2.72	1.29	0.24	0.54	0.572	0.91
mot15_1	1.66 %	3.05	1.27	-0.15	0.61	0.612	0.908
mot6_2	5.80 %	2.47	1.28	0.53	0.49	0.679	0.906
mot15_2	5.89 %	3.02	1.33	-0.08	0.6	0.625	0.908
mot7_1	0.92 %	1.9	0.96	0.97	0.38	0.637	0.908
mot16_1	1.56 %	2.35	1.14	0.42	0.47	0.593	0.909
mot7_2	5.98 %	1.96	0.98	0.94	0.39	0.688	0.906
mot16_2	6.35 %	2.22	1.14	0.67	0.44	0.641	0.907
mot8_1	1.01 %	2	1.04	0.89	0.4	0.543	0.91
mot17_1	1.38 %	1.84	0.92	1.03	0.37	0.66	0.907
mot8_2	6.07 %	1.99	1.05	0.97	0.4	0.64	0.907
mot17_2	5.98 %	1.85	0.93	1.07	0.37	0.66	0.907
Mean inter-item-correlation=0.409 · Cronbach's alpha=0.914							

Table 1.3: Item Analysis Amotivation

Row	Missings	Mean	SD	Skew	Item Difficulty	Item Discrimination	Alpha if deleted
mot9_1	1.20 %	1.99	1.08	0.92	0.4	0.626	0.841
mot18_1	1.56 %	2.24	1.14	0.66	0.45	0.713	0.806
mot9_2	6.07 %	2.04	1.06	0.92	0.41	0.692	0.815
mot18_2	6.72 %	2.16	1.14	0.76	0.43	0.75	0.789
Mean inter-item-correlation=0.592 · Cronbach's Alpha =0.853							

1.2.1 Outliers

In the literature of cluster analysis, the detection of outliers can be defined in two ways: (1) these are extreme values in the dataset or (2) these are values that are detected during the cluster procedures by specific algorithms like Outlier Removal Clustering (ORC; Hautamäki et al., 2005) or Density-Based Spatial Clustering of Applications with Noise (DBSCAN; Sander et al., 1998). In the current report, we will discuss the first option for two reasons. First, we want to provide the most appropriate way in general data analysis how to handle outliers, out of the scope of specifically cluster analysis. By doing this, we want to emphasize how strongly this topic is underestimated, especially in the field of psychology. Secondly, the second option would be too narrow information as it contains the explanation of specific types of algorithms from which we believe the definition of cluster analysis as such is too unfamiliar at this point in the report.

Detecting outliers

With its underestimated importance, outliers are a required concern regarding data analysis. Outliers, also called 'extreme values' or 'outlying data', are mostly not even handled in scientific papers, especially in the field of social sciences (Leys et al., 2019). There, listwise deletion seems to be the most applied tool to handle outliers, while it results in a great loss of information. In the other case, when outliers are mentioned, they are mostly defined as values that are higher than 3 standard deviations from the mean. Though, this is not a robust technique as the calculation of the mean itself is affected by outliers (Cousineau & Chartier, 2010; Miller, 1991). Based on

the procedure of Leys et al. (2019), we advice to use the *Median Absolute Deviation* (MAD; Hampel, 1974) to detect univariate outliers. In R, this can be done using the `Routliers` package (Leys et al., 2013). Instead of the mean, MAD uses the median as central tendency, which is more robust and less sensitive to the presence of outliers in the dataset. Therefore, a more accurate threshold can be used to define outliers. After calculating the median, each absolute difference between a data point and the median is calculated. Next, the median of these absolute differences is calculated, which corresponds the MAD. In practice, a constant 1.4826 is used to make the MAD and SD of normal distributions comparable. According to Miller (1991), a MAD threshold of 3 is conservative, which we will use in the current report. The lower the threshold, the less conservative the MAD.

Results are visualized in figure 1.1-1.3. The variable `auto_mot` (i.e. autonomous motivation) has 8 extremely low values (figure 1.1), the variable `control_mot` (i.e. controlled motivation) has 4 extremely high values (figure 1.2) and the variable `amot` (i.e. amotivation) has no extremely high values (figure 1.3). In the next step, we calculate the Mahalanobis distance (Mahalanobis, 1930) as indicator for bivariate outliers. Here, the idea is that a data 'cloud' is constructed based on the two variables, by which data points are detected as extreme when they are 'too far' from the centroid of the cloud. For the same reason of univariate outliers, we will use the median of the cloud as a less robust indicator. Therefore, the procedure is performed in pairs of two variables showing 9 outliers for autonomous and controlled motivation, 19 outliers for amotivation and autonomous motivation and 17 outliers for amotivation and controlled motivation (see Appendix A.1 for plots). To be informative about the results with or without the outliers, the `Routliers` package calculates two regression lines. This can be used to compare the differences in associations between the variables including and excluding the detected outliers showing small changes with a higher negative slope between amotivation and autonomous motivation.

Handling outliers

From the moment we have detected outliers, the handling of these data points should be considered extensively (Leys et al., 2013). At first sight, we are not dealing with 'error outliers' from which the base is unknown (e.g. typo's). All values are within the range of 1 to 5. According to Backer and Wicherts (2014), it is argued to keep outliers in the case they do still belong to the distribution of interest. Removing outliers is assumed to be a dramatic decision as a lot of information will be lost. Moreover, it might lead to error estimation of the parameters when these extreme values belong to the distribution of interest. However, in the light of the upcoming cluster analyses, previous literature supports the decision to remove outliers before starting the cluster procedure (e.g., Hautamäki et al., 2005). As the estimation of the cluster mean has an important role in K-Means cluster analysis (Chapter 2), the procedures are not quite robust and highly sensitive to outliers by which the results could be different when outliers are still included. Actually, the COR, ORC or DBSCAN algorithms removes the outliers at the end.

Still, the importance of considering outliers cannot be emphasized sufficiently. To demonstrate, we subset the dataset into one with (`df`, $n = 1078$; including 2.96% outliers) and without (`dfwithout`; $n = 1046$) outliers in favor of making a comparison in the further cluster analyses (see Chapter 4, Knowing the data before clustering).

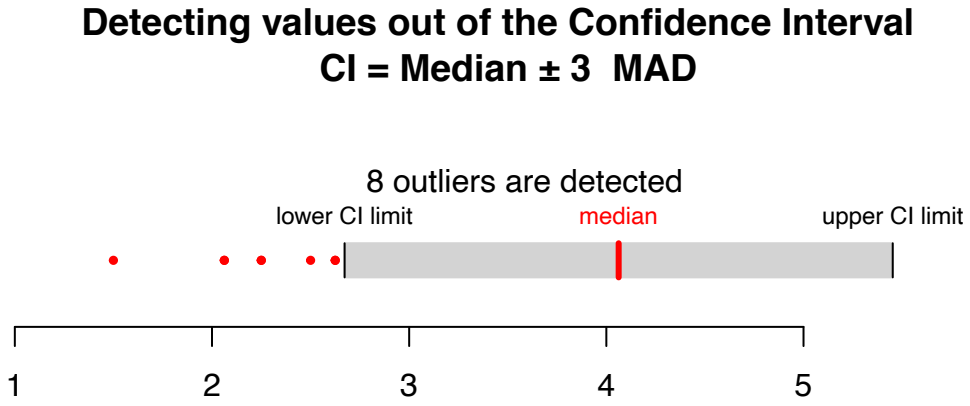


Figure 1.1: univariate outliers in auto_mot

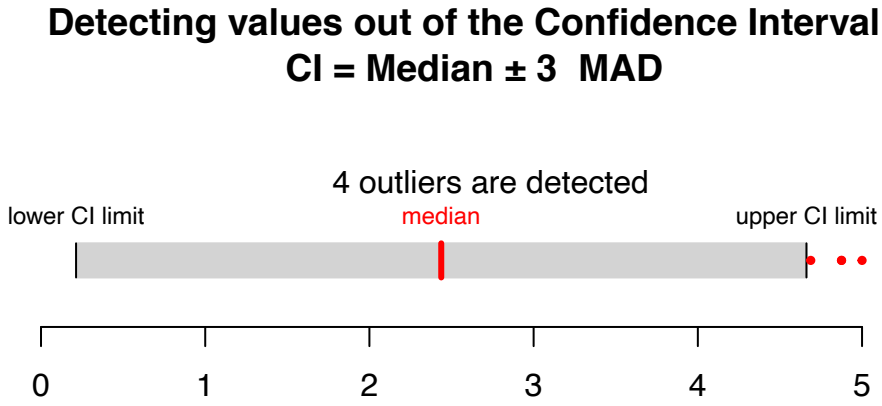


Figure 1.2: univariate outliers in control_mot

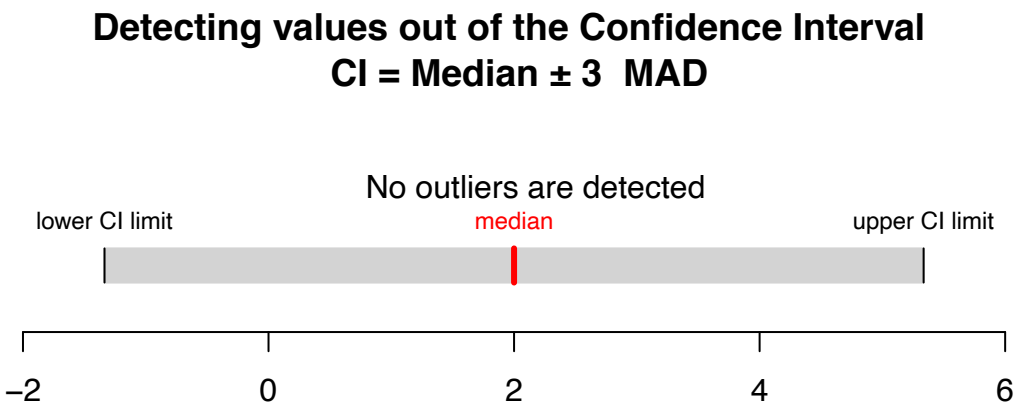


Figure 1.3: univariate outliers in amot

1.2.2 Descriptives

As can be observed in table 1.4, participants were more autonomously motivated compared to the mean scores for controlled motivation and amotivation. Both the scores for skewness (within the range of -1 and +1), and the distribution plots in figures 1.4-1.6 indicate no outspoken skewness in the distributions (Hair et al., 2017), except for `amotivation` (figure 1.6). Both the scores for kurtosis (within the range of -1 and +1) and the distribution plots in figures 1.4-1.6 indicate no outspoken peaked distributions (Hair et al., 2017). In more details, the red dashed line in figures 1.4-1.6 shows the normal distribution of the variables calculated on their mean and the standard deviation. The blue density curve shows the raw data. Noticeably, the score distribution for amotivation show a highly positive skewness score with a skewed distribution to the right.

Table 1.4: Summary dotoset

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
auto_mot	1	1046	4.06	0.51	4.06	4.08	0.48	1.5	5.00	3.50	-0.47	0.79	0.02
control_mot	2	1046	2.45	0.75	2.44	2.43	0.74	1.0	4.88	3.88	0.30	-0.21	0.02
amot	3	1046	2.10	0.92	2.00	2.01	1.11	1.0	5.00	4.00	0.69	-0.14	0.03

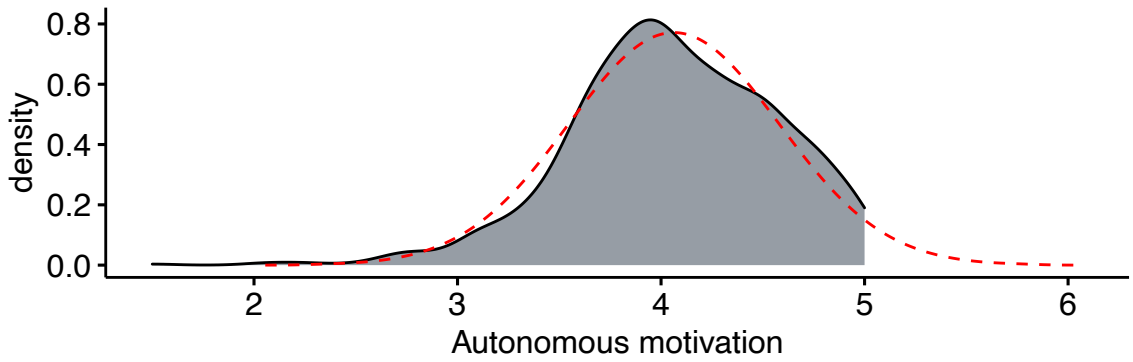


Figure 1.4: Density curve for autonomous motivation

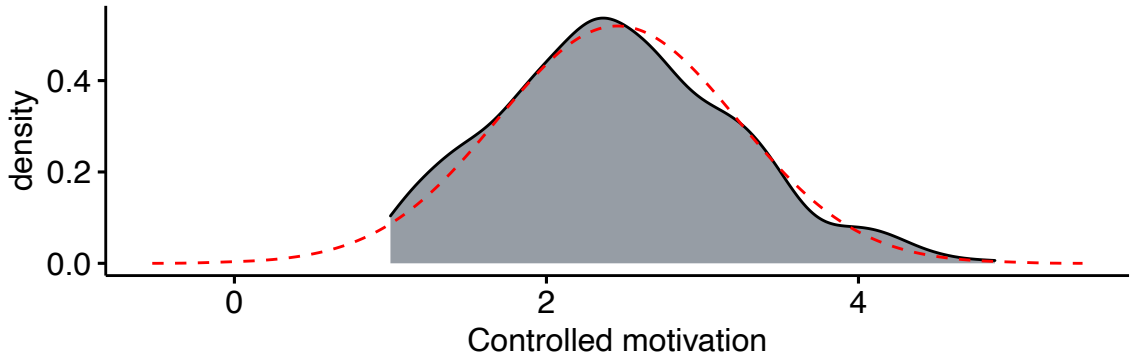


Figure 1.5: Density curve for controlled motivation

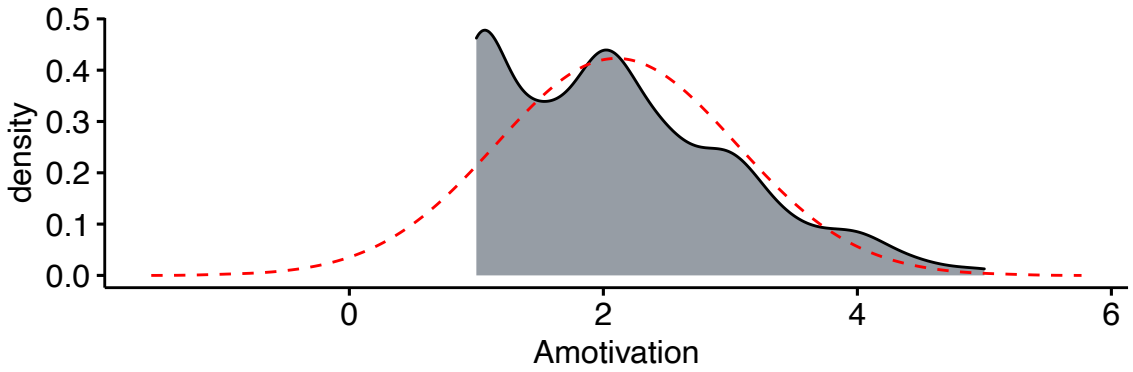


Figure 1.6: Density curve for amotivation

Data transformation

Based on the scores for skewness (table 1.4) and the distribution in figure 1.6, we notice a positive skewed distribution for `amotivation`, indicating a 'floor effect' as many data points seem to 'glued' to the y axis. In the current report, we aimed to emphasize the point of data transformation, as it is unclear how to handle skewed variables in the scientific literature. Some researchers transform their data immediately, this with the idea of satisfying statistic assumptions. However, this is not always required. For instance, regression modelling indicate model assumptions with normalized residuals rather than normal distributed variables, especially when the sample size is small. Also, the skewness of variables can be informative as it is contains information about the study variables. For instance in the current study, we could conclude that participants were low to extremely low amotivated to participate the walking tour. In the literature, some authors argue for data transformation when distributions are skewed. For example, the skewness of the data might result in misleading and inaccurate results, while other argue for the robustness of multivariate clustering to skewness.

In case of data ***transformation***, some authors argue it could affect the (clustering) results, especially in Gaussian mixture modelling (see Chapter 3) which depends on the within-class normality assumption (e.g. Bauer & Curran, 2003; Schork & Schork, 1988). For instance, latent and spurious classes can be found, only to accommodate the heavy tails of non-normal distributions rather than substantively meaningful latent subpopulations. In the literature, several new R packages (e.g. `teigen`, `EMMIXskew`) and clustering techniques (e.g. Bayesian approach; Lin et al. 2007) provided the capacity for skewed-distribution clustering. Hence, this is out of the scope of the current report where we aim to address general cluster techniques and their comparison. One way to check data transformations to the skewed variable is by using the `bestNormalize` package in R. This package estimates normality statistics across a list of transformations (e.g. Box-Cox, square root). It proposes the Ordered Quantile technique as the best transformation giving more weight to the values within the bell curve (see Peterson, 2019 for more information).

Although the reasoning in favor of transforming the data could make sense, others argue for ***no transformation***. For instance, the skewness of variables is seen as a

confirmation of applying mixture modelling (Muthén, 2003). The finding of having multiple maximum densities in the same variable supports the idea multiple classes are within the same variables (see figure 3.1 in section Gaussian Mixture Modelling as an example). In the current report, the floor effect of `amotivation` clearly provides important content-wisely information. In contrast to others, Asparouhov and Muthén (2015) argue for the strong usefulness of skewed variables in mixture modelling. The presence of strongly skewed variables prevents the result of having more classes, only compensating for some deviation of the normal distribution.

Based on this consideration of `amot`, we do not decide to apply transformation.

Data content

Figure 1.8 provide more insight in the associations between the motivation variables by scatterplots and Pearson correlations, showing significant positive correlations at the significance levels of .05 both between autonomous motivation and controlled motivation and between controlled motivation and amotivation. A significant negative correlation is found between autonomous motivation and motivation. These findings are in line with the literature, such that people who are autonomously motivated are less amotivated.

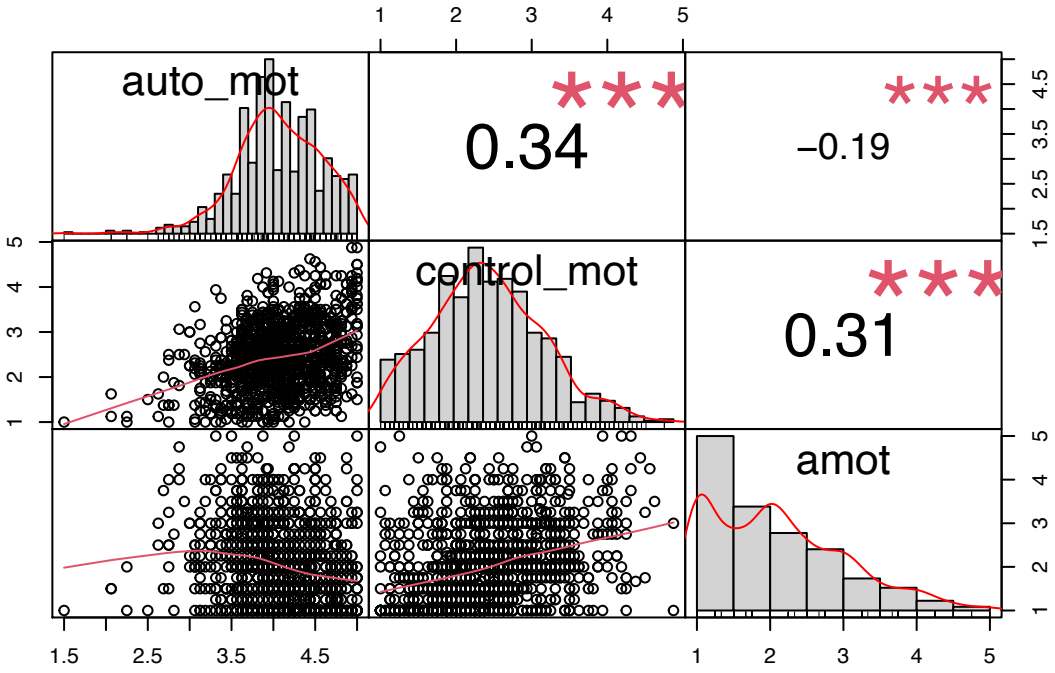


Figure 1.7: Correlations, distributions and scatterplot motivation variables in dotoset

1.2.3 Standardizing

In the final step of data preparation, we standardize all study variables that will be included in the cluster analysis. This is important, as the amount of 'similarity' be-

tween data points within cluster analyses are related to the scale of the variables in cluster analysis. When variables have different scales, they will be treated differently, while we want them to be equally important. The goal here is to make them comparable by using the formula $(x_i - mean(x))/sd(x)$ with x_i referring to the variable score of case i. As a result, we observe all means equaling 0 with standard deviations of 1 in the dataset summary. By comparing the heatmaps of the dataset in figure 1.9 and figure 2.0, the usefulness of scaling observations can be shown as they are more comparable towards each other. In these figures, color codes are used to map the variables' density curve with red having a higher density and blue pointing to the lowest density values. As noticed before, the variable `amot` has two peaks by its positive skewness.

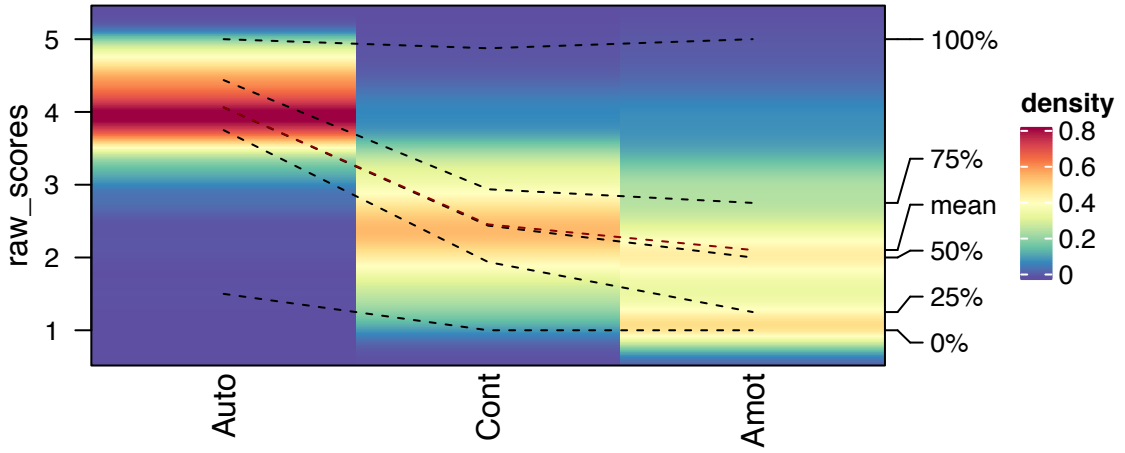


Figure 1.8: Heatmap of the dataset with raw scores

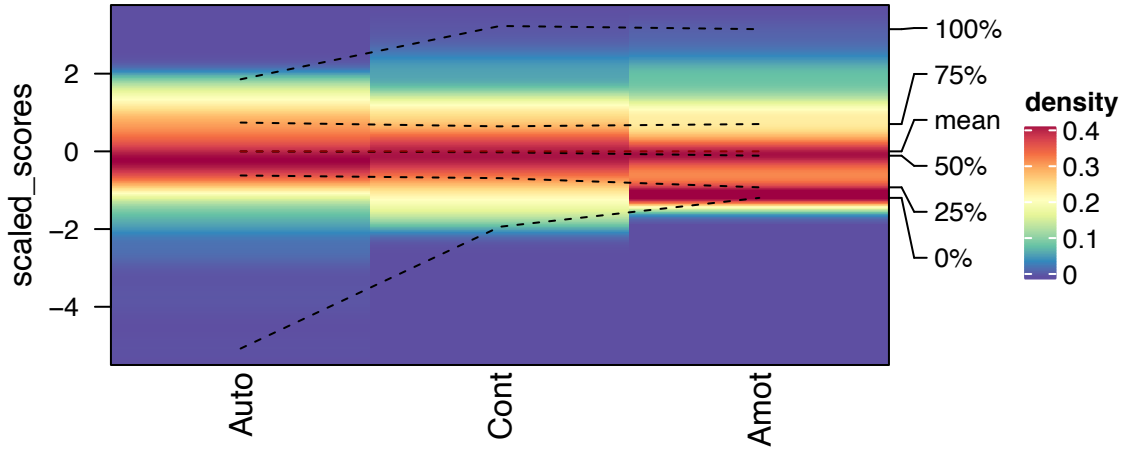


Figure 1.9: Heatmap of the dataset after standardization

Chapter 2

K-Means Cluster Analysis

Cluster analysis refers to a collection of methods to uncover hidden, unknown groups within a dataset. In doing this, it is the goal to partitionate the dataset in the most optimal way by maximizing both the similarity of cases *within* clusters (i.e. increasing the *compactness* or *internal homogeneity* of clusters) and the dissimilarity of cases *between* clusters. In the most optimal way, their external homogeneity is increased by which there is no overlap beween clusters (e.g., Growe, 1999; Hastie et al., 2009).

Partitioning the dataset represents the core idea of cluster analysis. For instance in Figure 2.1, the data is partitioned into two clusters. This figure helps us to think about cluster analysis in a graphical way. These clusters are constructed based on their mean (the black crosses). A perpendicular line (in case of bidimensionality) or a hyperplane (in case of multidimensionality), that passes the middle point of a line connecting the two means, divides the space into two separate subspaces. Using the mean of each cluster is known as the *K-Means* clustering procedure.

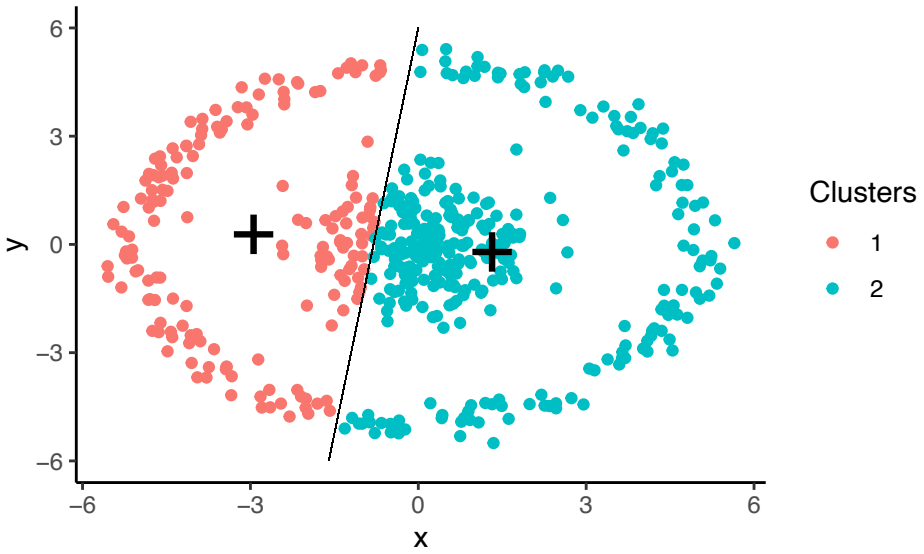


Figure 2.1: Visualisation K-Mean clustering idea

In this report, we will mainly focus on the *K-means clustering* (MacQueen, 1967) as it is the most famous method within cluster analysis. Herein, the number of k clusters are represented by a certain prototype, like the *geometric center* or, in other words, the *mean* (i.e. *the centroid*) of each cluster. The procedure is popular because it is both efficient and effective. Specifically, it produces the fewest possible number of clusters while still maximizing their meaningfulness and statistical importance. Applying this idea on the measurements of motivation, it simplifies the complexity of the continuous scores of different measurements of motivation into a small set of clusters, or as MacQueen quoted:

> "The main use of K-means clustering is to be more of a way for researchers to gain qualitative and quantitative insight into large multivariate datasets than a way to find a unique and definitive grouping for the data." $-MacQueen, 1967$

This quote of MacQueen endorses the initial idea of the current report, such that we can apply cluster analysis to motivational measurements in favor of making interpretations in terms of both the quality and the quantity of motivation. This is valuable, as the position of a researcher towards cluster analysis is double-sided. As we will see in the upcoming K-means cluster analyses, a theoretical angle at the start of cluster analysis can be useful on the one side, while a more exploratory approach can be interesting from an analytical point of view on the other side (see Chapter 4, section Combine knowledge and data).

In practice, there are three commonly used partitioning clustering techniques:

1. **K-means clustering**: a non-robust technique in which each cluster is represented by a centroid.

2. **K-mediod clustering** or **PAM** (Partitioning Around Mediods): a more robust technique as clusters are based on the median rather than the mean of the cluster.

3. **CLARA algorithm** (Clustering Large Applications): a technique that is useful for large data sets.

In the following sections, we will discuss and demonstrate the *K-means clustering* for the reasons we handled outliers in section 1.2.1 and the size of the `dotoset` is moderately large ($N < 2000$).

2.1 (Dis)similarity measurement

A first step is to compute the distance or (dis)similarity between each pair of observations. As we have multiple data points, the result of this computation is generated into a dissimilarity matrix or '*distance matrix*'. In this matrix, the values represent the distance between data points, while the values on the diagonal represent the distance between data points and themselves (which is zero). This matrix is critical, as

it defines the similarity or *proximity* of cases and, thereby, the shape of the clusters. Here, we want to have elements within clusters as close as possible to other elements in the same cluster space. In other words, we try to minimize the sum of variances within clusters $E = \Sigma_{i=1}^{k}\Sigma_{i=1}^{n_i}||x_{ij} - c_i||^2$. In this formula, n_i is the number of cases included in cluster k and $\Sigma_{i=1}^{k} n_i = n$.

The most common measure for the 'cluster compactness' is the *Euclidean distance* (see formula 3.4 with c as the cluster center, x the case it is compared to, i the dimension of x (or c) being compared and k as the total number of dimensions). In other words, it represents the length of a 'distance line' between two points in the Euclidean space. Such Euclidean space is a geometric dimension n. For instance, a plane is an Euclidean 2-space.

$$dE = \sqrt{\Sigma_{k}^{i}(c_i - x_i)^2} \tag{2.1}$$

Alternatives for the Euclidean distance are the *Squared Euclidean distance*, *the Manhattan distance* (i.e. the absolute squared Euclidean distance) or *the Maximum distance between attributed vectors*. These latters are more applicable when the study variables do not approach a normal distribution and significant differences from normality are found. For instance, when we would ignore the skewness of `amot` in section Descriptives. Also, these distance measurements are more robust than the Euclidean distance. For instance, when we would ignore the topic of outliers in section Outliers. Moreover, the similarity between the Euclidean distance and the other alternatives become strong when we standardize it. Because, by standardization (see section Standardizing), there is a relationship between the Pearson correlation coefficient $r(x, y)$ and the (standardized) Euclidean distance $d_{euc}(x, y) = \sqrt{2m[1 - r(x, y)]}$ with m referring to a standardized vector with zero mean and a standard deviation equal to 1. Alternatively, the researcher can choose to use correlation-based distance measurements, like the *Pearson* or *Kendall correlation distance*. This is mostly used in gene expression data analysis. Here, the idea is that the similarity of cases is measured by how they are correlated, even though the values may be far apart in terms of Euclidean distance.

By following R code, the distance matrix is generated for the current *continuous* study variables. The first 3 cases are shown in the output. However this is out of the scope of the current report, the function `get_dist()` of the `factoexra` package also can be used for the alternative distance measurements as mentioned before and the `daisy()` function can be used in the case of *categorical* variables.

```
library(factoextra)
dismatrix <- factoextra::get_dist(dfwithout.sc, method= "euclidean")
round(as.matrix(dismatrix)[1:3, 1:3], 1)
```

```
  1   2   3
1 0.0 2.2 1.4
2 2.2 0.0 0.7
3 1.4 0.7 0.0
```

We can also visualize the distance matrix by a color code where *red* stands for high similarity and *blue* stands for high dissimilarity. For the sake of clarity, the following distance matrix is shown for a random 4% of the cases. It can be useful as it provides a first idea about the presence of clusters in the dataset. The red diagonal stands for the similarity between cases and themselves (maximum similarity).

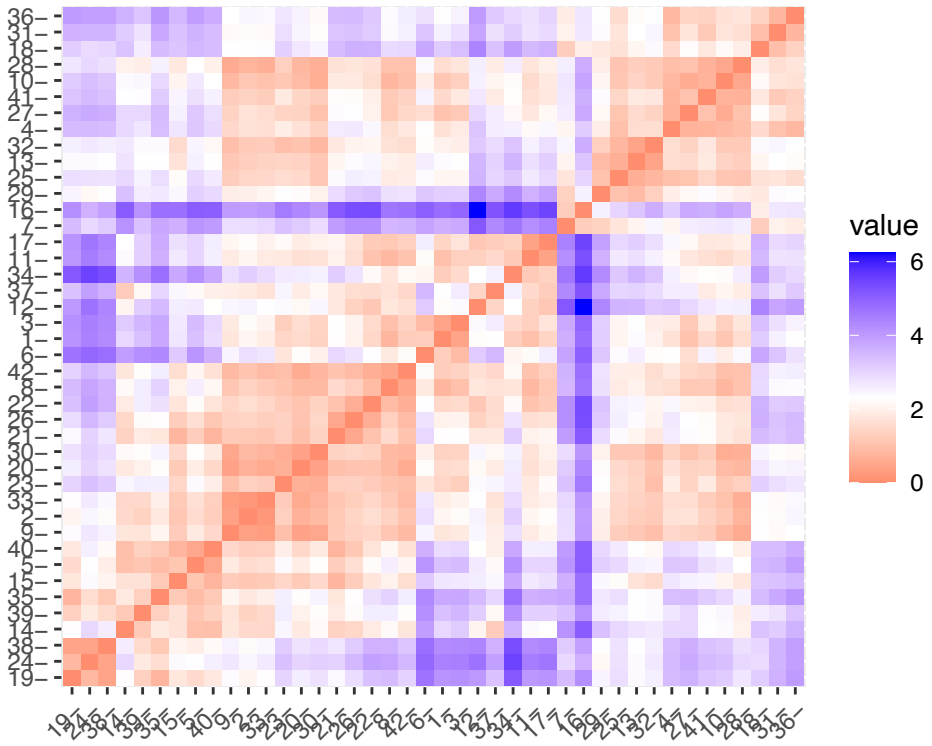


Figure 2.2: Colorized distance matrix (4% of the cases)

Based on the distance matrix, the first step in the K-means clustering procedure is to specify the number of cluster k we want to retrieve. However, this pre-specification might be an issue. More specifically, K-Means clustering selects clusters by an iterative procedure with a random starting point for the centroids of the k clusters. Then, the Euclidean distances based on the data points and the random chosen centroids are used till convergence, which points to the moment of 'ultimate classification' (Kövesi et al., 2001). At that point, all data points are classified to the most optimal chosen cluster according to that specific procedure. When performing another trial, it is possible the results might change by the random chosen start points.

It is thought that by doing more iterations, the initial starting points will be closer to the optimal final solution because it minimizes the sum of distances between datapoints within clusters at each iteration. However, when these random starting points are not close to the final solution, there is a high change of finding no unique, every changing and, maybe, incorrect clustering results when the number of iterations is low (e.g. Cheung, 2003). This describes the *sensitivity* of the algorithms to the initial center values. Even here, the order of variables in the dataset might have an impact on the final solution (Scrucca & Raftery, 2015). One way to handle is to

check the consistency of multiple trials and, thereby, choosing the solution with the lowest total within-cluster variation. As alternative, many authors proposed specific calculations or algorithms to optimize the K-mean clustering results (e.g. Bradley & Fayyad, 1998; Duda & Hart, 1973). In 2007, Arai and Barakbah discussed the idea of implementing *Hierarchical cluster analysis* as a new approach to centroid initialization for K-Means clustering, calling the *Hierarchical K-Means* clustering procedure. In their paper, they showed the increased accuracy of applying Hierarchical clustering to multiple datasets and method comparisons. They concluded a fast (K-Means) and precise (Hierarhical) conclusion compared to K-Means clustering with random starting points. These latter took more computational issues as the number of iterations were very high. In R, this procedure can be done easily by the `hkmeans` function. However, to demonstrate the steps of this procedure, we show each step to make reasonable considerations.

2.2 Intervention of Hierarchical Clustering

In Hierarchical clustering, the basic idea is that each case is initially considered as a cluster of its own. The most similar clusters are successively merged or 'linked' until there is just one single big cluster. This bottom-up clustering approach, 'from case to cluster', is called *Agglomerative Clustering*, which will be utilized in the following sections. In reverse, starting with one cluster from which the most dissimilar cluster is divided until all observations are one cluster on their own, is called *Divise Clustering* as a 'top-down' approach.

2.2.1 Linking methods

The merging of cases in hierarchical clustering is based on an iterative procedure in which the cases with a high similarity are linked/merged/paired into a new cluster untill all clusters are all linked by one cluster. This procedure can be done by different methods. The following methods are the most commonly used and have a specific rule in the iteration process of hierarchical clustering (based on Agglomerative clustering). Data points are merged by...

1. **Average linkage method**: ... the average distance between any data point of one cluster to any data point of another cluster.
2. **Single linkage method**: ... the minimum value of all pairwise distances between data points of one cluster and data points of another cluster. Typically, this method produces long thin clusters in which the opposite ends of a cluster are farther from each other compared to two cases of different clusters. This can be an issue in partitioning the data.
3. **Complete linkage**: ... the maximum value of all pairwise distances between data points of one cluster and data points of another cluster. Simply, at each step of the iteration, it seeks for the 'farthest neighbor' to cluster. Typically, it produces compact clusters.

4. **Ward's minimum variance method**: ...the minimum total within-cluster variance. At each iteration step, the pair of clusters with minimum between-cluster distance are merged.

All these methods can each be graphically represented by a *dendogram* (see figure 2.3). At the bottom (x-axis), all cases are shown and represented as 'one cluster' each. As we move up, cases are merged based on a specific linkage method. The *Height* (y-axis), referring to the *Cophenetic distance*, indicates the distance between clusters in the Agglomerative procedure. The higher, the less similar they are. By plotting all linkage methods at once, differences can be seen easily, like the thin clusters provided bij the single linkage method.

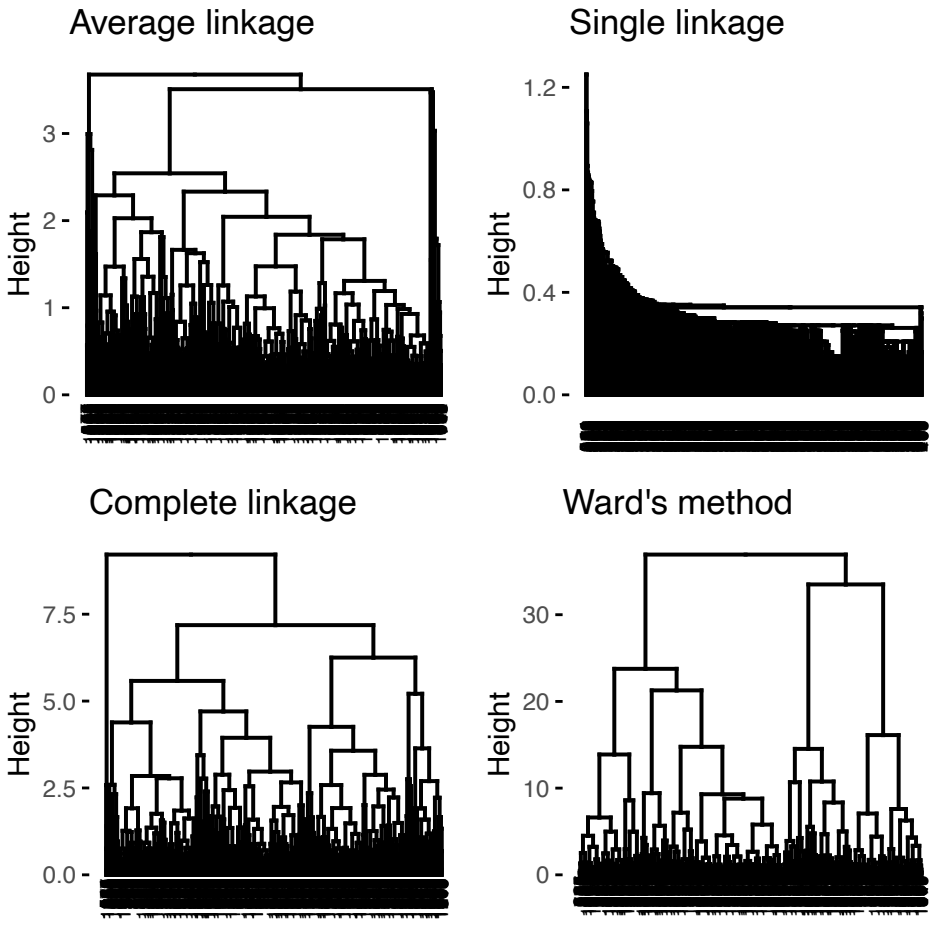


Figure 2.3: Linkage methods for Hierarchical clustering

2.2.2 Validation and choosing the number of clusters

Validating linkage methods

Dendrograms are useful, because they can be used to identify subgroups / clusters visually. However, before doing that, we could verify to what extent the dendrogram reflects the original data. This can be done by calculating the correlation between

the heights and the original distance (generated in the distance matrix, see section (Dis)similarity measurement). The closer the value to 1, the more accurately the linkage method solution reflects the data. Table 2.1 shows that the *average linking method* seems to reflect the data the best. Still, we have to keep in mind that this method commonly has the highest correlation. For this reason, the average linkage method is quite popular.

Table 2.1: Correlation Linkage Methods and Distance Matrix

Linkage method	Correlations
Average	0.578
Single	0.474
Complete	0.529
Ward	0.508

Another and more preferred way to find the best linkage method is by calculating *the agglomerative coefficient (ac)*. This allows us to find the linkage method that can identify clustering structures more strongly. The higher and closer to 1, the better the identification. Tabel 2.2 shows that *Ward's method* identifies the strongest clustering structure of the four methods.

Table 2.2: Agglomerative Coefficient for Linkage methods

Linkage method	ac
average	0.931
single	0.823
complete	0.969
ward	0.994

Number of clusters

After validating the linkage methods, we still do not know *where* or *at which height* to 'cut' the dendogram to identify the number of subgroups. Importantly, we should remember that we are doing the hierarchical procedure to find the most optimal initial start values for the K-Means (iterative) algorithm. At this point, we are not focusing on the procedure finding the most optimal number of clusters in the general clustering analysis, which we will discuss in section Determining the number of clusters. In case a more optimal number of clusters is found, this procedure of Hierarchical K-Means clustering should be redone. As mentioned before, we can do this easily by the `hkm` function.

However, this does not mean no analytical methods for choosing the most optimal number of clusters in Hierarchical clustering exists. Just for the sake of transparency, one way to do this is using *the Stopping rule*. The rule 'stops' the cluster linking at the

moment 65% of the variance is explained in the study variables by the clusters. For each number of clusters, two indices are calculated, namely the *Duda-Hart index* and the *pseudo T^2 test statistic*. The Dunn-Hart index (1973) is an internal evaluation technique which equals the ratio of inter-cluster similarity and intra-cluster similarity (see formula 2.2). Here, the distance between cluster centroids ($d(c_i, c_j)$) is divided by the maximum inner cluster variation Δk. When this Dunn index is the largest, this is the most compact solution for the cluster analysis. Therefore, the goal is to find the number of clusters where the Duda-Hart index is the highest (the larger, the more distinct the clusters) and the pseudo T^2 test statistic is the lowest (the lower, the more distinct the clusters). This can be done by using the `duda$Best.nc` code, returning 3 clusters as the most optimal number of clusters.

$$DI = \min_{i=1...m} \{ \min_{j=1...m, i \neq j} \{ \frac{d(c_i, c_j)}{\max_{k=1...m \Delta k}} \} \} \tag{2.2}$$

Table 2.3: Duda-Hart indices and Pseudo T2 statistics

	2	3	4	5
Duda-Hart index	0.5476	0.7674	0.6722	0.5248
Pseudo T2	348.6941	187.8931	210.1981	190.1517

Based on the previous steps regarding Hierarchical clustering, we end up with three clusters based on the Euclidean distance measures and the Ward linkage method. At this point, we can use colors in the dendogram to summarize the results.

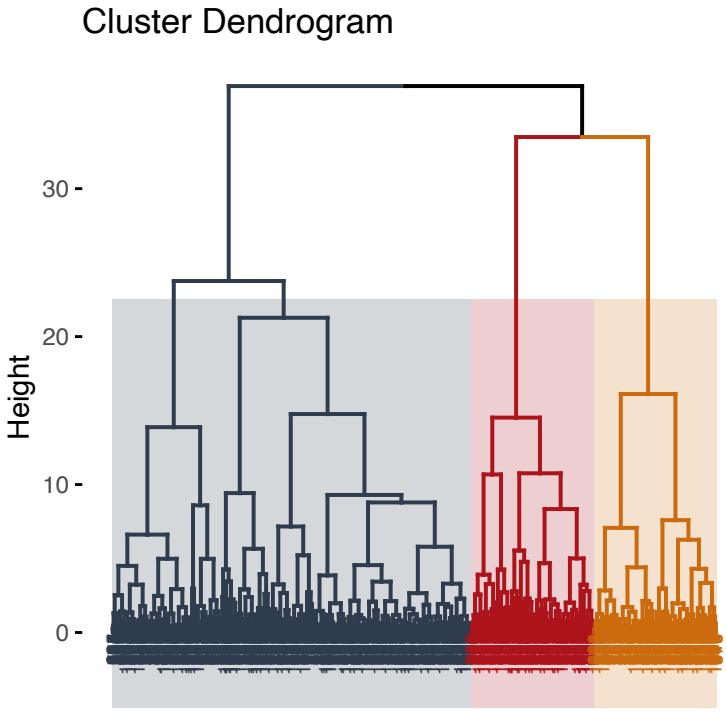


Figure 2.4: Final dendogram of Hierarchical Clustering

Remember, the result of the Hierarchical Clustering procedure is used in the K-means cluster procedure The centers of each cluster following Hierarchical clustering (see table 2.4) equal the *initial start values* for the K-Means clustering procedure, such that the K-means clustering algorithm is optimized (Arai & Barakbah, 2007).

Table 2.4: Centroids based on Hierarchical Clustering procedure

	cluster 1 (51%)	cluster 2 (25%)	cluster 3 (24%)
auto_mot	0.54	0.30	-0.71
control_mot	1.03	-0.93	-0.17
amot	0.47	-0.78	0.20

2.3 K-Means clustering algorithms

As the most widely used K-Means clustering techniques, there are several types of algorithms available, namely Lloyd, Forgy, MacQueen and Hartigan & Wong algorithms. However it is possible to arbitrarily choose the algorithm based on the characteristics of the dataset, all algorithms can be tested to gain the best possible outcome (Jain, Duin & Mao, 2000).

The Lloyd algorithm (1982) (see formula 2.3) and the **Forgy algorithm** (1965) (see formula 2.4) (with p as the probability density function and d as the distance function with c_k referring to the centroid) are good in large datasets because they apply the transformations on all cases at once. Using the k initial centroids (chosen at random or based on the hierarchical clustering procedure), all cases become part of a centroid subspace $C(R^d)$, which is based on the distance between a case and the initial centroid. Using the mean of the cases within the subspace, the values of the centroids are updated till convergence (i.e. the centroids stop changing or the difference do not reach a researcher-based threshold). As a data characteristic, Lloyd should be used for discrete variables, while Forgy is more applicable for continuous variables.

$$E = \Sigma_{i=1}^{k} \Sigma_{i=1}^{n} d(c_i, x_{ij}) \tag{2.3}$$

$$E = \Sigma_{i=1}^{k} \int p(x) d(c_i, x_{ij}) dx \tag{2.4}$$

Different from Lloyd and Forgy, **the MacQueen algorithm** (1967) reassignes a case to another centroid when that centroid is closer to the case. After such change, the centroids are updated. When the current centroid is the nearest one, the case remains in that particular subspace. This iteration process continuous till convergence.

Compared to the Lloyd, Forgy and MacQueen algorithms, **The Hartigan & Wong algorithm** (1979) is not an iteration process based on differences between old and new centroids. The distance is not from that importance. From the moment a centroid is updated by all assigned data points (that are the nearest to that centroid), the difference between the old and the new within-cluster Sum of Squares of Errors

(SSE) for each data point is calculated when that particular data point should belong to another cluster. When this SSE is smaller than the old SSE, the data point is assigned to the new cluster (see formula 2.5). Again, this procedure is repeated till convergence (i.e. no case changes to another cluster). Basically, this means that all data points belong to the cluster that results in an optimal internal similarity. One disadvantage of both the MacQueen and Hartigan & Wong algorithms is their sensitivity to the order in which the data points are relocated, by which solutions can differ according to their order. By using Hierarchical cluster analysis to calculate initial starting values, this problem disappears.

$$SSE_{other} = \frac{N_i \Sigma_j ||x_{ij} - c_i||^2}{N_i - 1} < SSE_{current} = \frac{N_1 \Sigma_j ||x_{1j} - c_1||^2}{N_1 - 1} \tag{2.5}$$

To compare, we generate both the cluster sizes (table 2.5) and within sum of squares (table 2.6) for each algorithm, showing no fundamental differences. The same conclusions can be observed in figure 2.5, where the visual result of K-Means clustering algorithms is visualized in biplots. Herein, the data is plotted across two dimensions, calculated by a Principal Component Analysis on the data selecting the two biggest components (here, explaining 85.5% of the variance). In sum, there is no outspoken algorithm to choose in the further analyses with Hartigan & Wong providing the most balanced solution.

Table 2.5: Sizes of clusters

	1	2	3
Lloyd	0.34	0.3	0.37
Forgy	0.34	0.3	0.37
MacQueen	0.34	0.3	0.37
Hartigan	0.35	0.3	0.35

Table 2.6: Within sum of squares

	1	2	3
Lloyd	475.01	612.76	592.49
Forgy	475.01	612.76	592.49
MacQueen	475.01	612.76	592.49
Hartigan	485.93	621.46	572.63

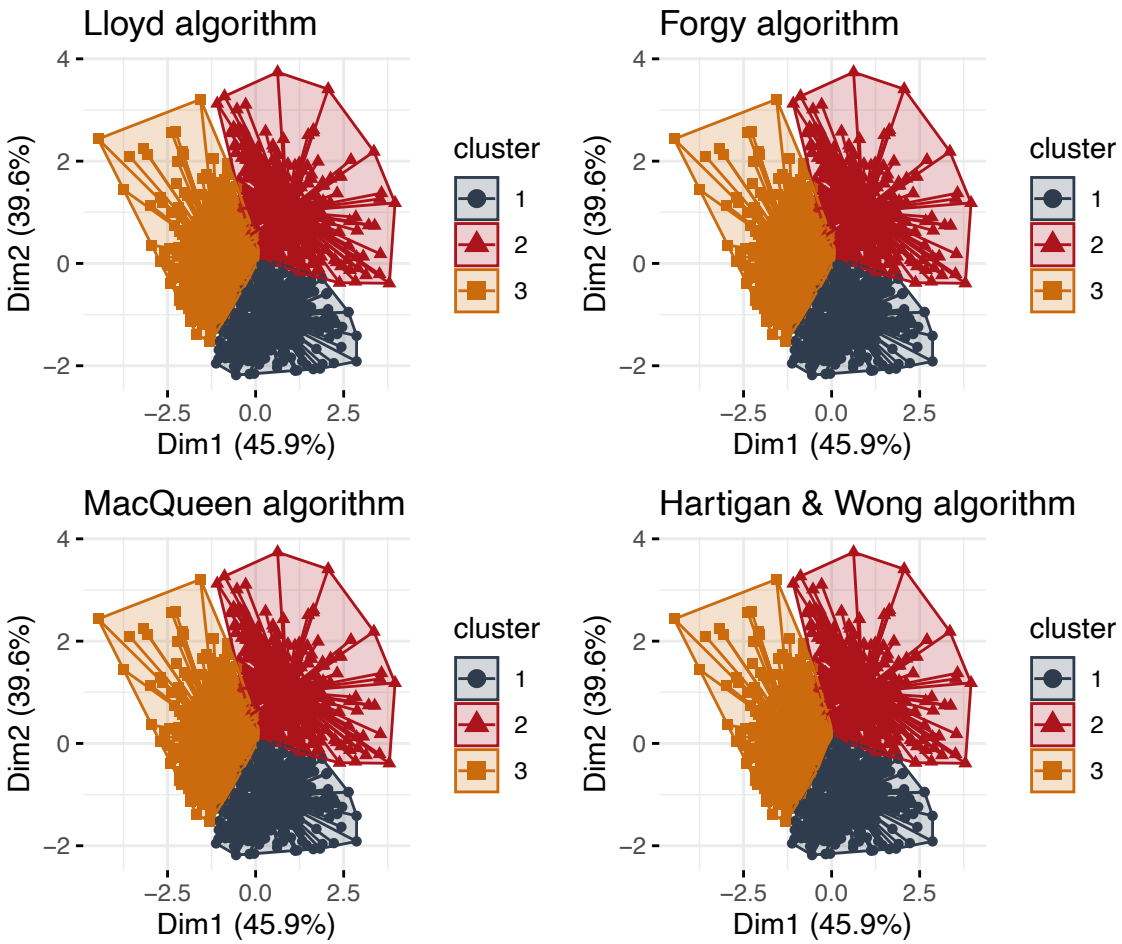


Figure 2.5: Visualizations K-Means algorithms

Based on the dataset characteristics of having continuous variables and the goal to reach minimum SSE, we choose the Hartigan & Wong algorithm. Besides, it is also the most popular method. For the current dataset, the resulting centroids of this K-Means clustering procedure are presented in table 2.7. Remark that these centroids are based on the K-Means Clustering procedure having the result of the Hierarchical Clustering procedure as initial starting points.

Table 2.7: Centroids based on Hierarchical K-Means Clustering procedure

	cluster 1	cluster 2	cluster 3
auto_mot	0.54	0.38	-0.98
control_mot	1.14	-0.53	-0.37
amot	0.56	-0.79	0.50

2.4 Analytical considerations

As we mentioned at the start of this report, the number of clusters could depend on multiple factors. For instance, a researcher can choose the number arbitrarily, based on theory or based on several validation techniques. At this point, there is no single method to determine the optimal number of clusters. In doing cluster analysis, there are multiple considerations to be made in favor of choosing the final number of clusters. Theoretically, we could prefer to test the five classic motivational profiles (e.g. Haerens et al., 2010). One could choose to stop here and use the generated results from cluster analysis. However, this idea does not guarantee qualitative and valid cluster analysis. Therefore, we recommend to use theoretical base as a starting point, followed by a series of analytical validation techniques. This is useful, as it could support the theoretical idea about motivational profiles. On the other hand, it could demonstrate the information that is available in the dataset, as more or less clusters could be found. In sum, cluster analysis is not a story of being true or not and, just like outlier handling, we recommend to use several techniques and discuss them extensively in research papers.

In the following sections, we will present several techniques to check whether the data is 'clusterable' and how the optimal number of clusters can be determined.

2.4.1 Clustering Tendency

`dotoset` versus random dataset

When applying a clustering algorithm to the dataset, it is reasonable to first assess *the clustering tendency*. By this, we mean to check to what extent it is suitable to cluster the dataset into meaningful clusters or whether just random structures will be found. This is important, as the current procedure allows to have clusters in any kind of dataset, while there might be no clusters to discuss. This procedure also responds to the paper of Vansteenkiste et al. (2009) checking the existence of motivational profiles. In doing this, we make a random dataset based on the variance of the original (scaled and without outliers) `dotoset`. This is done to compare the original dataset, from which we assume to have clusters, to a random dataset, from which we assume not to have clusters.

To compare both datasets visually, we first reduce the dimensionality (i.e. 3 variables) to a bidimensional scatterplot by performing PCA and extracting the two biggest components (declaring 85.5% of the variance in the original dataset and 68.1% in the random dataset; see figures 2.6 and 2.7).

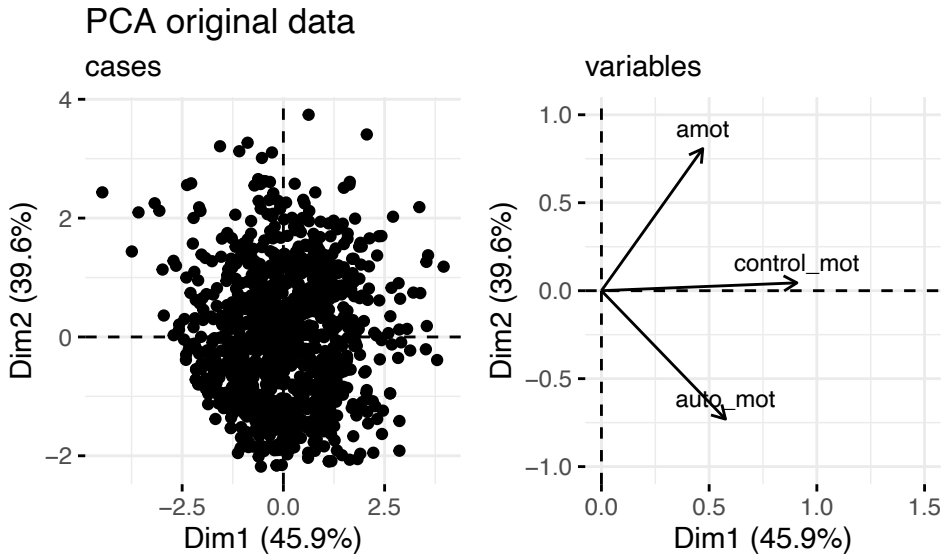


Figure 2.6: PCA original dataset

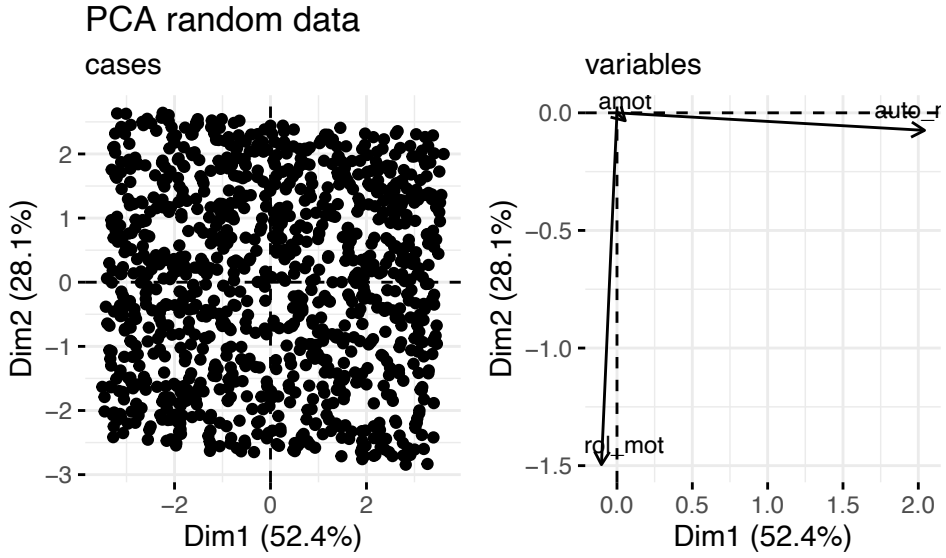


Figure 2.7: PCA random dataset

Performing Hierarchical K-Means clustering on both datasets, a comparison between figure 2.8 left (original dataset) and right (random dataset) shows that there is a classification in both datasets, however the same clustering procedure becomes other results. On the left, we might notice a more clear partitioning of the dataset into 3 clusters, while the same procedure provides less clear clusters in the random dataset (right). However they are less clear, we remark that the Hierarchical K-Means clustering still imposes a partitioning of the data, even when it comes from a random uniformally distribution and no meaningful clusters are included. This shows the reason why it is important to evaluate the cluster tendency and cluster validations.

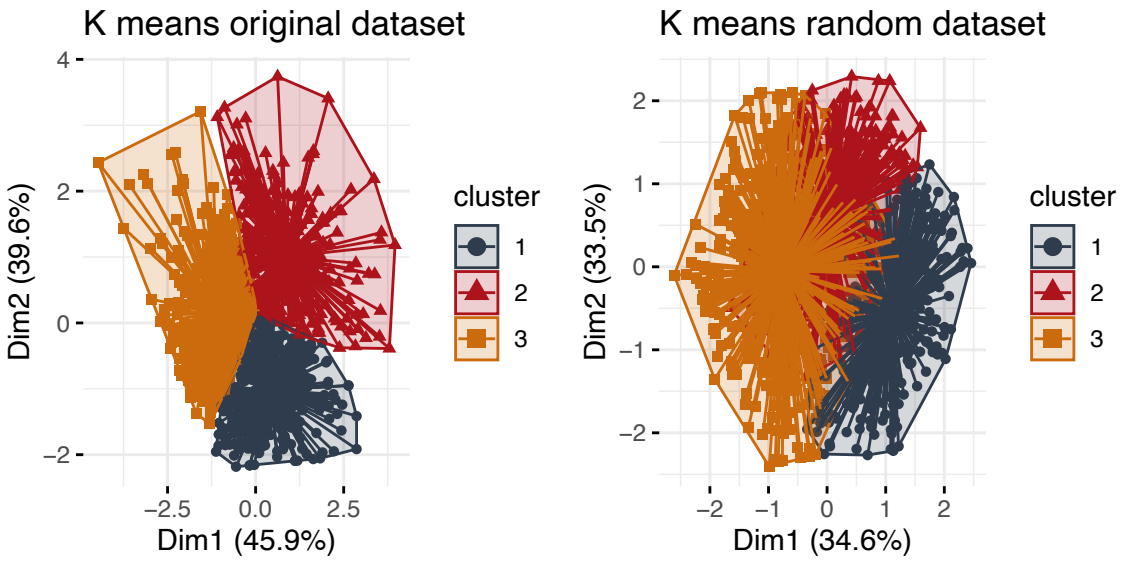


Figure 2.8: Comparison clustering results between dotoset and random dataset

The Hopkins statistic

In line with previous procedure, we can also check analytically to what extent the data is generated at random or not. The Hopkins statistics H (Lawson & Jurs, 1990) measures this 'spatial randomness' of a dataset. As a result, a probability is given representing to what extent the dataset is generated by a uniform data distribution. In the calculation, a uniformly sample of n points is sampled from the dataset. For each data point, the distance to each nearest neighbor is calculated, denoted in the distribution x_i. Next, a uniformly distribution y_i is denoted based on the distances between datapoints in the random dataset. In the formula of the Hopkins statistic (see), the mean nearest neighbor distance in the random dataset is divided by the sum of the mean nearest neighbor distances in the real and across the random dataset. When the original dataset is uniformly distributed, then $\Sigma_{i=1}^{n} x_i = \Sigma_{i=1}^{n} y_i$, meaning that the Hopkins statistic would be 0.5. When their would be clusters in the original dataset, the distances in the random dataset would be larger on average, by which the value of H would increase. A value for H higher than 0.75 indicates a good clustering. When it is lower than 0.5, then it is unlikely that the dataset has statistically meaningful clusters, indicating that there is no difference between the original dataset and the random dataset.

$$H = \frac{\Sigma_{i=1}^{n} y_i}{\Sigma_{i=1}^{n} x_i + \Sigma_{i=1}^{n} y_i} \tag{2.6}$$

Calculations on the current datasets shows that the `dotoset` dataset is more clusterable than the random dataset (see table 2.8).

Table 2.8: Hopkin statistics

	dfwithout.sc	random dataset
Hopkin statistic	0.804	0.501

2.4.2 Determining the number of clusters

Based on a theoretical framework, a researcher can determine the number of desired clusters a priori the analyses. Analytically, we discuss several techniques we advice every researcher to check, even when the number of clusters is only based on theory. Performing multiple validation techniques is important as the interpretation of the results contain a high level of subjectivity. Therefore, an important consideration in determining the optimal number of clusters is to keep in mind that we are searching for the lowest number of clusters, while still being statistically relevant. In doing this, we present several statistical tests to check the optimal number of clusters in a range of maximum 5 clusters given the theoretical background (e.g. Haerens et al., 2010) and the current dataset.

Elbow method

In clustering, the goal is to find a number of cluster to maximize the 'compactness' of clusters or, in other words, to minimize the total intra-cluster variation or Within-Cluster-Sum of Squared distances ($WSS(k) = \Sigma_{i \in k}(x_i - \mu_k)^2$ with x_i data point and with μ_k as the centroid of k). With the elbow method, we calculate the WSS for a range of cluster numbers. Subsequently, the results are shown graphically as a function of the number of clusters k. Typically, a 'knee' or a notably bend in the plot is considered as an appropriate number of clusters. This bend represents the number of clusters from which point the added WSS has no fundamental contribution anymore. Here, we observe a bend at the number of 3 or 4 clusters (see figure 2.9, left). Figure 2.9 (right) shows the same figure, with adding the proportion of between-subject variance. This can be useful as we want to maximize both the within-cluster variance and the between-cluster variance. The number of clusters showing a balance can support the decision of the 'knee'. Though, the interpretation might be ambiguous. A good alternative is using the Average Silhouette Method.

Average Silhouette method

The Average Silhouette method (Kaufman & Rousseeuw, 1990) is an indication for the quality of clustering or, in other words *"how well each case lies within its assigned cluster"*. Analytically, the silhouette coefficient measures the similarity of a data point to the other data points in the same cluster versus those in a neighbor cluster. The closer to 1, the better a data point is clustered or the more similar it is to other data points in the cluster. The more it approaches -1, the more dissimilar the data point is to the other data points in the same cluster. For each data point, the silhouette coefficients is plotted in figure 2.10 (left). Next, it was sorted by value and cluster,

in favor of observing the number of data points with a low or a negative silhouette coefficient. As analysts, we can find the number of the case and assign it to its closer (neighbor) cluster (table 2.9). Remark that the number of negative values indicate the quality of clustering, which is 3% of the total cases in case of 3 clusters. These coefficients also can be used to determine the optimal number of clusters. In line with the Elbow method, the average silhouette coefficients for a range of clusters is plotted as a function of this range (figure 2.10 (right)). The location with high values for the average silhouette width indicates a good clustering.

Gap statistic

The Gap statistic (Tibshirani, Walther, & Hastic, 2001) can be used to perform hypothesis testing (see formula). In doing so, the total within intra-cluster variation is calculated for k number of clusters, W_k. Second, the same is done based on a (merged) dataset coming from B datasets with random uniform distributions, W_{kb}. Third, the observed values W_k and the expected values W_{kb} are compared under the null hypothesis.

$$Gap(k) = \frac{1}{B}\Sigma_{b=1}^{B} log(W_{kb}^{*}) - log(W_k) \tag{2.7}$$

The smallest number of clusters where the Gap statistic is within one standard deviation of the gap at *k+1* represents the optimal number of clusters $Gap(k) >= Gap(k+1) - s_{k+1}$. Using $B = 500$ give precise results so that the gap plot is basically unchanged after an another run. Here, we notice in the Gap statistic figure 3 as the number of clusters with the highest Gap statistic, compared to 1 (which is uninformative).

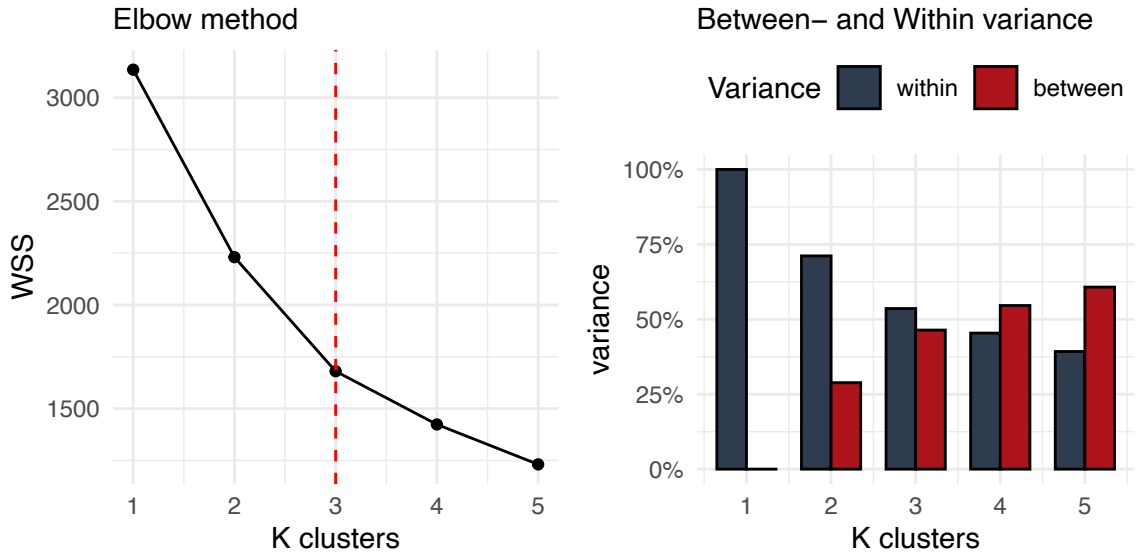


Figure 2.9: Elbow method (left) and Variance (right) plot

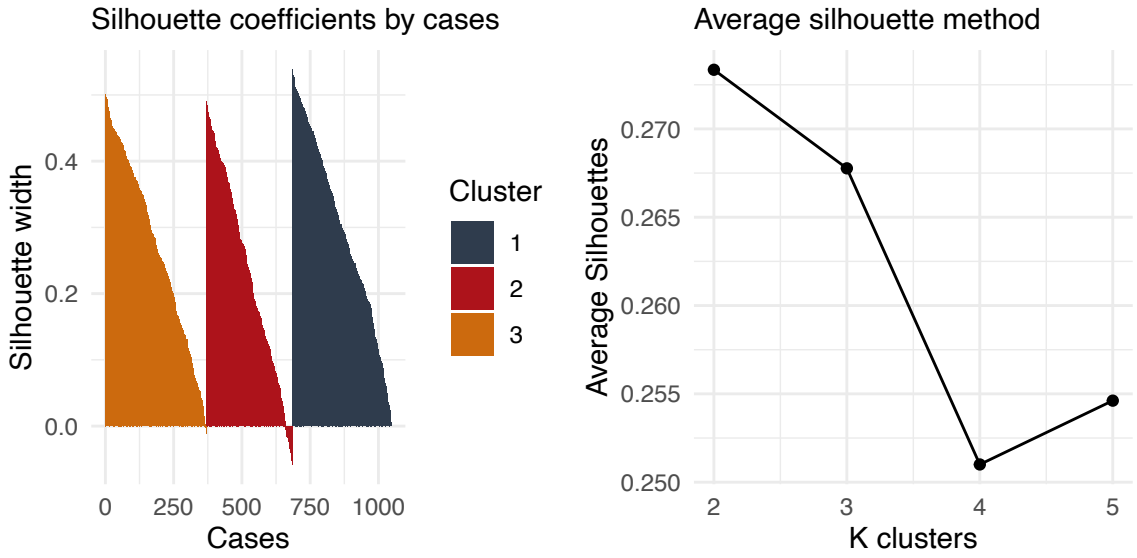


Figure 2.10: Individual (left) and Average (right) Silhouette plot

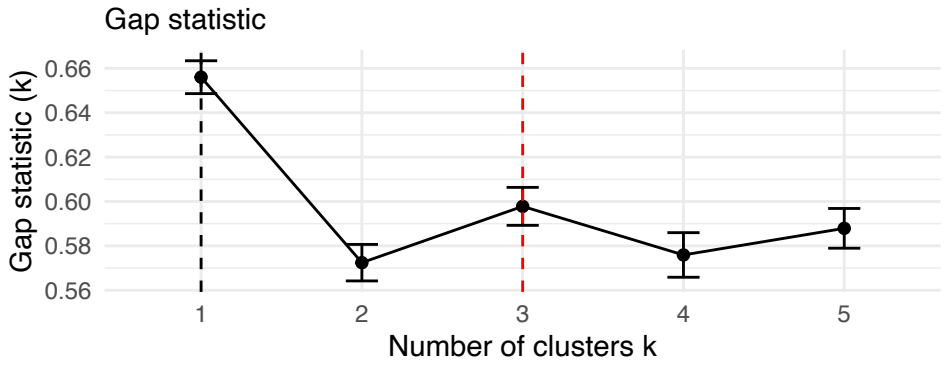


Figure 2.11: Variance comparison method

Table 2.9: Head of cases with negative silhouette widths

	cluster	neighbor	silhouette
849	1	3	-0.0011541
227	1	3	-0.0025773
205	1	3	-0.0047139
441	1	2	-0.0051060
333	1	3	-0.0068432
395	1	2	-0.0085794

Summary of 30 indices

Charrad et al. (2014) developed a package `NcBlust` which allows to calculate 30 validation indices simultaneously and, subsequently, generate a frequency of how many times specific indices propose a specific number of cluster as most optimal according to the rules of each specific index. However Milligan and Cooper (1985) also investigated 30 indices for cluster analysis, these were not all implemented in the `NcBlust` package. Charrad et al. (2014) described that some details could not be found or some indices were method-dependent. Still, the Caliniski and Harabasz (CH-index; 1974) is one recommended method by Milligan and Cooper (1985), showing the largest *CH*-index (=342.55) for 3 clusters for the current dataset. In addition, multiple indices are included for which no package was available up to today in R. For the sake of simplicity, we will not go into further details. For more information about the indices, we would like to refer to the paper of Charrad et al. (2014) in Journal of Statstistical Software (2014) where all indices are explained extensively. The code `$All.index` can be used to observe the results of all indices and `$All.CriticalValues` generates all critical values resulting from these indices. This summarizing approach is useful, as it could be an important factor in choosing the number of optimal clusters. In the current report, we plot the output of `$Best.nc` to show the number of indices as a function of the number of clusters. Convincingly, 13 indices out of the 30 show 3 as the most optimal number.

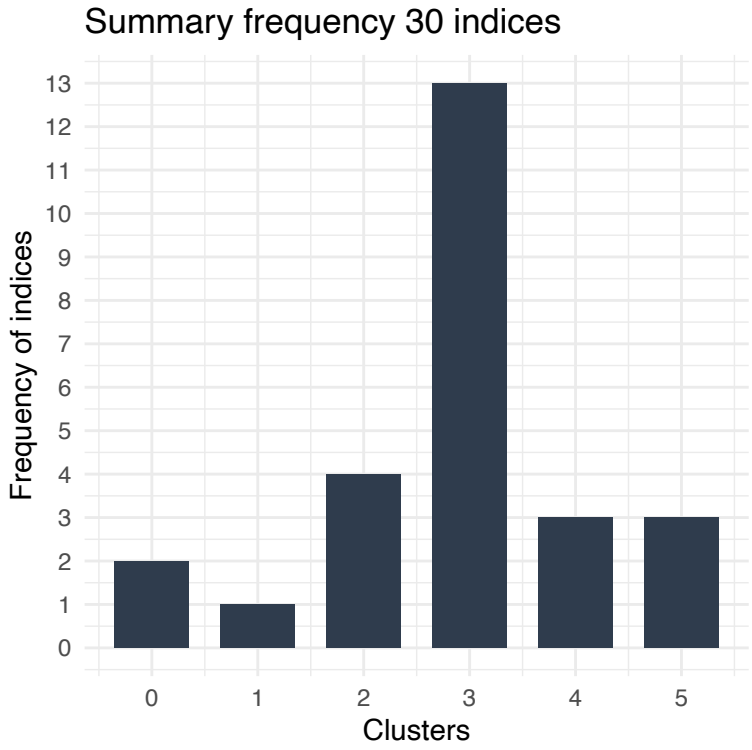


Figure 2.12: Summary of 30 indices

Cluster overlap

K-Means has the goal to partition the data in the most optimal way. After calculating the distance matrix, validating the hierarchical clustering method and choosing the K-Means algorithms, we can use the biplots to compare the K-Means results over a range of clusters. By doing this, we have a visual detection of overlap, size of the clusters and their orientation towards each other. In figure 2.13, we plotted the 'data coverage' surfaces for the results of 2 - 5 clusters showing a good partition of the data up until 5 clusters. Beware, based on this figure, a researchr could decide any number of clusters fitting the hypothesis. Therefore, presenting this visual method at last, we emphasize to use the validation techniques as we mentioned before. The visual graphs do not have any indication of the within- versus between-variance, the quality of clustering, etc. By using these, the researcher can make a statistically-based decision.

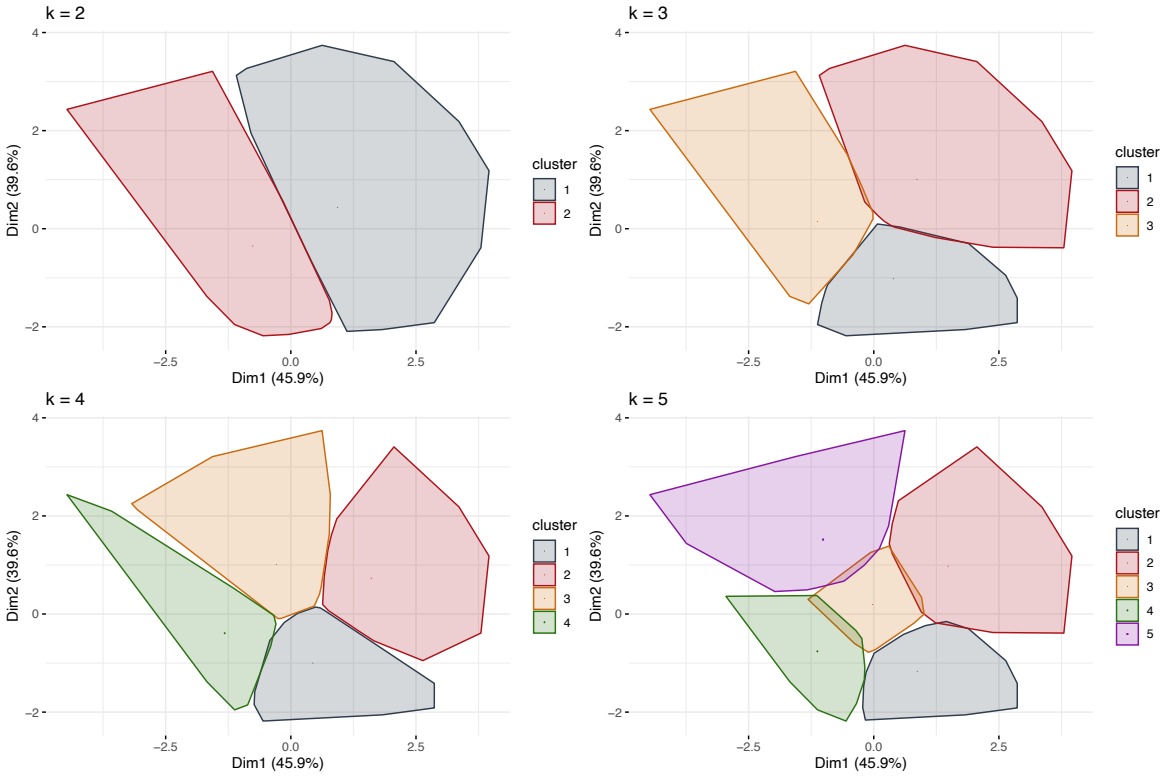


Figure 2.13: Range of K-Means clusters

To conclude, we have presented the Hierarchical K-Means clustering procedure, this with the goal to provide the most optimal way to find cluster in the current `dotoset` dataset. In the current report, we discussed this procedure extensively. In R, all these steps can be done by the `hkmeans` function (as shown below). Importantly, the function allows the specific steps we have discussed, namely the distance measure, the hierarchical clustering linkage method, the number of iterations (by default 10) and the preferred K-Means algorithm.

```
hkm <- hkmeans(dfwithout.sc, k = 3,
               hc.metric="euclidian",
               hc.method='ward.D2',
               iter.max = 10,
               km.algorithm = "Hartigan-Wong")
```

2.5 Cluster characteristics

Ending up with a new variable `K-Means cluster membership` and its centroids regarding the study variables, different visualization techniques are available presenting each cluster's characteristics.

2.5.1 Cluster stability

In the study of Vansteenkiste et al. (2009), a procedure was described to test the stability of the cluster solution. On purpose, we did not mentioned this as an index to determine the number of clusters, as it approaches a cluster feature rather than a validation technique (Breckenridge, 2000). In this 'double split cross-validation' procedure, the first step is to divide the total sample into two random subsamples A (50%) and B (50%). Next, the Hierarchical K-means clustering procedure is performed in both subsets. In the third step, the K-means procedure is applied again. This time, the centroids of the other subset are used as initial values instead of the centroids resulting from hierarchical clustering. For instance, the centroids of subset A are used in subset B and vice versa. In the end, the stability is checked by calculating Cohen's Kappa-index k for the correspondence between the subsample-clustering result and the clustering result coming from the original hierarchical K-means clustering (see section [Cohen's kappa] for more information regarding the Kappa index). For subset A, $k = .42$ with a p-value of $<.001$. For subset B, $k = .94$ with a p-value of $<.001$. Here, the p-value refers to the hypothesis test of having agreement between both cluster solutions by chance (H_0), which is rejected by the low p-values. To conclude, both kappa's are averaged to .68, stating for an acceptable cluster stabilty ($>.60$; Asendorpf et al., 2001).

2.5.2 Heatmap

The first visualisation of content-widely cluster characteristics is a heatmap (figure 2.14) where all cases in the dataset are sorted according the result of the hierarchical clustering procedure. This can be useful to have a first insight to clusters' characteristics and sizes. Here, three approximately equally sized clusters can be noted. The color code refers to the z-score on each variable for each case. Interpreting this figure, the upper cluster shows high scores for autonomous motivation and very low scores for amotivation. The middle cluster shows low scores for all study variables. The bottom cluster shows high scores on amotivation and controlled motivation, while scoring low for autonomus motivation.

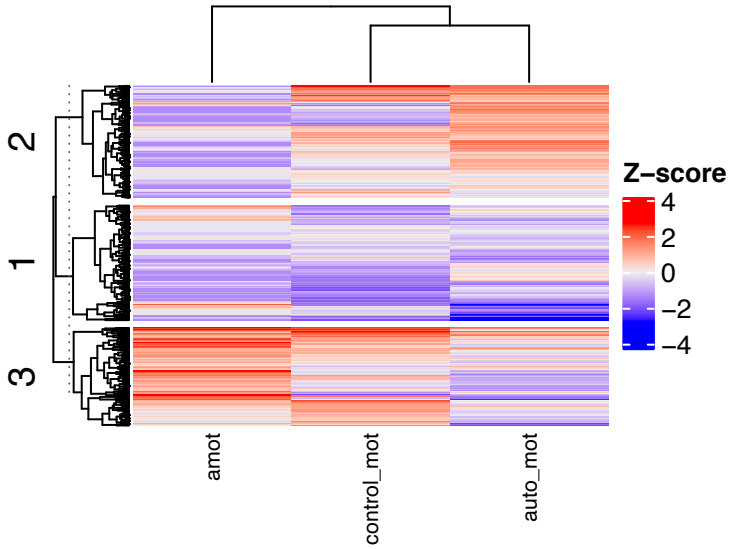


Figure 2.14: Heatmap clustering results

2.5.3 Biplot

The second is *a bidimensional plot* or biplot (figure 2.15). Herein, the function automatically performs a Principal Component Analysis (PCA) and uses the two largest components for both axes (explaining 85.5% of the variance). Next, all data points are plotted against these axes and colored according to their assigned cluster. The color surfaces are added to presents the clusters more clearly.

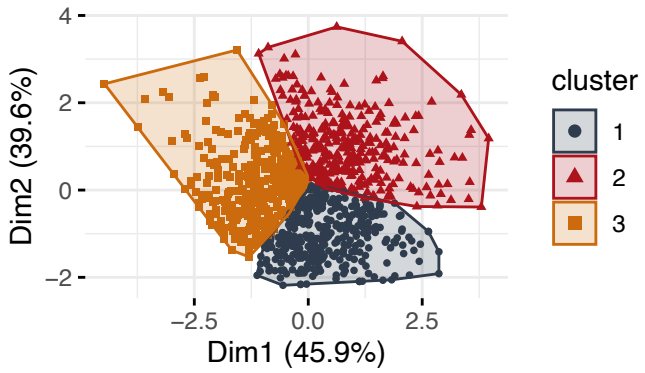


Figure 2.15: PCA biplot clustering results

2.5.4 Barplot

The third is a barplot (figure 2.16), showing the centroids of each study variable sorted by clusters. As we standardized the variables in section Standardizing, the barplot is useful to notice clearly to what extent the cluster score on variables compared to the other clusters. Moreover, the figure is useful as it provides content information about the clusters. In terms of the current dataset, cluster 1 clearly has high cores for autonomous motivation, followed by controlled motivation and low

scores for amotivation, referring to the features of a *'good quality'* motivation group (Vansteenkiste et al., 2009). The second cluster has high scores for amotivation and controlled motivation and low scores for autonomous motivation, referring to the *'bad quality'* motivation group. Cluster 3 clearly refers to a *'low'* motivation group, scoring low on all types of motivation.

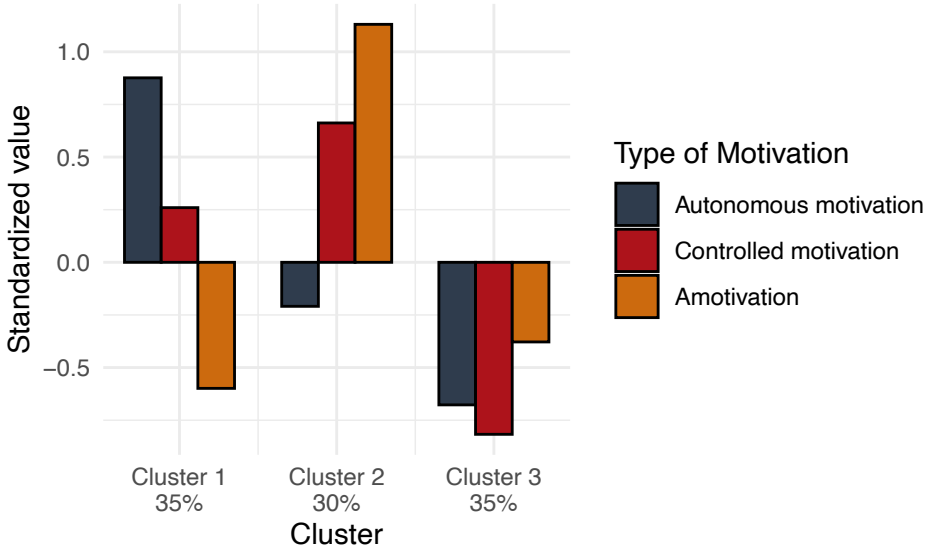


Figure 2.16: Barplot clustering results

2.5.5 Between-cluster differences

At last, we can test the differences between clusters in terms of the (unstandardized) study variables by performing a Multivariate Analysis Of Variance (ANOVA), resulting in significant cluster differences (Wilks's lambda = .20, $F(6, 2082) = 425.47$, $p < .001$). By doing this, we provide feedback of the clustering result to the theoretical concepts (e.g. Friederichs et al., 2015). Exploring univariate regression test with the *Good motivation* profile as reference level, significant differences between clusters are found for each study variable, showing that the *Good motivation* profile scores significantly higher for autonomous motivation compared to the other profiles (table 2.10). By looking the R^2, the cluster membership variable explains 44.1% of the variance in the variable 'autonomous motivation.'[1] Table 2.11 shows that the *Bad motivated* profile scores higher and the *Low motivated* profile scores significantly lower on controlled motivation, in comparison to the *Good motivated* profile ($R^2 = .392$). Table 2.12 shows that the *Low motivated* profile scores significantly higher on amotivation,

[1]Some researchers argue that the percentage of explained variance also can be used as validation technique to decide the number of clusters. For instance, this principle is used in the VARCLUS procedure in SAS or the 'stopping rule' in hierarchical clustering (e.g. Calinski & Harabaz, 1974) which 'stops' at the height of the cluster number explaining 65% of the variance, claiming for increased 'group tightness' (see Number of clusters). However, remember that we used other techniques to provide clustering properties (e.g. elbow method) in favor of maximizing the within-cluster variance and minimizing the between-cluster variance of the data.

followed by the *Bad motivated* profile in comparison to the reference level ($R^2 = .56$). These results are in line with the theoretical expectations.

Table 2.10: K-Means clusters in the prediction of autonomous motivation

	term	estimate	std.error	statistic	p.value
1	(Intercept)	4.51	0.02	226.45	<.001
2	Kmean_Bad	-0.55	0.03	-18.81	<.001
3	Kmean_Low	-0.78	0.03	-28.06	<.001

Table 2.11: K-Means clusters in the prediction of controlled motivation

	term	estimate	std.error	statistic	p.value
1	(Intercept)	2.65	0.03	85.98	<.001
2	Kmean_Bad	0.30	0.05	6.68	<.001
3	Kmean_Low	-0.81	0.04	-18.64	<.001

Table 2.12: K-Means clusters in the prediction of amotivation

	term	estimate	std.error	statistic	p.value
1	(Intercept)	1.55	0.03	48.16	<.001
2	Kmean_Bad	1.59	0.05	33.77	<.001
3	Kmean_Low	0.20	0.05	4.49	<.001

Chapter 3

Latent Profile Analysis

Latent Profile Analysis (LPA) is a statistical method to uncover hidden groups in a (continuous) dataset. The word *'latent'* refers to the fact that the groups are unknown and no a priori information is available. When the variables are categorical, this procedure is called Latent Class analysis (LCA). Importantly, LPA is the name for mixture modelling techniques within social sciences. Simply stated, *mixture modelling* refers to the statistical concept of finding an unspecified combination of multiple density functions, as sub-populations or 'groups' or 'profiles' or 'clusters' or 'mixture components' (all are synonyms), within a general density function, referring to the total population. The *'discovering'* of these profiles is done by statistical inferences. Herein, we try to estimate the parameters of k profiles / density functions in the most optimal way. A popular statistical inference method in doing this is the *EM-algorithm* (see section The EM-algorithm). Subsequently, the profiles differ in terms of these parameters. For instance, Gaussian distributed mixture components (or clusters) will differ in terms of their means and variances. This points to the assumption that all k mixture components are thought belonging to the same parametric distribution family (e.g. all Gaussian distributions). In the results of mixture modelling, a mixture weight is calculated for each data point. It refers to 'a certain probability of belonging to each of the k mixture components or clusters'. Based on this probability, each data point has a particular level of 'uncertainty'. At the end, the data point is assigned to the cluster with the highest probability of belonging to.

Here, we have drafted the idea of LPA, also called 'mixture modelling', in a brief way. In the next sections, we want to focus on the most common used methods, namely Gaussian Mixture Modelling using the EM-algorithm. Before continuing, we will focus on these concepts in the most sufficient way in the scope of the current report, this for the reason that the literature of statistical inference knows a high level of mathematics.

3.1 Gaussian Mixture Modelling

Gaussian Mixture Modelling or *GMM* refers to the mixture modelling application where the underlying density functions in the dataset are thought to have Gaussian

(Normal) probability distributions. In short, it is thought the model has a mixture of Gaussian distributions. In figure 3.1, the idea is shown in case of a univariate model with the Gaussian density plot of the total population on the left and the mixture Gaussian distributions coming from GMM on the right. As we are working with more variables (i.e. 3 motivation variables), we will work with a multivariate design. Figure 3.2 shows the visualization of a bidimensional density plot, imaged as a mountain. The color legend can be used to interpretet the height of the density. The plot on the right is a 3D representation of the plot on the left, showing the 'density mountain' resulting from the multivariate density function (see formula 3.1). As a consequence, we need to estimate the mean and the variance, both as parameters of the Gaussian density function $N(\mu_k, \sigma_k)$ with k as the number of mixture components. Here, we want to know the probability that the ith data point comes from the k Gaussian distribution, expressed as $p(x_i = k|\phi)$ where ϕ stands for the Gaussian parameters mean μ_k, covariance σ_k and weight π_k. This weight π_k is the unknown probability of selecting component k, satisfying $\Sigma_{k=1}^{k}\pi_k = 1$, meaning that the total probability space for a data point is equal to one. For all cases, the loglikelihood function is thereby expressed as $L(\phi) = p(x|\phi)$.

$$\phi(x; \mu_k, \sigma_k) = \frac{1}{|2\pi\sigma_k|^{1/2}} exp(-\frac{1}{2}(x - \mu_k)^T \sigma_j^{-1}(x - \mu_k)) \tag{3.1}$$

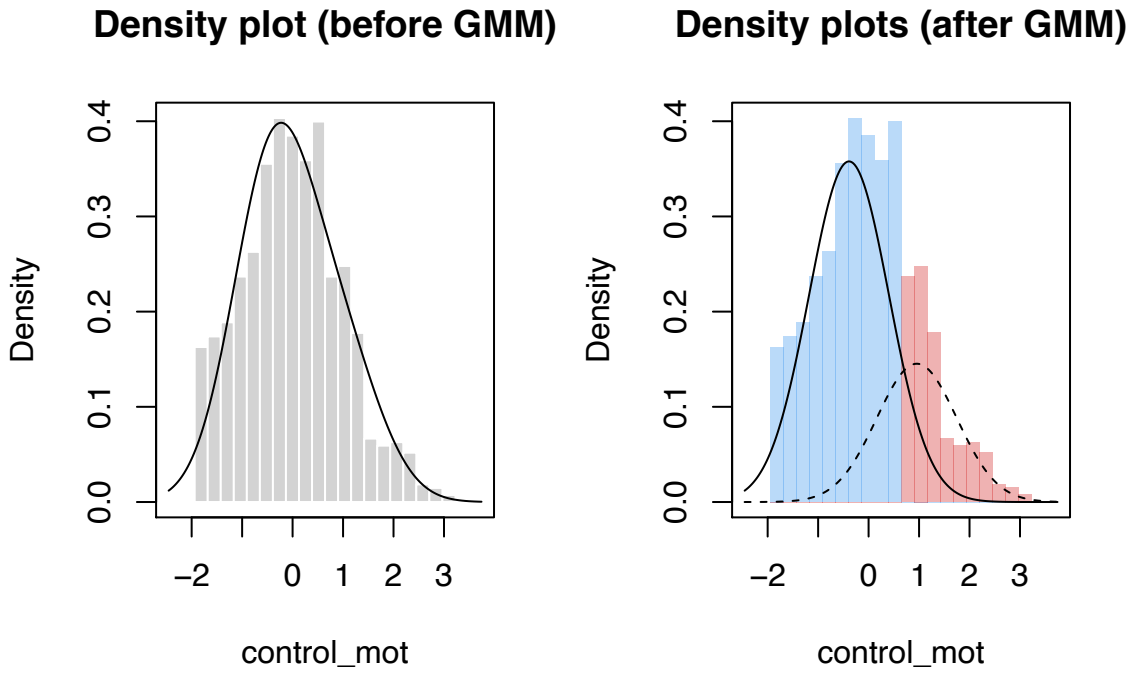


Figure 3.1: The idea of univariate mixture modelling

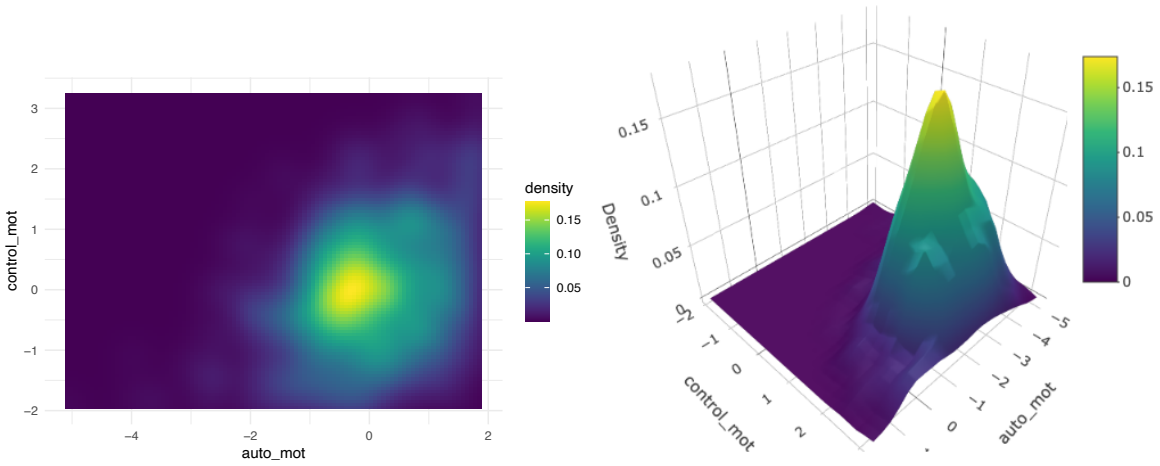


Figure 3.2: Colorized bivariate density function in 2D (left) and 3D (right)

However, the cluster membership of each case is hidden and unknown. Therefore, GMM utilizes an *iterative process* starting from some initial values. The determination of these initial values is important, because it can affect the consistency and accuracy of the results (Wu, 1983), while mostly random start values are chosen. Equivalent to K-Means clustering (section Intervention of Hierarchical Clustering), *(model-based) hierarchical clustering* is applied to select the most optimal initial start values of each mixture component (Maitra & Melynkov, 2010). More specifically, model-based hierarchical (agglomerative) clustering (MBHAC) is done by combining those clusters with the smallest decrease in the classification likelihood (Banfiel & Raftery, 1992). Additionally, MBHAC is computationally advantageous because one single run can be used for different number of mixture components, while an iterative process with random values would take more iterations to become the same result. In the literature, this iteration process refers to the EM-algorithm.

3.1.1 The EM-algorithm

In LPA, it is assumed that each data point has an unobserved data point, namely the 'mixture component' it belongs to. To find the probability of belonging to each mixture component (i.e. the mixture weight), several techniques can be used to estimate the parameters of the related probability distribution. Importantly to mention is that these are no methods to determine the optimal number of mixture components given the observed dataset (see section Choosing number of components), as we did in section Determining the number of clusters for K-Means cluster analysis.

The most popular technique for parameter estimation is *Expectation Maximization* or the *EM-algorithm* (Dempster, Laird, & Rubin, 1977). The basic idea is that one set of values is used to estimate a second set of values, which is used in his turn to find a better estimate of the first set. The name refers to the two steps in this algorithm, namely the *Expectation* and the *Maximization* step. Given an a priori given number of components and initial start values provided by the (model-

based) hierarchical clustering procedure, the E-step assesses the likelihood a data point is coming from cluster k (i.e. estimating cluster memberships), while the M-step maximizes the parameters that the data point is coming from the cluster k. To put it in a practical example, the learning process of an exam is comparable to the the EM-algorithm. A student will gather all course information, this with the goal to maximize the chances of passing the exam, which will stimulate the student to gather more information, this to maximize the chances, etc. This process stops when the student passes the exam. In statistical terms, the EM-algorithm is an iterative procedure 'till convergence', which refers to the endpoint where the comparison between the underlying statistical model of the data and the latent representation of the data cannot be updated anymore.

In reality, the EM-algorithm exists of three steps, where the *initialization step* equals step 'zero'. Herein, the first initial values for the mixture weights are provided. This can be done by (model-based) hierarchical clustering procedure. After this step, an iterative procedure starts between the E- and the M-steps.

Step 1: the Expectation step

In the E-step, the mixture weights are updated. Basically, it provides updated parameter values θ for the latent variable 'cluster assignment' Z. This is done by finding the expectation of the log-likelihood $log[P(X|Z,\Theta)]$ (see formula 3.2) with respect to the current mixture components Z, conditioned on the previous statistical parameters Q^* and the observed data X. This means that it assesses how likely each data point is coming from the k mixture components $P(Z|X,\Theta^*)$. By this, it finds estimates for Θ that finds maximum likelihoods $P(X|\Theta)$.

$$Q(\theta,\theta^*) = \Sigma_{i=1}^{n}\Sigma_{i=j}^{k}P(z_i = j|X,\theta^*)log[P(z_i = j|\theta)P(x_i|z_i = j,\theta)] \tag{3.2}$$

Step 2: the Maximization step

With the expectation values for the cluster membership coming from the E-step (i.e. complete data), the Maximization step attempts to maximize the parameters of the mixture model. Herein, we 'evaluate' the estimated parameters by finding their maximum by the formula $\theta^* = \underset{\theta}{argmax} Q(\theta,\theta^*)$. This iterative procedure is repeated till convergence, which refers to the moment that each update of the model parameters cannot reach a certain threshold. Because we use the model-based hierarchical clustering procedure to initiate values to the process, it is not required to rerun the procedure several to optimize the parameter estimation.

In the EM-algorithm of GMM, the Gaussian density parameters μ and σ are calculated, this in the light of mixture weights (probabilities) to weight each case in the parameter estimation. This 'weighting' is important, as it will assign cases with higher probabilities to a particular profile at each iteration in the EM-algorithm. In the case of multivariate GMM, this results in means and a covariance matrix.

An important issue in GMM to mention is that the results of the EM-algorithm depends on the order of the variables. To avoid this, it is recommended to use the model-based hierarchical clustering procedure and the Singular Value Decomponsition-transformation or the *SVD* on the study variables (Scrucca & Raftery, 2015). By this latter transformation, the separation among clusters is

enhanced, which is positive for the MBHAC. Importantly, from the moment the EM-algorithm starts, the variables will be used in the original scale (remember we have standardized these).

In practice, the MPlus software is commonly used to perform LPA. However, it is costly and closed-source. Using R, the package `mclust` is an increasingly popular package to perform mixture modelling. It provides sufficient functions for clustering, classifying and density estimation compared to other R packages (e.g. `mixture`, `EMCluster`, `bgmm`) and can be combined easily with other packages like `factoextra`, which we will use for plotting. When the researcher is dealing with non-Gaussian components, it is recommended to use the `Rmixmod` (the second popular R package), the `mixtools` (the third popular package) or the `flexmix` package. At last, when dealing with a large dataset ($>$ 2000 cases), it is recommended to provide a subset before applying the EM algorithm.

3.2 Model selection

Before discussing several fit indices to determine the optimal number of clusters, it is important to emphasize the *flexibility* of mixture modelling compared to other types of cluster analyses. For instance, GMM assumes a multivariate Gaussian distribution for each mixture component. The parameters of the different mixture components are not constrained to be equal. This indicates that all components could be ellipsoidal (all axes are perpendicular), centered at the mean μ_k and combined with *other geometric features*. More specifically, a researcher has options to put constraints on the covariance matrix in line with the expected geometric features or to interpret the fit indices (see section Choosing number of components). Regardless of the choice, the researcher should discuss extensively the process and the decisions that are made to reach the final cluster result.

3.2.1 Geometric flexibility

By putting *constraints* on the covariance matrix, the researcher is able to direct the mixture modelling procedure. More specifically, the `mclust` package allows 14 different constraints based on the *volume* (i.e. the size of the cluster, the number of observations within clusters), the *shape* (i.e. variance or shape of the distribution of a cluster) and *orientation* (i.e. alignment across axes of two-dimensional space) (e.g. Celeux & Govaert, 1995). Instead of only referring to these options, we decided to put Table 3.1 and figure 2 (Appendix A) in the current report (see Table 3 and Table 4, Scrucca & Raftery, 2015, page 292). This is important, because it stands for an important feature of comparing the mixture modelling with the K-Means clustering procedure (see section Chapter 4). In table 3.1, E refers to 'equal variance' and V for 'varying variance'.

Table 3.1: Parameterisations of the within-group variance matrices

Model	Distribution	Volume	Shape	Orientation
EII	Spherical	Equal	Equal	-
VII	Spherical	Variable	Equal	-
EEI	Diagonal	Equal	Equal	Coordinate axes
VEI	Diagonal	Variable	Equal	Coordinate axes
EVI	Diagonal	Equal	Variable	Coordinate axes
VVI	Diagonal	Variable	Variable	Coordinate axes
EEE	Ellipsoidal	Equal	Equal	Equal
EVE	Ellipsoidal	Variable	Variable	Equal
VEE	Ellipsoidal	Equal	Equal	Equal
VVE	Ellipsoidal	Variable	Variable	Equal
EEV	Ellipsoidal	Equal	Equal	Variable
VEV	Ellipsoidal	Variable	Equal	Variable
EVV	Ellipsoidal	Equal	Variable	Variable
VVV	Ellipsoidal	Variable	Variable	Variable

3.2.2 Choosing number of components

After loading the `mclust` package, the dataset `dotoset` (with standardized variables and without outliers) is specified in the `Mclust` function to perform the model-based clustering. Herein, we did provide a range of mixture components numbers (which can be done by adding `G = 1:5`) and no covariance constraint (do not mention `modelNames=` in the function). By unspecifying this, the model will automatically run all available models allowing us to evaluate different fit indices and, thereby, to consider the most optimal number of mixture components and covariance structure. In what follows, we discuss four different types of fit indices, namely the BIC, AIC, ICL and Entropy.

BIC

The *Bayesian Information Criteraton* or *BIC* (Schwarz, 1978) is a metric for the goodness of fit (see formula 3.3) with k referring to the number of free parameters in the model, n to the sample size, $log()$ to the log-likelihood and $\hat{\theta}$ to the estimated parameter θ given the model.

$$BIC = -2log(L(\hat{\theta})) + k * log(n) \tag{3.3}$$

The idea is to calculate the BIC for different models, by which the BIC-values can be compared. The larger the difference between two models, the stronger evidence for one model over the other. The best fitting model is the one with the lowest BIC value. By visualizing the output, we get an overview of the BIC values as a function of the different covariance structures and the number of mixture components (figure

3.3). Herein, two decreasing 'collections' of models can be noticed. The model-related color and linetype allows us to find the model and component number with the lowest BIC value.[1]

Sometimes, this is referred as the 'knee point', equivalent to the interpretation of the Elbow method. However, a researcher might be unsure about the model to choose. For instance, the lines of the VEV model seem to be the lowest across the total range of mixture components. To be sure, two alternative interpretations for the BIC are available: the *BIC difference* and the *Likelihood Ratio Test*

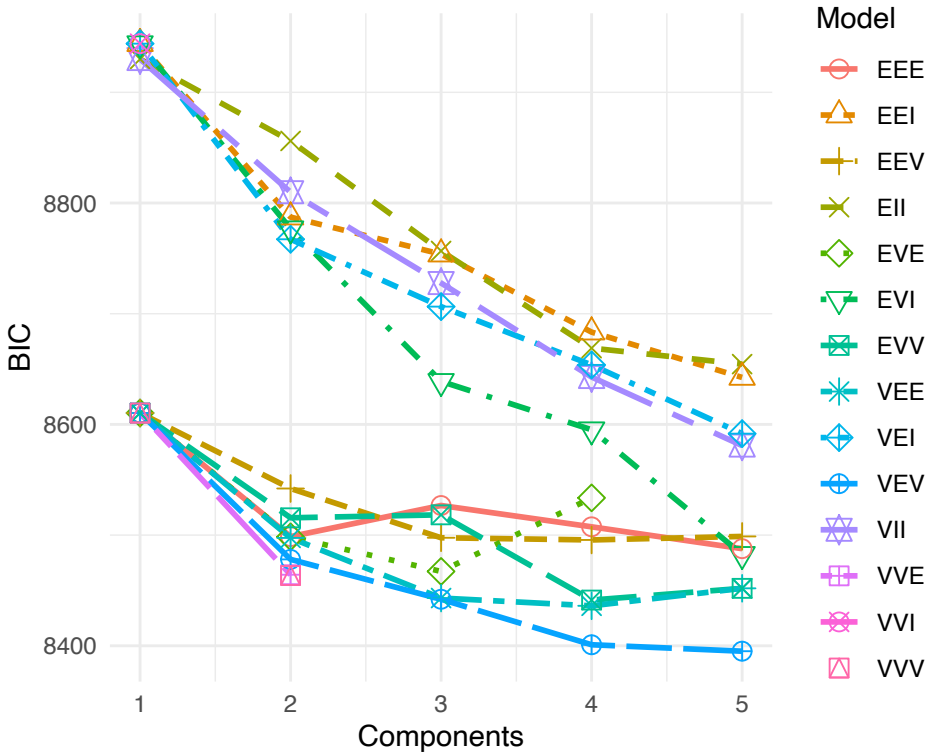


Figure 3.3: Plot of BIC values as a function of model and mixture components

Making a 'BIC' difference Some studies showed several issues concerning cluster algorithms, focusing on the point of finding the optimal number of clusters being unclear and subjective. To overcome this, Zhao, Xu and Fränti (2008) proposed the *BIC difference* or *BICdiff* calculation as an alternative to discover the 'knee point' in a decreasing BIC curve. The idea indicates that the most informational value should be reflected by a combination of both the BIC values and the number of clusters k. Simulation studies demonstrated that the *BICdiff* was better able to detect the correct number of clusters. Thereby, the visualization of the *BICdiff* provides a more

[1]Beware that BIC is calculated with the assumption if the true model is included within the model space. As we allow for covarying components, the BIC values can decrease for a large number of components, while we still have to seek for a meaningful number. Therefore, it is useful to think about a finite dimension before obtaining BIC values in cluserting.

clear interpretation compared to the BIC curves (figure 3.4) where the lowest local minimum of a certain model represents the most optimal and informational number of clusters. By the lack of an implementation of a *BICdiff* calculation in R, a function *BICdiff* (see Appendix B) was self-written, returning a dataframe with the variables *C1*, *Ck*, *k*, *Model*, *C2* and *BICdiff* that are calculated based on the formula steps of Zhao, Xu and Fränti (2008):

1. Calculate ***BIC*** values for each component k and model (see BIC)
2. Calculate the ***normalized BIC*** values resulting in $C1 = (k_{max} - k_{min})(BIC_k - BIC_{min})/(BIC_{max} - BIC_{min})$
3. Calculate the ratio between normalized BIC value and number of clusters to obtain $C_k = C_1/k$
4. Normalize this ratio to obtain $C2 = (k_{max} - k_{min})(Ck - Ck_{min})/(Ck_{max} - Ck_{min})$
5. Obtain the ***BIC difference*** by calculating the absolute difference between $C1$ and $C2$, such that $BICdiff = |C1 - C2|/2$

By visualizing the output, a different story returns compared to the BIC figure 3.3, showing a global pattern of 3 clusters as the most optimal, with the 4-EVE and 5-VEE models as exceptions.

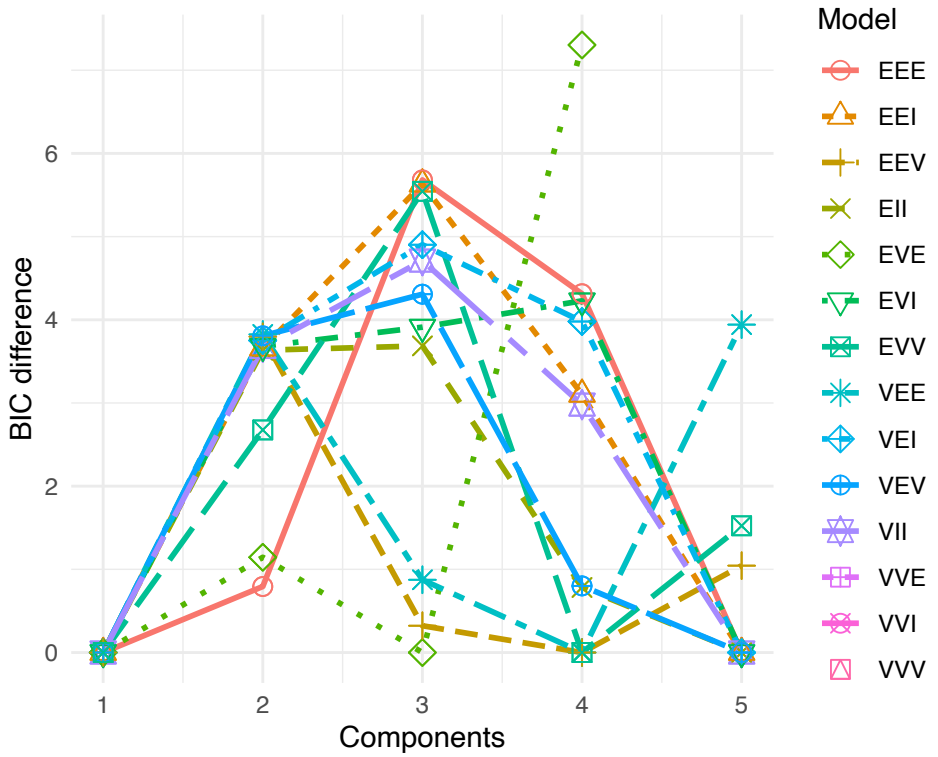


Figure 3.4: Plot of BIC differences as a function of model and mixture components

LRT To test whether the BIC differs for a specified model, one can use a *Likelihood Ratio Test (LRT)*. Herein, the maximum likelihood estimators of two models are compared, thereby speaking about a 'ratio'. The larger this ratio, the more evidence is

found against the null hypothesis stating that no difference can be found between both models'. To become a p-value for the hypothesis testing, a 'resampling' or bootstrapping procedure is used (McLachlan, 1987) in which MLE's obtained from the original data (using the EM algorithm) under H_0 are fitted in a bootstrap sample, doing this a specific number of times. Bootstrapping (Efron, 1979) is a popular method to perform approximations of the sampling distribution of a statistic of interest. Thereby, the *resampling* refers to the definition of bootstrapping, namely taking a number of samples from the original sampling with replacement. In the `mclustBootstrapLRT` function, the dataset, the model we want to test (for instance 'VEV'), the number of bootstrap samples and the maximum number of mixture components has to be specified. The output provides p-values resulting from LRTs. When the LRT is not significant with a p-value $> .05$, the next number of components is indicated not to fit the data better than the current number. As can be noticed in table 3.2, the LRTs shows significance up until 5 components, which is the maximized specified number of components with 999 bootstraps.

Table 3.2: LRT bootstrap results for the VEV model

Components	LRTS	bootstrap p-value
1 vs 2	187.86	0.001
2 vs 3	91.63	0.001
3 vs 4	96.72	0.001
4 vs 5	61.44	0.001

AIC

Close to BIC, the Akaike Information Criterion or *AIC* (Akaike, 1973) is a metric for the goodness of fit (see formula 3.4) with k referring to the number of parameters in the model and L as the likelihood of the model.

$$AIC = 2k - 2log(L) \tag{3.4}$$

Notice that both measures differ in the applied penalty for the number of parameters with $2 * k$ for AIC and $k * log(n)$ for BIC. In other words, BIC penalizes more strongly when the model becomes more complex. Compared to AIC, BIC is argued for selecting the 'true model' with probability 1 as $n \to +\infty$. In defense of AIC, it is argued that the 'true model' is never in the candidates as this equals the 'reality', while in statistics, the basic idea is that 'all models are wrong' (Vrieze, 2012). In in with this idea, AIC select the model that approximate the 'true model' in the most optimal way. It was even shown that the risk of selecting a very bad model was minimized by using AIC, because BIC is sensitive to the sample size.

In the `mclust` package, an automatic calculation of AIC is absent. However, both k and L are provided, by which we can calculate the *AIC* by ourselves for each number of mixture components by self-written *AICmclust* function (see Appendix

C). In figure 3.5, we aim to find the number of components and the model with the minimum level of AIC, again showing the 5-VEV model to have the best fit.

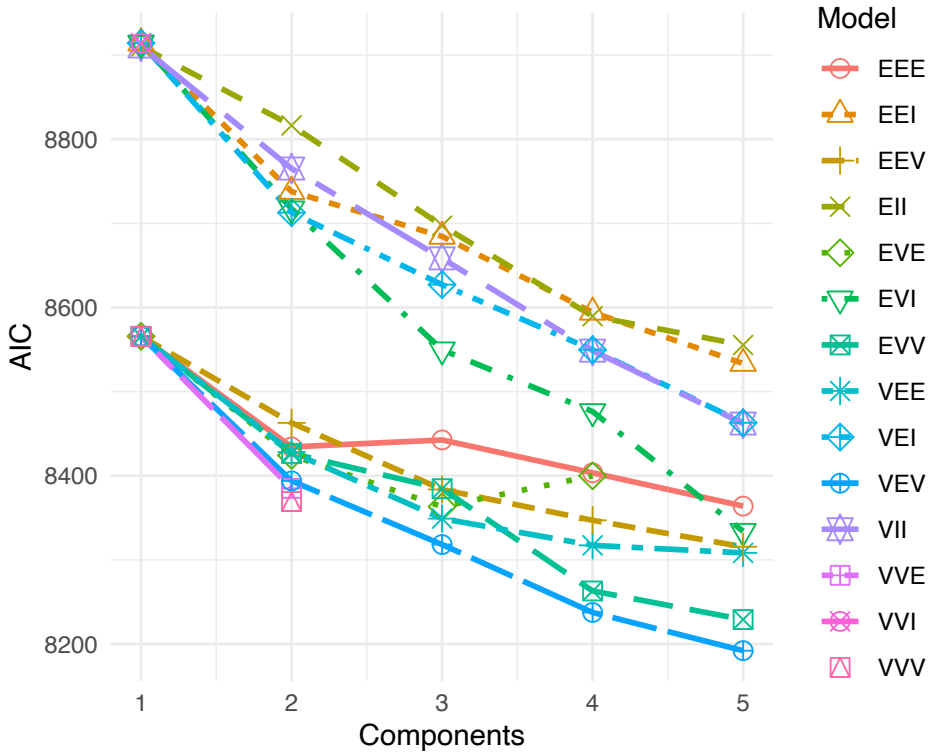


Figure 3.5: Plot of AIC values as a function of model and mixture components

Entropy

Another common-used fit measure is the *entropy*. This refers to a measure of 'disorder' or 'chaos' in the dataset. For instance, when writing this report, my desk knows a high level of entropy. The higher the level of entropy, the more messy the data is. When I would clean my desk and organize stuff, I would decrease the amount of entropy. When we have only 1 profile, the entropy is the least. The measurement of entropy is useful, as the entropy represents a way of overlap between profiles, which is useful as we pursue the goal to find clusters with minimized overlap. In addition, clusters with low entropy would indicate that data points within the clusters are highly similar.

The measurement of entropy is mostly known as fit index in the Mplus software performing LPA. The software calculates a number between 0 and 1, where a value of entropy approaching 1 shows a better partitioning of the components. This calculations refers to the *relative entropy* (RE) as proposed by Celeux and Soromenho (1996). The RE is calculated in three steps. First, the entropy is calculated on the level of components ($\Sigma_{i=1}^{k}(-p * log(p)$; with p for proportions and k for the components) and on the levels of datapoints ($\Sigma_{i=1}^{n}(-p * log(p)$; with p for proportions and n for the sample size). In the second step, the mean entropy for all data points is subtracted from the mean entropy of the classes. This difference is divided by the

entropy of the classes in the last step resulting in the RE. Using this formula, the RE for all components and models was calculated by a self-written *rel_entropy* function (see Appendix C), from which the result is plotted in figure 3.6. Here, we seek for the number of components and the model with the highest relative entropy. In the literature, some authors propose .80 as cut-off to have a good entropy, however, to our knowledge, there is no evidence available in the literature and others argue for the dependence on domain and variables. We mention this to avoid the idea that none of the proposed model is useful, as all relative entropy values are beneath .80. Also, no entropy value is calculated for $k = 1$, as there is no partition of the dataset. According to the figure, the top 3 of models with the best fit are, respectively, 3-EVE, 5-EVI and 4-EVV.

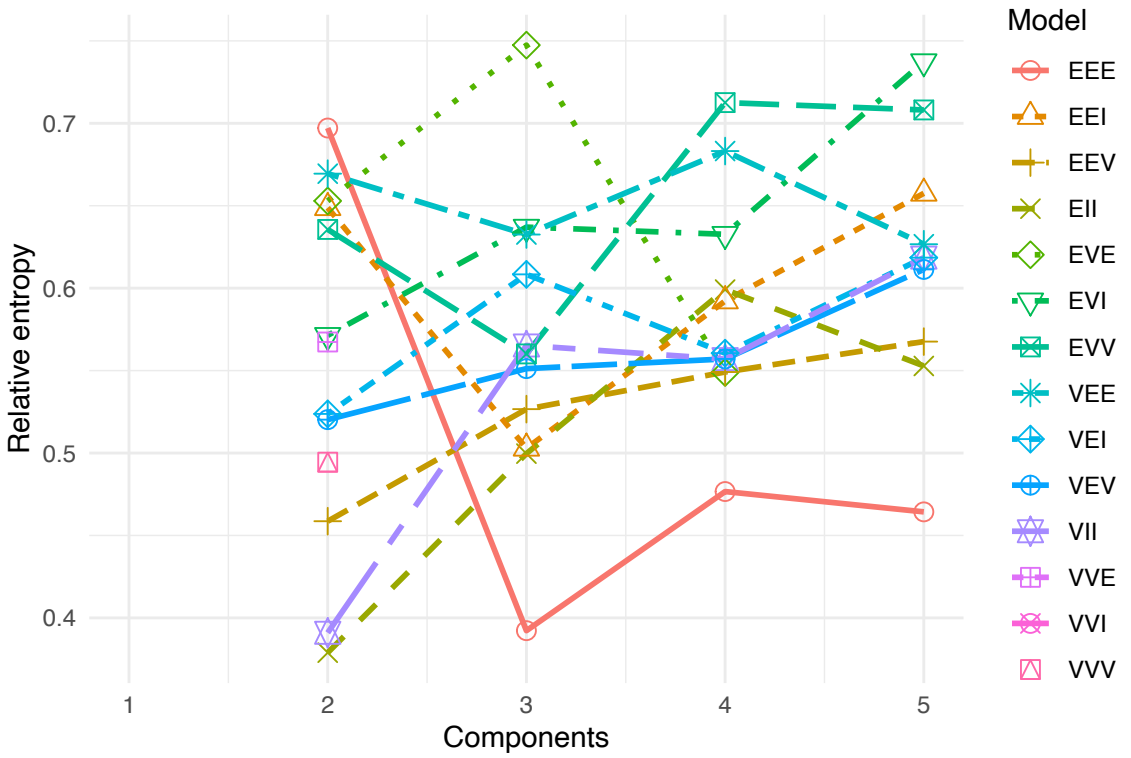


Figure 3.6: The levels of relative entropy by increasing number of components

ICL

The *Integrated Complete-data Likelihood criterion* or *ICL* (Biernacki et al., 2000) is proposed as an important fit measure (McLachlan & Krishnan, 1997) with z_{ik} referring to the conditional probability from mixture component k and $\pi_{ik} = 1$ if the data point i is assigned to component k, otherwise zero.

$$ICL = BIC + 2\Sigma_{i=1}^{n}\Sigma_{k=1}^{k}\pi_{ik}log(z_{ik})$$

As BIC and AIC are more focused on approximating the density, the ICL is more focused on separating groups. More specifically, it includes *an entropy term* measuring

the overlap of clusters (see Entropy), showing a stronger focus on approaching the number of clusters as such.

As seen in the output (figure 3.7), we ignore the solution of choosing 1 cluster as the most optimal number, as we are dealing with finding multiple hidden groups in the data. Here we aim to find the number of components and the model where the ICl is the highest.

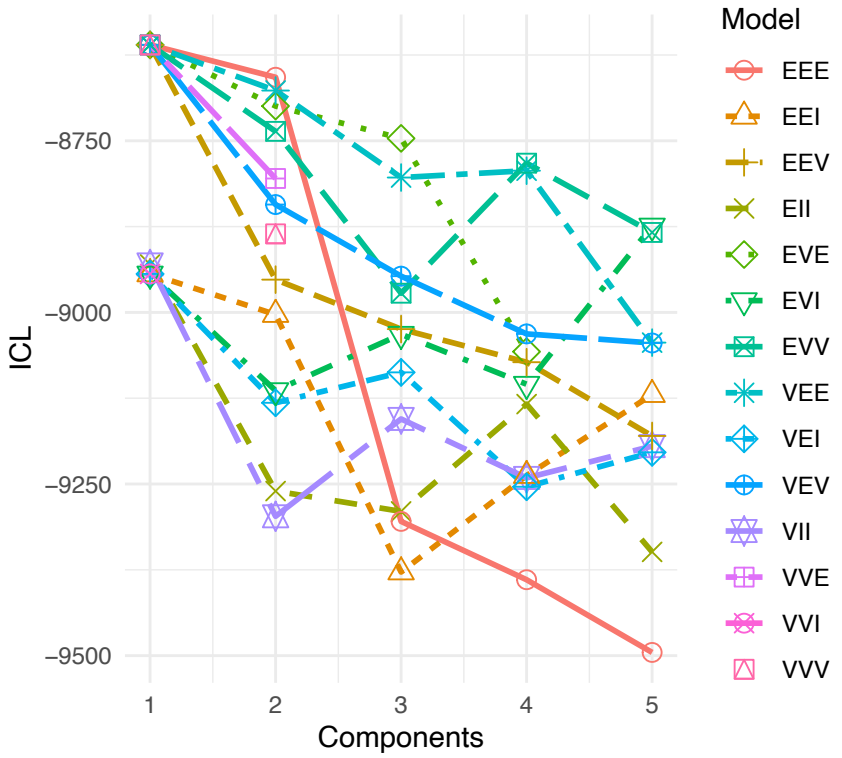


Figure 3.7: Plot of ICL values as a function of model and mixture components

3.2.3 Analytical considerations

LPA refers to the name of GMM in social sciences. By performing GMM to the multivariate `dotoset` dataset, we attempted to estimate the parameters of a series of underlying (latent) mixture components to the global population using the EM-algorithm. To find the most optimal number of mixture components, including a specific model describing geometric features, we performed four different techniques, each calculating a specific fit index of the model on the data. Based on the previous sections, all information regarding the fit of models to the data has been presented in Table 3.4, this in favor of considering the most optimal model. For the sake of interpretation, we need to find the lowest values for BIC and AIC and the highest values for ICL and Entropy. This results in 5 components with a VEV model according to BIC and AIC, a 2-EEE model for ICL and a EVE-model with 3 components for Entropy. As BIC and AIC are closely related fit indices and no direct correspondence can be found with the other fit indices, the results require more consideration.

Table 3.3: Best fitting models for each number of components across fit indices

	BIC values	model	AIC values	model	ICL values	model	Entropy values	model
1	8610.30	EEE	8565.72	EEE	8634.01	EEE	NA	NA
2	8463.48	VVV	8369.38	VVV	8705.17	EEE	0.70	EEE
3	8442.04	VEV	8318.22	VEV	8708.02	EVE	0.75	EVE
4	8400.94	VEV	8237.50	VEV	8760.89	EEV	0.71	EVV
5	8395.13	VEV	8192.06	VEV	8806.88	EEV	0.74	EVI

As can be noticed, the 3-EVE model and the 5-EVI model has both high values for entropy (respectively .75 and .74). Therefore, we consider the other fit indices for these two models, observing *relative differences*. Considering the fit results for a 3-EVE model, the values for BIC and AIC are close to the lowest values (both the third lowest; figure 3.3 and 3.5) with a global knee point for BIC differences at 3 components and for ICL the highest (see figure 3.6). Considering the 5-EVI model, it has the fourth lowest value for BIC and AIC, a knee point located at 4 components and the highest ICL value for 5 components. By reconsidering all fit indices, both models seem to be great candidates. At this point, a researcher could use the theoretical knowledge about the clusters to decide which candidate to choose. However, there are still two techniques to make a more conceived decision. In the first, we check the uncertainty of each model regarding data classification. In the second, we remember the aim of doing clustering analysis and, therefore, check the partitioning of the dataset by each model visually.

1. Model uncertainty

Model uncertainty refers to the mixture weights that has been calculated for each data points for each mixture component. As a result of LPA, each data point has a particular probability of belonging to each of the mixture components. To support the final decision of the researcher, the amount of probability can be checked both in numbers (table 3.5) as in figures (figure 3.8). In calculating the 'uncertainty', the probability of the most likely group for each case is subtracted from 1, followed by sorting the cases by uncertainty and providing a table where the percentage of uncertainty are shown by quantiles with '100%' representing the total uncertainty of a case. By this, the table demonstrates how many cases have a certain level of uncertainty. Figure 3.8 is a visual representation of table 3.5 where the uncertainty is plotted in a continuous line, rather than in quantiles. Noticeably, the 3-EVE model seems to show small little less uncertainty compared to the 5-EVI model.

Instead of evaluating the total uncertainty of models, we can look into more detail by plotting the distributions of mixture weights (i.e. probabilities) for each cluster. To compare, this was done for the 3-EVE model (figure 3.9, upper) and the 5-EVE model (figure 3.9, bottom). The more bimodel a distribution, the better the news. A bimodel distribution indicates that data points with greater than 0.50 probability will be aligned to a given cluster and that the other data points have a really low

Table 3.4: Uncertainty for components in quantiles

Components	0%	25%	50%	75%	100%
3 EVE	0.00	0.04	0.17	0.34	0.63
5 EVI	0.00	0.05	0.24	0.40	0.68

probability of belonging to that particular cluster. Looking at figure 3.9, all clusters show a great pattern in the distribution. For both models, cluster 3 show more cases with moderately levels of probability, which will indicate that most of the data points will be overlapping with other clusters.

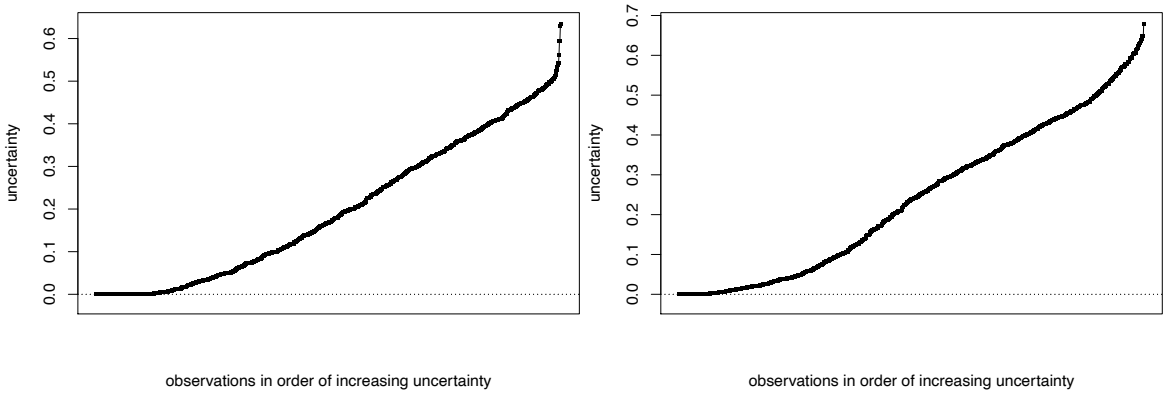


Figure 3.8: Plots showing level of uncertainty

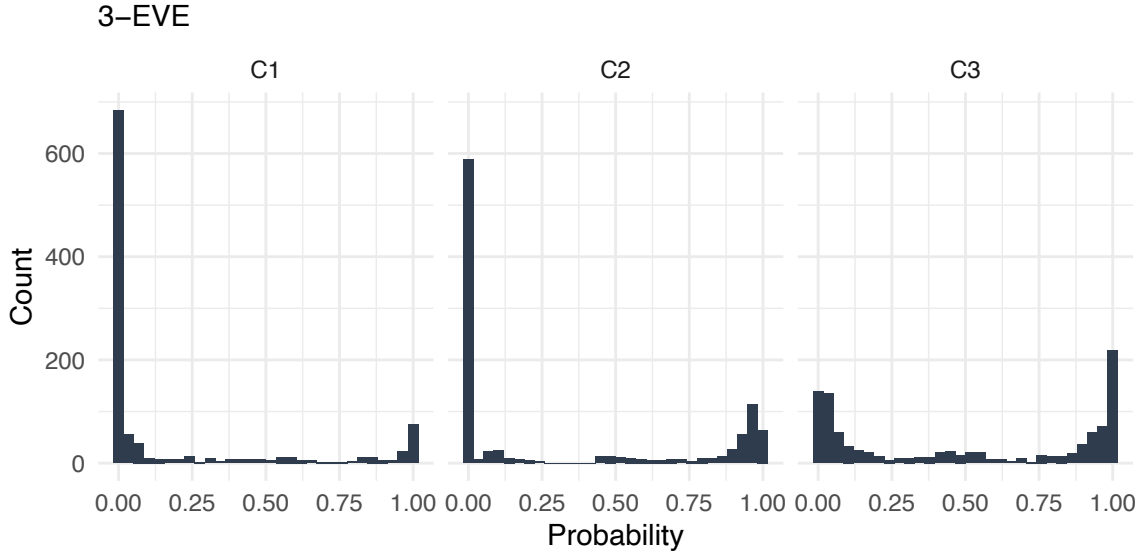


Figure 3.9: Results of the 3-VEV model: probabilities for each cluster

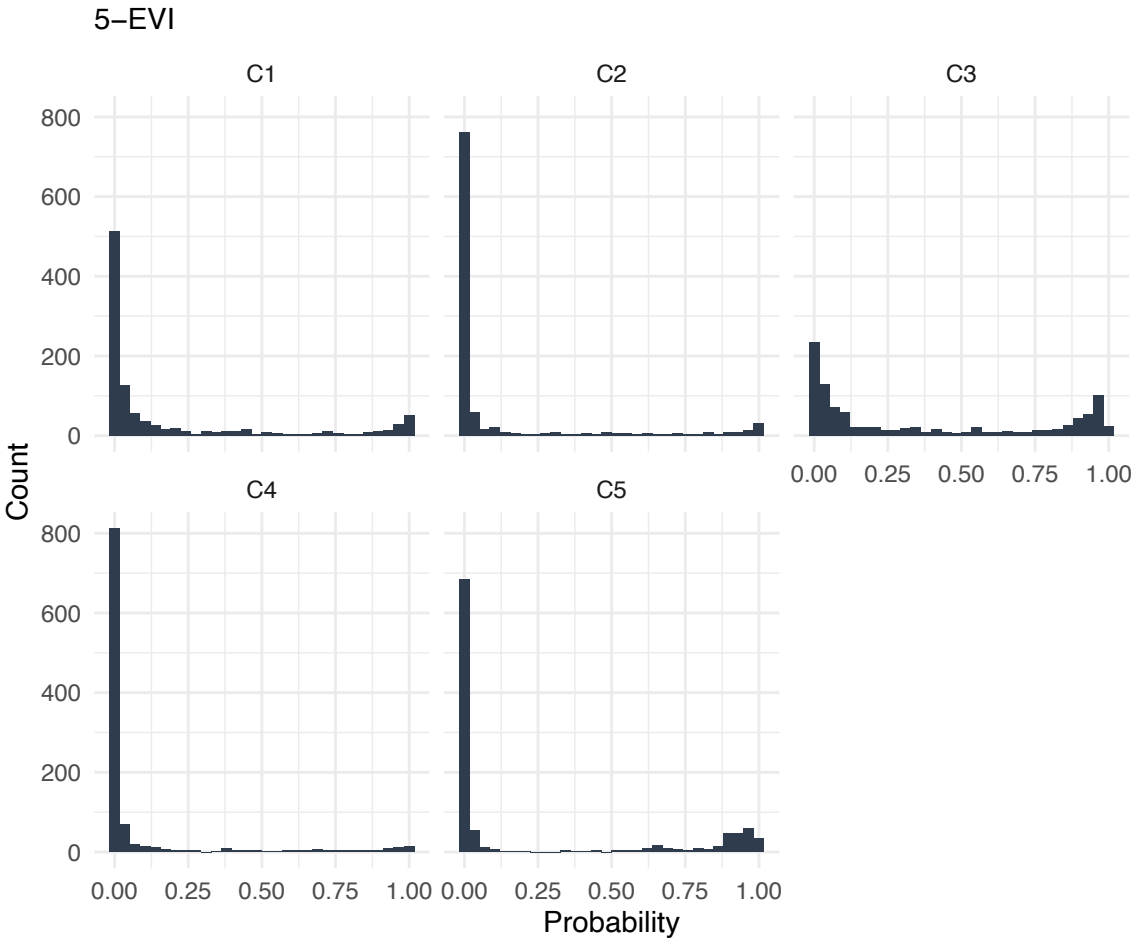


Figure 3.10: Results of the 5-EVI model: probabilities for each cluster

2. Cluster overlap

Similarly to the procedure of section Cluster overlap regarding K-Means clustering, we can plot a range of components in terms of their data coverage. By plotting the 'data surfaces' using a biplot, we get an indication of the data partitioning given a certain model. On the left, the mixture components are plotted, showing ellipsoidal components with variable volumes, shapes and equal orientations for the 3-EVE model and diagonal components with equal volume, variable shapes and coordinate axes for the 5-EVI model. On the right, the data surfaces are colored based on the data points and the assigned component based on the highest mixture weight. For the 3-EVI model, cluster 3 shows overlap with the other clusters, a finding that was also shown in figure 3.9. For the 5-EVE model, more overlap is found between clusters where cluster 3 is almost covered in total by clusters 1 and 4.

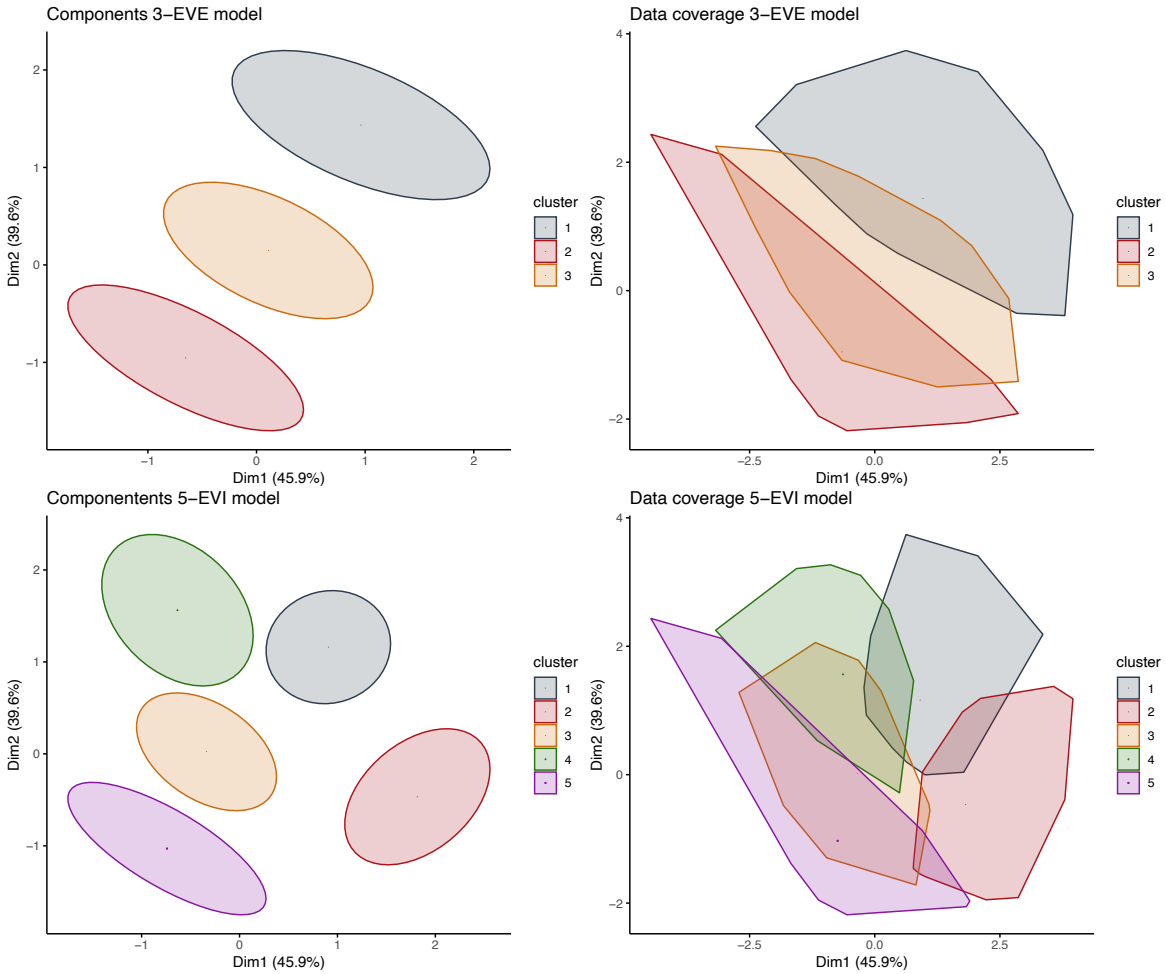


Figure 3.11: Visual comparison between the 3-EVE and the 5-EVI model

3.3 Model characteristics

By the procedure of model selection and consideration, we propose the 3-EVE model as the most optimal model given the theoretical knowledge, the current dataset and the LPA procedure. Equivalent to the Chapter 2, there are several ways to describe the characteristics of each mixture component in terms of the study variables. First, Figure 3.11 shows the result of the LPA procedure by plotting the density curves (in 2D, from top view) in a multivariate matrix. By comparing the density curves (figures 1.4 - 1.6) to the result in figure 3.12, it can be noticed that the LPA included the skewed distribution of amotivation (red).

Next, a barplot was constructed to show the characteristics of the clusters in terms of the standardized study variables (figure 3.12). Cluster 1 could refer to the *good quality* motivation group including participants scoring low on controlled motivation and very low on amotivation. In total, 34% of the cases were assigned to this group. Cluster 2, as the smallest one with 17% of the participants, seems to have high scores

on amotivation, moderately on controlled motivation and no high or low score on autonomous motivation, indicating a *bad quality* motivation group. Finally, cluster 3, wherein 49% of the participants were assigned to, shows a no high or low scores for all types of motivation, referring to a *low* motivation group.

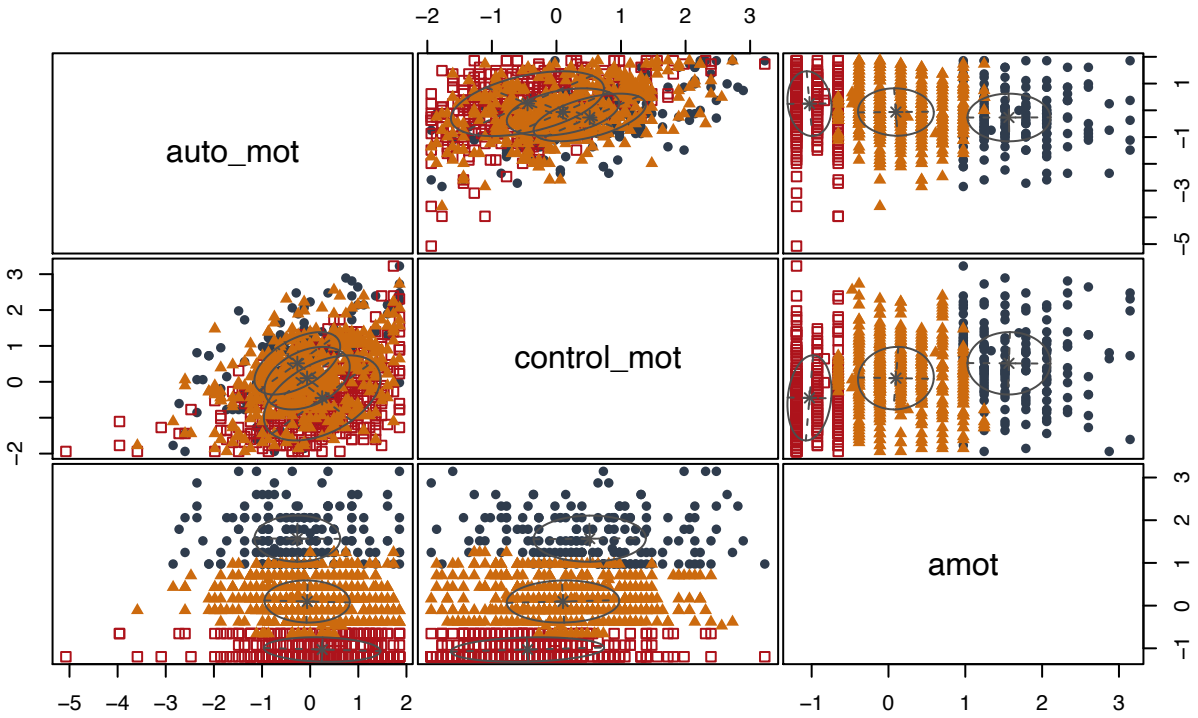


Figure 3.12: 2D-density plot as a result of LPA

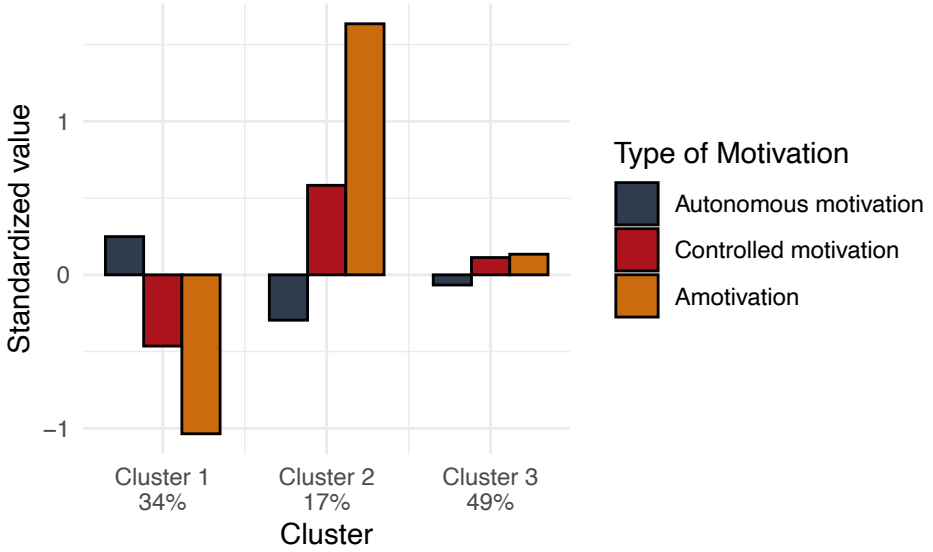


Figure 3.13: Model characteristics by LPA in a barplot for each curve and study variable

3.3.1 Between-cluster differences

Similarly to the procedure of the K-Means clustering, we test clusters differences in terms of the (unstandardized) study variables by performing a Multivariate Analysis Of Variance (ANOVA), resulting in significant cluster differences (Wilks's lambda = .16, $F(6, 2082) = 526.57$, $p < .001$). Exploring the univariate test, we observe significant differences between clusters for each study variable, showing that the *Good motivation* profile scores significantly higher for autonomous motivation compared to the other profiles ($R^2 = .038$; table 3.6). In reverse, table 3.7 shows significantly higher scores for controlled motivation of both the *Bad* and *Low* motivation group compared to the *Good motivation* ($R^2 = .137$). At last, all profiles score significantly higher on amotivation compared to the *Good motivation* profile ($R^2 = .83$; table 3.8).

Table 3.5: LPA clusters in the prediction of autonomous motivation

	term	estimate	std.error	statistic	p.value
1	(Intercept)	4.19	0.03	158.26	<.001
2	LPA_Bad	-0.28	0.05	-6.05	<.001
3	LPA_Low	-0.16	0.03	-4.64	<.001

Table 3.6: LPA clusters in the prediction of controlled motivation

	term	estimate	std.error	statistic	p.value
1	(Intercept)	2.11	0.04	56.60	<.001
2	LPA_Bad	0.79	0.06	12.29	<.001
3	LPA_Low	0.43	0.05	8.96	<.001

Table 3.7: LPA clusters in the prediction of amotivation

	term	estimate	std.error	statistic	p.value
1	(Intercept)	1.15	0.02	56.59	<.001
2	LPA_Bad	2.46	0.03	70.59	<.001
3	LPA_Low	1.08	0.03	40.94	<.001

Chapter 4

Discussion: a Considerative Comparison

Studies arguing for the idea of *"It is not the quantity of motivation that matters"* has been increasing in the last decade. Contributing to the literature of punishing, rewarding and conditioning (Skinner, 1938), the SDT presents a qualitative perspective on people's motivation (Ryan & Deci, 2017). As a continuum to what extent one's motivation is 'internalized' or 'is incorporated in his/her internal values', multiple subtypes of extrinsic motivation has been proposed. While someone is curious and enthusiastic to go for a run (intrinsic motivation), someone else might think it is important and personally valuable to do (identified motivation). Both types refer to *autonomous motivation*. On the other side, a person could go for a run to avoid feelings of guilt and shame (introjected motivation) or even to satisfy the expectations of another (external motivation), both describing the types of *controlled motivation*. At last, one could have no idea to go for a run, thinking it is a waste of time and, thereby, begin *amotivated*.

More than one decade ago, Vansteenkiste and colleagues (2009) proposed a new perspective on the measurements of motivation. In the results of the study, they attempted to look for combinations, such that people could have a *good qualitative* motivational profile (scoring high on autonomous motivation, low on controlled motivation and amotivation), a *strong quantitative* motivation profile (scoring high on both autonomous and controlled motivation and low on amotivation), a *low quantitative* motivational profile (scoring low on both types of motivation and high on amotivation), a *bad qualitative* motivational profile (scoring low on autonomous motivation, scoring high on controlled motivation and moderately on amotivation) or even an *amotivated* profile (scoring high on amotivation and low on autonomous and controlled motivation). By using cluster analysis to analyze these profiles, individuals assigned to such a specific group, showing that, for instance, *good motivated* students had significantly higher scored on learning performance and engagement in school, while *bad motivated* students had the highest scores for pressure and even anxiety (Vansteenkiste et al., 2009). Nevertheless, it was noticed that many papers in the upcoming years started using different techniques to uncover 'typical features' in the population regarding motivation. This started to be an issue, as different results with

the same measurements were found between studies and, moreover, different results with other techniques were found within the same study. As a matter of fact, multiple authors did not describe extensively the cluster procedure they had used and how they made decisions.

As the literature of SDT is still increasing exponentially, the current report argued for a moment to stop and think about which cluster technique to use, for which reasons, how to do it and how to report is. Therefore, we discussed the most commonly-used techniques in motivation literature, namely the *K-Means cluster analysis* (Chapter 2) and the *Latent Profile Analysis* (Chapter 3). In these chapters, we did calculations, explanations and interpretations of the analyses, using the `dotoset` dataset (Chapter 1). In the latter chapter, we discussed this dataset in terms of descriptive statistics, standardization and outlier detection. In this final chapter, we take a moment of reflection to draw up a list of important considerations regarding both types of cluster analysis. Before starting, we first summarize the steps for both cluster analyses:

Steps of (Hierarchical) K-Means clustering:

1. Calculate distance matrix by the Euclidean distance
2. Choose and validate linkage method for Hierarchical clustering
3. Choose and validate K-Means clustering algorithm
4. Perform clustering tendency of the dataset
5. Validate and determine number of clusters
6. Check cluster characteristics

Steps of Latent Profile Analysis:

1. Check density curves of the dataset
2. Perform mixture modelling
3. Analyze multiple fit indices
4. Consider output extensively
5. Check cluster characteristics

4.1 Comparing the final results

1. Visually

Figure 4.1 shows barplots for the results of both the K-Means clustering (left) and the Latent Profile Analysis (right) in terms of the study variables. This figure is shown as we can compare the final results. Based on a list of validation techniques and fit indices, we made extensively considerations about the number of clusters to gain and which covariance model to use in LPA. In both methods, we concluded three components. Observing figure 4.1, the clusters seem to be comparably in terms of standardized scores of the study variables, showing cluster one as a *good motivation*

clusters (scoring high on autonomous motivation, on average/low on controlled motivation and low on amotivation), cluster 2 as a *bad motivation* cluster (scoring low on autonomous motivation, high on controlled motivation and very high on amotivation) and cluster 3 as a *low motivation* group (scoring on average/low on all types of motivation).

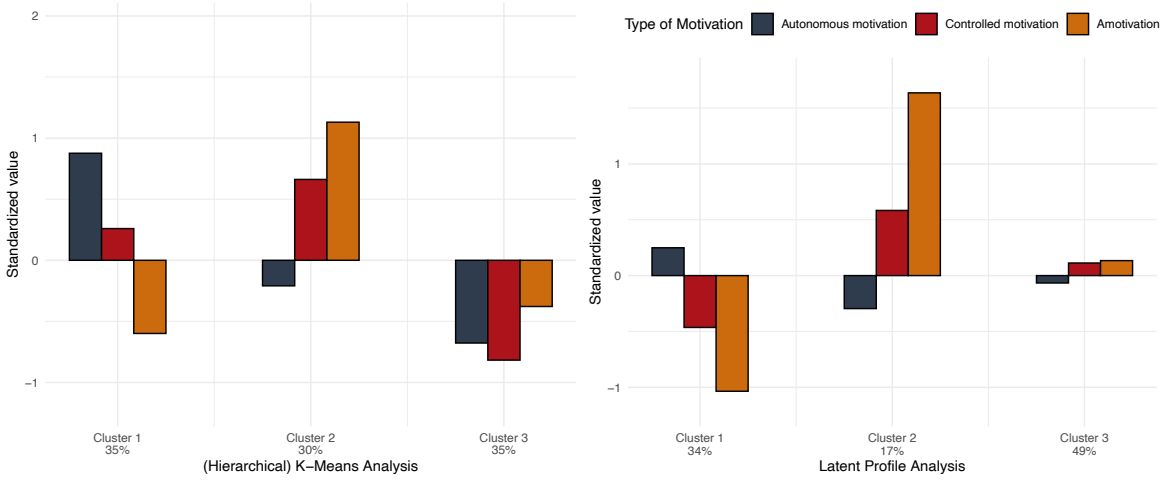


Figure 4.1: Final results K-Means clustering and LPA

2. Contingency table

Still, both results differ in terms of the sizes of the clusters. The K-Means clustering seem to have more equal-sized clusters (30%-35%), while LPA shows more unequal groups (17% - 49%). To compare the assignment of data points to clusters, contingency tables were constructed in table 4.1. Herein, we would expect to find the largest numbers on the diagonal of the table, indicating correspondency between both categorical cluster membership variable for K-Means clustering and LPA clustering. This is a finding we can observe, with a strong correspondence of 96% of the data points between the *bad motivated* LPA cluster and the K-Means cluster. In detail, this means that 96% of the data points that has been assigned to the *bad motivation* group by LPA can be found in the *bad motivation* K-Means cluster.

Further, it can be noticed that none of the *good motivated* cases has been assigned to the *bad motivation* group by one of the methods. This is interesting, because this shows support for a typical patterns in the data, namely a profile of being good or bad motivated without overlap. For both methods, the differences are situated in the the assignment of *good motivated* data points to the *low motivated* group, the assigment of *bad motivated* data points to the *low motivated* group and, in reverse, the assignment of *low motivated* data points to the *bad motivated* or *good motivated* group. Clearly, one's assignment depends on the clustering method.

Table 4.1: Contingency table for rows (left) and columns (right)

		LPA					K-Means		
		Good	Bad	Low			Good	Bad	Low
	Good	0.53	0	0.47		Good	0.54	0	0.33
K-Means	Bad	0	0.55	0.45	K-Means	Bad	0	0.96	0.28
	Low	0.43	0.02	0.55		Low	0.46	0.04	0.39

3. Chi-squared

Analyzing these frequencies of the contingency table using a Chi-Squared test, a significant result was found at the p-value of .001 ($\chi^2(4) = 526.55, p < .001$), by which we reject the null hypothesis arguing for independence between the two variables. This indicates that both variables are statistically associated and thereby, extracting comparable clusters in terms of 'typical features' in the dataset.

4. Cohen's kappa

Cohen's kappa (Cohen, 1968) could be used to measure the agreement between both cluster assignment variables. Be aware that both variables has to be categorical, include the same levels and each data points requires a value for each variable. Additionally to representing only the percentages of agreement, Cohen's kappa corrects for the possibility that data points were assigned to a particular level 'at random'. The result of the kappa index is a number between 0 and 1, representing to what extent the agreement is based on chance. The `Kappa()` function of the `vcd` package provides both unweighted (nominal or ordinal variables) and weighted (only ordinal variables) values for kappa with a p-value testing the H_0 stating an agreement by chance. Because the clusters assignment is a nominal variable, Cohen's kappa (k) = .30 with p-value lower than .001, which represents a fair strength of agreement that is significantly different from zero (see for classification interpretation: Fleiss et al., 2003).

5. Biplot

A final approach to compare the result of both cluster techniques is performing a (Simple) Correspondence Analysis (CA). CA (categorical data) is equivalent to Principal Component Analysis (continuous data) and approaches the distance between categorical levels in a geometrical way. It determines the distance between levels by their positions in a low-dimensional space and provides a clear way of interpretation. A *symmetrical biplot* is shown in figure 4.2 where blue points refer to rows (K-means assignment) and red triangles to columns (LPA assignment). This graphical visualization represents an easy way to interpret how similar clusters assignments are from both techniques in a global way. The strongest similarity is shown for the *Good motivation* cluster, followed by the *low motivated* cluster and the *bad motivated* cluster.

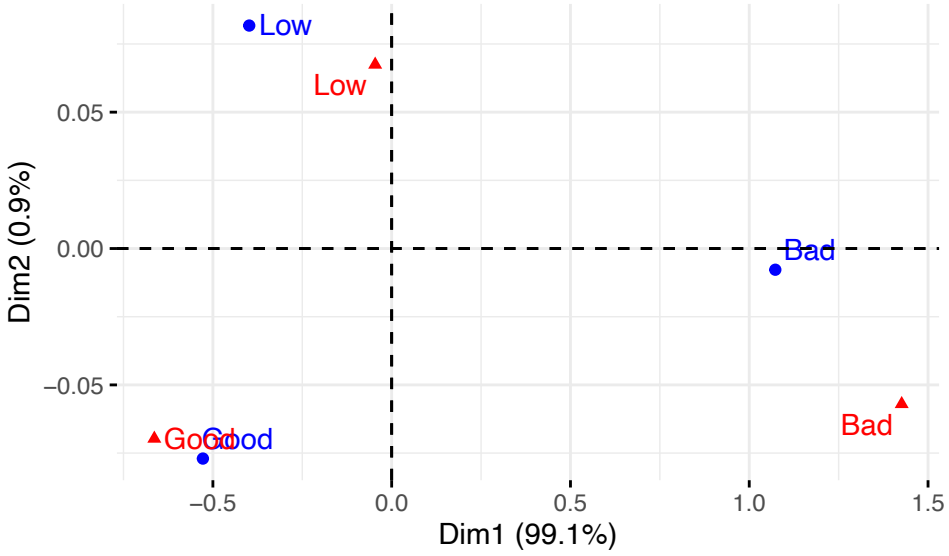


Figure 4.2: Symmetric CA biplot

4.2 Combine knowledge and data

Before performing analyses seeking for 'typical patterns' in the continuous dataset, a researcher is demanded to think and to consider both the expected clusters and which variables are included. Like each hypothesis that is constructed in science, a researcher needs to theoretically state the types of classifications that are expected in the dataset. This is done for two reasons. In addition, the researcher needs to consider cluster analysis a priori in a more analytical way, supporting which clustering method will be used.

A cluster-range hypothesis

First, the researcher has to construct a *'cluster-range' hypothesis*. Before performing cluster analysis, it is recommended to consider about the range of clusters that will be analyzed. In terms of the current report, four motivational profiles with the same contents were found over different studies, namely the *good* motivated, the *bad* motivated, the *highly* motivated, the *lowly* motivated and the *amotivated* profiles. Therefore, a researcher could propose the aim to seek for 5 clusters equaling these motivational profiles from the literature. Nevertheless, the researcher cancels out the evidence of partitioning the data into these clusters in a statistically valid way. By describing a *range* of clusters in advance that is going to be investigated, the researcher allows both the theoretical considerations and the statistical 'space' of testing the most optimal number to the given dataset. A constructed and well-reasoned hypothesis by the literature could be a good way to limit the number of tested components, this in favor of reaching the optimal goal of cluster analysis: generating interesting and statistically meaningful clusters.

'Cluster-hacking'

Another reason to theoretically consider a priori a range of components is the avoidance of *'cluster-hacking'*. Coming from the understanding of 'p-hacking', a researcher could have the tendency only to work with the number of components showing the most suitable partition of the dataset, while another number of components was expected. As we demonstrated in Chapter 2 and Chapter 3, some of the clustering validation techniques requires a certain level of interpretation of the researcher. Nevertheless, it is believed that this subjective flexibility of interpretations might be a fundamental influence in reporting cluster results (Lo Siou, et al., 2010). For instance in visual techniques as the Elbow method for K-Means clustering and the BIC curve for LPA, there are no clear cut-offs to determine the optimal number of clusters. Moreover, the researcher is expected to decide to what extent the WSS stops to decrease or the relative entropy does not increase fundamentally. In some cases, these techniques are only used as arguments to determine a specific number of clusters, however the information is not sufficient valuable. Here, it is required to report as much information a researcher could gain, this to inform how decisions were made both before, during and after clustering. Informing the reader about the whole process is demanded, as the risk exists the researcher chooses an inaccurate of uninformative number of clusters, just to be in line with the hypothesis.

4.3 Knowing the data before clustering

Apart from being a statistician, an employer, an employee, a student or a researcher, everyone who works with a dataset should know the data. Which variables are included and what are the descriptive statistics of these variables (e.g. mean and standard deviation)? How are these variables associated (correlations)? Do they have a good *internal consistency* (see section Measurements)? Are variables comparable before starting the cluster analyses? The latter questions refers to the *standardization* of the variable. By doing this, their variance is comparable by which they can be treated with equal importance in the cluster analysis. Another important question, which we did not handled extensively in the current report, is how many missing data is in the dataset and how these should be handled. However there are more questions to be asked regarding statistical analyses, we handled the most important ones in preparation of cluster analysis.

Before starting cluster analysis, it is fundamental to check to what extent the study variables are normal distributed (section Descriptives) and how many *outliers* are detected (section Outliers). This is important, especially in K-Means clustering, as algorithms are non-robust as the centroid of each cluster is based on the mean of that cluster. The mean is a strongly non-robust parameter, while the median is more robust. Therefore, we performed the *Median Absolute Deviation* procedure. Also, LPA (specifically GMM) is sensitive to outliers as it attempts to estimate the mean as one of the parameters. As mentioned in Chapter 1, we saved the `df.sc` dataset, still including the outliers. To demonstrate, figure 4.3 shows the resulting biplots of

the final Hierarchical K-Means clustering (left) and LPA models (right), based on a data set with and without outliers. We can notice the clear differences between the ending results, possibly generated by extreme values in the dataset affecting parameter estimations. For the sake of transparancy, we only performed the final models to generate these plots, while it is required to question to what extent the fit indices would point to this number of clusters taking the outliers into account.

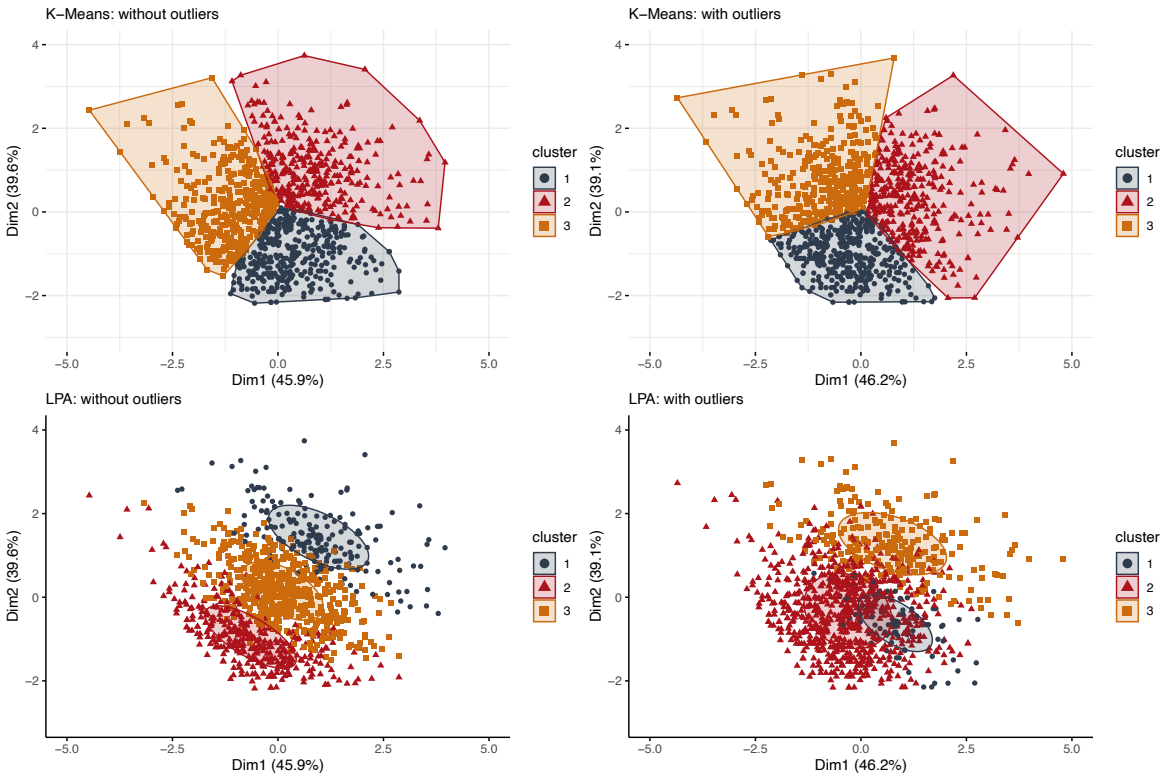


Figure 4.3: K-Means (top) and LPA (bottom) results with and without outliers

4.4 Deciding the type of cluster analysis

As the core of the current report, we described, demonstrated and discussed the two most commonly-used cluster analysis within the motivation literature. While they have the same goal - seeking for hidden groups in the dataset -, both analyses showed fundamental differences. However, it is hard to state which one is the best and the researcher is asked to be informative about the choice which cluster method will be used. In doing this, we discuss several difference between both methods.

4.4.1 No distribution versus distribution

K-Means clustering only incorporates the mean of the variables, while LPA incorporates both the mean and the variances. Currently, we focused on the procedure of

Gaussian Mixed Modelling, referring to the application of LPA (mixture modelling) to the Gaussian family. When a researcher would assume the underlying mixture components do have a Poisson, rather than a Gaussian distribution, other parameters would be included.

4.4.2 Flexibility versus no flexibility

In line with the difference in parameters, LPA allows for geometric flexibility, while K-Means clustering is not flexible in the shape of clusters. To put it simply, K-Means clustering draws a circle around the calculated centroid of the cluster with a radius of the maximum Euclidean distance, by which all data points are assigned to one of the closest cluster. Whenever the data has different shapes, the same procedure will be applied by which important information of the data might be lost. In contrast, LPA includes a specific within-covariance matrix model, by which the shape of the cluster can fit the data in the most optimal way. In a practical way, the EM-algorithm is applied to digital face recognition in Machine Learning as it is able to identify a series of data points as 'a mouth' or, in line with circular clusters, two separated eyes. For the reason of geometric flexibility in LPA, one might think that K-Means clustering is a special case of LPA, especially the cases where geometric flexibility is at a minimum level. In reality, this might be true as K-Means clustering refers to the covariance model with spherical distributions with equal volumes and equal shapes (i.e. 'EII' in the `mclust` package). Figure 4.4 shows the result of the K-Means clustering procedure (Chapter 2) and the performance of a GMM with 3-EII model. Importantly, there is no 1:1 correspondence as both techniques are based on different calculations, however the visual similarities can be noticed.

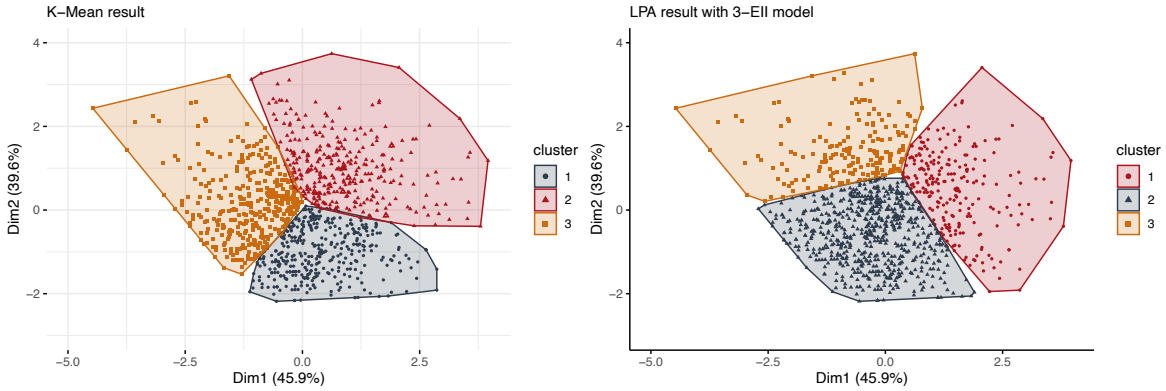


Figure 4.4: K-Mean and Latent Profile Similarities

4.4.3 Statistical whitening in K-Means clustering

In addition to point 2, mixture modelling is useful for clusters/components with varying variances and varying sizes, while K-Means cluster analysis assumes identity covariance structures (spherical clusters) and equal cluster sizes. In other words, it enforces statistical independence to the clusters, also called *statistical whitening*. When

the components are expected to correlate, LPA seems useful and an optimization of K-Means clustering. Nevertheless, statistical whitening brings several benefits. In K-Means clustering, this is done by reorienting the data by the two biggest (orthogonal) axes coming from PCA, by which statistical independence is generated. The statistical whitening or the 'decorrelation' of components can be useful for several reasons:

1. Easier interpretation

The results of non-orthogonal models, like in LPA, are more complicated to interpret. As we saw significant correlations between the study variables regarding motivation (see figure 1.7), the results of LPA should be interpreted more cautiously. For instance, the level of uncertainty has to be taken into account, while the K-means clustering implements a 'hard' assignment (see section Being certain versus being uncertain). This could increase the clarity of the results.

2. Multicollinearity

Incorporating the covariances between clusters brings the alertness of researchers in upcoming analyses. Mostly, the latent variable 'cluster membership' is used as a predictor for a series of outcomes. Accounting for the covariances results in a certain level of multicollinearity. This is the case when two or more variables in a regression model are linearly related. This might affect the standard errors (overestimation), type II error (false negatives), overfitting of the models and issues in parameter estimation.

3. Complete partition

Enforcing the components to be 'non-correlated' is beneficial for further analyses. For instance, when the factor variable 'cluster membership' is in a model based on K-Means clustering, the researcher is able to clearly identify the unique differences of the clusters compared towards each other. Incorporating the covariance between clusters would indicate that the analyses are not able to separate effects.

4.4.4 Size of the data

It has been shown that the sample size might affect the performance of cluster analysis (e.g. Vermunt, 2011). For instance when having a small sample size, adjusted fit indices (correcting for the sample size) for BIC (i.e. adjusted BIC) or AIC (i.e. AIC of the second order) can be used in LPA. However, when having a small sample size, meaning that the number of parameters and the number of cases are strongly out of balance, it might be beneficial to start with a simple covariance structure as the performance decreases when going too complex (Vermunt, 2011). Therefore, the K-Means cluster procedure might provide interesting and more valid results at first.

4.4.5 Model-free

While LPA provides the benefits of geometric flexibility in clustering, it also has the limitation of being dependent of an underlying model. In the current report, we focused on Gaussian Mixed Modelling which assumes multivariate normality within mixture components (Lee & McLachlan, 2013). By this, stronger assumptions are made about data properties, like normally distributed variables. On the other hand, it allows for model testing and accounting for uncertainty, while K-Mean clustering is model-free and, sometimes, even ends up in distorted results including these measurement errors.

4.4.6 Being certain versus being uncertain

K-Means clustering can be called the 'model-free clustering with hard assignment'. Based on the EM-algorithm, LPA results in a probability (i.e. mixture weight) for each case to what extent it belongs to each cluster. The latent variable 'profile' will be completed by the cluster with the highest probability within each case. This is absent in K-Mean clustering where every data point is assigned to one cluster. In this comparison, we need to remind that this 'hard assignment' is the core feature of K-Means clustering algorithms. The goal of the algorithm is to update each data point in terms of its distance to each centroid, while LPA estimates parameter distributions resulting and accounting for overlap of clusters based on the parametric models.

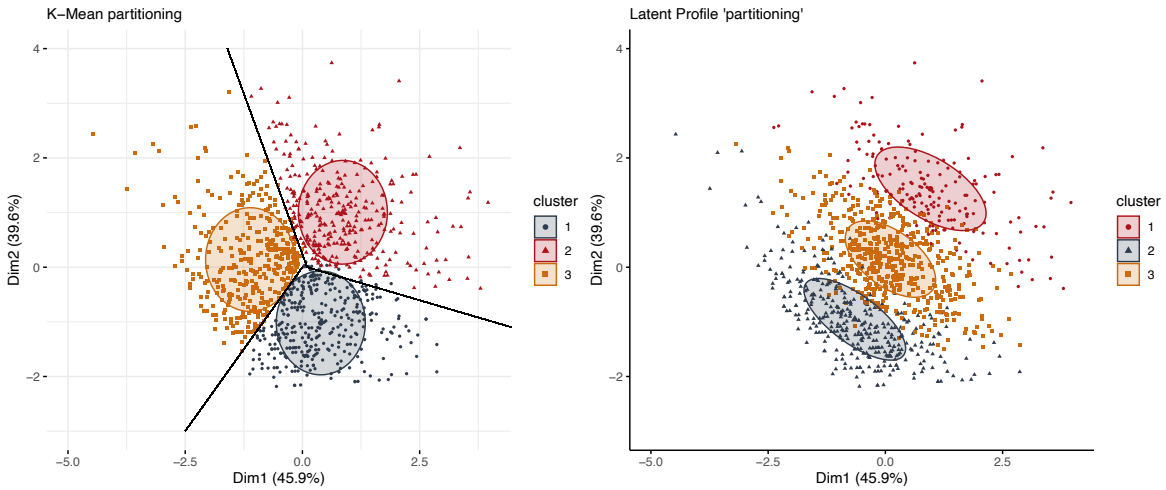


Figure 4.5: K-Mean and Latent Profile partitioning

4.4.7 Balanced versus unbalanced design

It is states that K-Means clustering has a 'bias' to create clusters of equal sizes. For instance, looking to point Size of the data, we could reason that we constrain the procedure to have components of equal sizes. Nevertheless, this is not useful when it is thought that the dataset contains clusters that are not comparable in size (Symons, 1981). In that case, it is recommended to use LPA and check for covariance constraints

with varying sizes. In the report, we did not mention the size of the component as technique to choose the optimal number of components. This is done because - to our knowledge - there is no analytical reasons available in the literature.

However, the difference in cluster sizedscan be an issue anticipating further analyses. Specifically, there are multiple tests contributing on the idea of having *a balanced design* (i.e. all levels of the predictor are equally sizes). A balanced design can be beneficial, as statistical tests (e.g. ANOVA) will have large statistical power and the tests are less sensitive to violations of the homoscedasticity assumptions (i.e. having equal variances). This could result in no reliable and misleading results with, for example, increased Type I error (false positive) showing highly significant results, especially in small sample sizes.

In line with point 3.4, many statistical tests including parameter estimation rely on the Central Limit Theorem (CLT), stating that the sampling distribution of a parameter, like the mean, will approximate a normal distribution, given a sufficiently large sample size. Knowing the implication of CLT on statistical inference (in terms of power and levels of significance), results changes with sample size. This points to the risk of having small groups using LPA, while a balanced design with sufficiently large clusters resulting from K-Means clustering are favorable in this perspective.

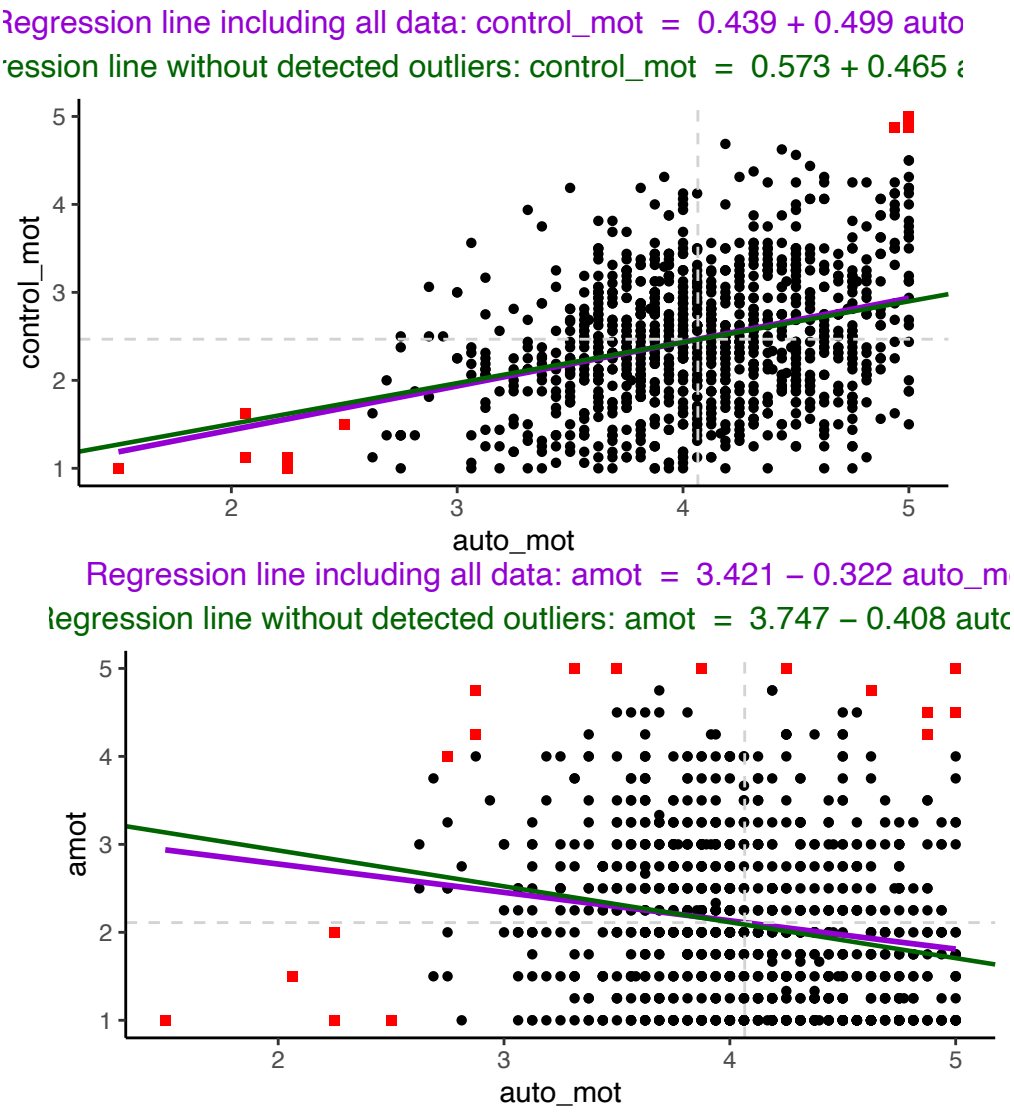
4.5 Conclusion

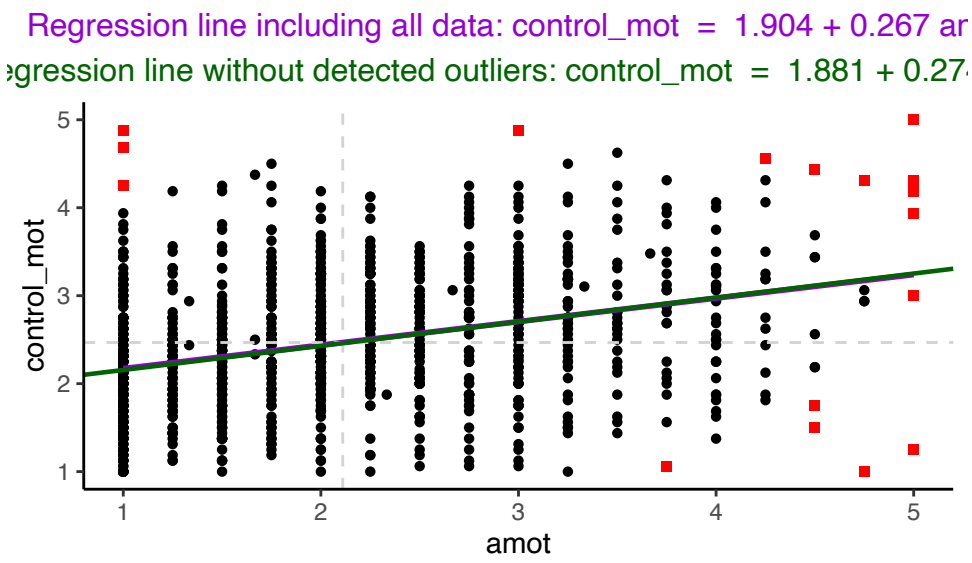
In the literature of motivation, there is an increasing interest in assessing motivational groups. Differentiating between quantitative and qualitative motivational profiles, multiple studies showed substantial differences between groups in terms of outcomes, which supports the predictive validation of the person-centered approach. Performing both K-Means cluster analysis and Latent Profile Analysis, the literature required a moment of reflection as many authors were using different types of analyses without considering the reasons to choose the one over the other. In the current report, we attempted to provide a brief overview of both cluster techniques, discussing the content and the implementation of both methods in R. Using a set of validation techniques, we concluded 3 clusters, both pointing to a *good*-qualitative profile (scoring high on autonomous motivation), a *bad*-qualitative profile (scoring high on controlled motivation and amotivation) and a *low*-quantitative motivational profile (scoring low on all types of motivation). By discussing a list of important considerations, we hope it could inspire researchers applying both techniques in future studies. Additionally to the theoretical and analytical considerations, we provided a practical R-based tutorial for both cluster techniques.

Appendix A

Additional Figures

A.1 Multivariate outliers





A.2 Geometric overview of mixture models in `mclust`

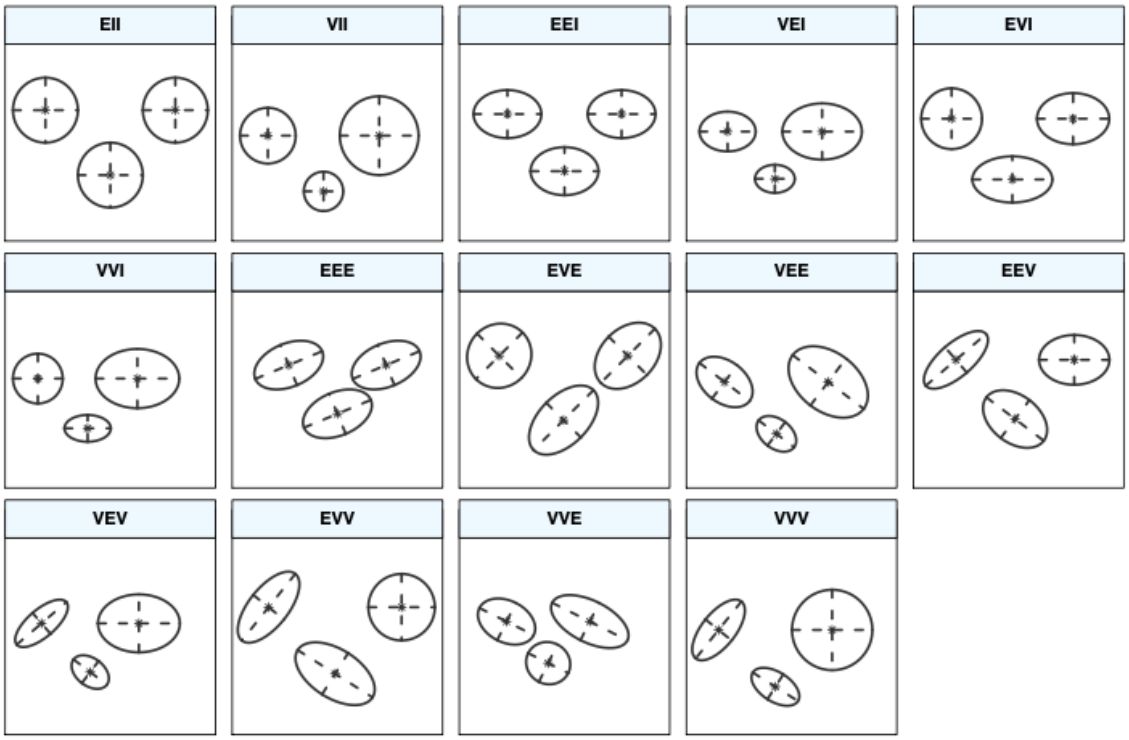


Figure A.1: Graphical overview of covariance parametrizations in mclust

Appendix B

The `BICdiff` function

```
# name of dataset
data <- dfwithout.sc

# number of tested mixture components
number <- 10

# names of mclust covariance structures
modelnames <- c('EII', 'VII', 'EEI', 'VEI', 'EVI', 'VVI', 'EEE',
                'EVE', 'VEE', 'VVE', 'EEV', 'VEV', 'EVV', 'VVV')

# the function
BICdiff <- function(data,number,modelnames){

## PART 1: calculate the BIC values by for loop
  for (i in 1:number){
      for (j in modelnames){
        IC_i <- Mclust(data=data, G=i, modelNames = j,
                 initialization = list(hcPairs = hc(data),
                                       use = "SVD"))
        bic_i_output<- as.numeric(
          ((IC_i$df)*(log(dim(data)[1]))) - (2*IC_i[[8]]))
        if(length(bic_i_output)>0){
           bic_i<- as.numeric(
             ((IC_i$df)*(log(dim(data)[1]))) - (2*IC_i[[8]]))
        }else{
           bic_i <- "NA"
        }
      out <- c(out, bic_i, i,j)
    }
  }
```

```
# make dataframe out of for loop results with colnames
m1bic <- matrix(out, ncol=3, byrow=TRUE)
d1bic <- as.data.frame(m1bic, stringsAsFactors=FALSE)
colnames(d1bic) <- c('BIC','number','Model')
d1bic$BIC <- as.numeric(d1bic$BIC)
d1bic$BICneg <- (d1bic$BIC)*-1
d1bic$number <- as.numeric(d1bic$number)
d1bic$Model <- as.factor(d1bic$Model)
d1bicc <- d1bic

## PART 2: calculate C1 and Ck by for loop
outt <- c()
outtbicdifftot <- c()

  for (i in 1:number){
    for (j in modelnames){
       bicdiffdata <- d1bicc[which(d1bicc[,'Model']== j),]
       C1_i_output <- ((i-1)*(bicdiffdata[which(
         bicdiffdata[,'number']== i),
         'BIC']-min(bicdiffdata$BIC,na.rm=TRUE)))/
                     (max(bicdiffdata$BIC,na.rm=TRUE)-
                        min(bicdiffdata$BIC,na.rm=TRUE))

       if(length(C1_i_output)>0){
           C1_i_outputbic <- ((i-1)*(bicdiffdata[which(
             bicdiffdata[,'number']== i),
             'BIC']-min(bicdiffdata$BIC,na.rm=TRUE)))/
                            (max(bicdiffdata$BIC,na.rm=TRUE)-
                               min(bicdiffdata$BIC, na.rm=TRUE))

            Cm_i_outputbic <- C1_i_outputbic/i

       }else{
            C1_i_outputbic <- "NA"
            Cm_i_outputbic <- "NA"
        }

  outt <- rbind(outt, C1_i_outputbic,Cm_i_outputbic,i,j)
  outtbicdiff <- as.data.frame(outtbicdiff, stringsAsFactors=FALSE)
  colnames(outtbicdiff) <- c('C1','Cm','i','Model')
  outtbicdiff$C1 <- as.numeric(outtbicdiff$C1)
  outtbicdiff$Cm <- as.numeric(outtbicdiff$Cm)
  outtbicdiff$i <- as.numeric(outtbicdiff$i)
  outtbicdiff$Model <- as.factor(outtbicdiff$Model)
```

```
  }
}

## PART 3: calculate C2 by for loop
outa <- c()
finaloutputbicdiff <- c()

      for (j in modelnames){
           outtbicdiffnam <- outtbicdiff[which(
                               outtbicdiff[,'Model']== j),]
           CC2 <- ((max(outtbicdiff$i,na.rm=TRUE)-
                       min(outtbicdiff$i,na.rm=TRUE))*
                      (outtbicdiffnam$C1-
                       min(outtbicdiffnam$Cm,na.rm=TRUE)))/
                     (max(outtbicdiffnam$Cm,na.rm=TRUE)-
                       min(outtbicdiffnam$Cm,na.rm=TRUE))

            if(length(CC2)>0){
                  C2 <- ((max(outtbicdiff$i,na.rm=TRUE)-
                              min(outtbicdiff$i,na.rm=TRUE))*
                         (outtbicdiffnam$C1-
                            min(outtbicdiffnam$Cm,na.rm=TRUE)))/
                         (max(outtbicdiffnam$Cm,na.rm=TRUE)-
                            min(outtbicdiffnam$Cm,na.rm=TRUE))
             }else{
                  C2 <- "NA"
            }

    outa <- rbind(outa,C2,j)
    finaloutputbicdiff <- matrix(outa, ncol=2, byrow=TRUE)
    finaloutputbicdiff <- as.data.frame(finaloutputbicdiff,
                                        stringsAsFactors=FALSE)
    colnames(finaloutputbicdiff) <- c('C2','Modell')
    finaloutputbicdiff$C2 <- as.numeric(finaloutputbicdiff$C2)
    finaloutputbicdiff$Modell <- as.factor(finaloutputbicdiff$Modell)
      }

## PART 4: calculate BIC difference and generate dataframe
BICdiff <-  cbind(outtbicdiff,finaloutputbicdiff)
BICdiff$bicdiff <- abs(BICdiff$C1-BICdiff$C2)/2
BICdiff <- BICdiff[,c('C1','cm','i','Model','C2','bicdiff')]
colnames(BICdiff) <- c('C1','Ck','k','Model','C2','BICdiff')
return(BICdiff)
}
```

Appendix C

Doing Clustering: an R-based Tutorial

Required packages

```
# list all required packages
packages <- c("foreign", "psych", "Routliers", "PerformanceAnalytics",
              "sjPlot", "ggplot2", "ggpubr", "moments", 'VIM',
              "ComplexHeatmap", "plyr", "dplyr", "tufte",
              "broom", "tidyverse", "factoextra", "cluster",
              "purrr", "NbClust", "tidyr", "vcd", 'mice',
              "ca", "mclust", "viridis", "RColorBrewer",
              "plotly", "processx","gplots", "cowplot")

# load them
lapply(packages, require, character.only = TRUE)
```

C.1 Chapter 1: the dataset

Load data

Function to upload a datafile with .sav format.

```
doto <- read.spss("PRE_doto.sav",           # directory and filename
                  use.value.labels = FALSE, # do not use labels
                  to.data.frame=TRUE)       # convert to data.frame
```

Scale scores

Functions to define items within variables and accounting for missing values

```
# define item names to variable name
keys.list <- list(
        auto_mot = c("mot1_1", "mot10_1", "mot1_2", "mot10_2",
```

```
                    "mot2_1", "mot11_1", "mot2_2", "mot11_2",
                    "mot3_1", "mot12_1", "mot3_2", "mot12_2",
                    "mot4_1", "mot13_1", "mot4_2", "mot13_2"),

        control_mot = c("mot5_1", "mot14_1", "mot5_2", "mot14_2",
                        "mot6_1", "mot15_1", "mot6_2", "mot15_2",
                        "mot7_1", "mot16_1", "mot7_2", "mot16_2",
                        "mot8_1", "mot17_1", "mot8_2", "mot17_2"),

        amot = c("mot9_1", "mot18_1" ,"mot9_2", "mot18_2")
      )

keys <- make.keys(doto, keys.list)

# if more than 80% of the items is missing,
# then the scale score becomes a missing values
Percentage <- 80

scores <- scoreItems(keys, doto,
                     min=1, max=5,  # define response scale
                     impute="none") # no imputation is done here

colnames(scores$missing) <- paste0('missing_',
                                   colnames(scores$missing))

# calculate the percentage of missing values
percNA <- as.data.frame((scores$missing/scores$n.items)*100)

# define cells that are missing
NAcells <- which(percNA > Percentage)

# code them as NA
scores$scores[NAcells] <- NA

# assign calculates scores to the original data frame
means <- as.data.frame(scores$scores)
dotoset <- cbind(doto, means)
```

Data imputation

```
# visual representation of missing data.
# check for patterns
aggr_plot <- aggr(doto, col=c('navyblue','red'),
                  numbers=TRUE, sortVars=TRUE,
                  labels=names(doto), cex.axis=.7,
```

```
                  gap=3, ylab=c("Histogram of missing data","Pattern"))

# imputing process
imputeddoto <- mice(doto,
                  m=5, # number of impute datasets
                  maxit=50,
                  meth='pmm', # imputation method
                  seed=500)
summary(imputeddoto)
doto <- complete(imputeddoto,1) # return to complete dataset
```

Internal consistency

Generate table 1.1 including measures of internal consistency

```
# extract variable from keys.list
aut_mot_rel <- dotoset[,c(keys.list$auto_mot)]

# load package
options(knitr.kable.NA = '')

sjt.itemanalysis(aut_mot_rel,
                  factor.groups.titles = "Autonomous motivation")
```

Outliers

Using the `Routliers` package to detect outliers using MAD. Visualisations are generated for univariate outliers (see figure 1.1) and multivariate outliers (see figure **??**)

```
# select variables from dataset
df <- na.omit(dotoset[,c('auto_mot','control_mot','amot')])

# calculate univariate outliers (example: auto_mot)
outliers.aum <- outliers_mad(x=df$auto_mot,
                             b = 1.4826,
                             threshold = 3,
                             na.rm = TRUE)

# display univariate outliers (example: auto_mot)
plot_outliers_mad(res = outliers.aum,
                  x = df$auto_mot,
                  pos_display = FALSE)

# bivariate outliers (example: auto_mot and control_mot)
outliers.aumcom <- outliers_mahalanobis(
```

```
                    df[,c('auto_mot','control_mot')],
                    na.rm=TRUE)

# display bivariate outliers (example: auto_mot and control_mot)
plot_outliers_mahalanobis(outliers.aumcom,
                          x=df[,c('auto_mot','control_mot')],
                          pos_display = FALSE)


# Define outliers in dataset:
## 1. make new variable 'outliers' with levels 'No'
df$outliers <- "No"

## 2. define all rows detected as univariate outliers as 'Yes'
## (example: auto_mot)
df[outliers.aum$outliers_pos,c('outliers')] <- "Yes"

## 3. define all rows detected as bivariate outliers as 'Yes'
## (example: auto_mot and control_mot)
df[outliers.aumcom$outliers_pos,c('outliers')] <- "Yes"

## Check percentage of deteceted outliers
round(prop.table(summary(as.factor(df$outliers))),2)

# Make dataset without deteceted outliers
dfwithout <- df[which(df$outliers == "No"), ]

# Remove outlier variable and empty rows
dfwithout <- na.omit(dfwithout[,c('auto_mot','control_mot','amot')])
```

Descriptive statistics

Descriptives of the dataset like table 1.4

```
# generate descriptive statistics
psych::describe(dfwithout)
```

Generating a density plot like figure 1.4

```
# density plot including raw data and
# normal density curve based on the mean and standard deviation
ggdensity(dfwithout,
          x = "auto_mot",
          fill = "#2F3C4D",
          title = NULL) +
```

```
  stat_overlay_normal_density(color = "red", linetype = "dashed")+
  xlab('Scores for autonomous motivation')
```

Data transformation

```
# check automatically multiple data transformations
BNobject <- bestNormalize(dfwithout$amot)

# check output
BNobject

# visualize out-of-sample estimates of normality
boxplot(log10(BNobject$oos_preds), yaxt = 'n')
axis(2, at=log10(c(.1,.5, 1, 2, 5, 10)), labels=c(.1,.5, 1, 2, 5, 10))

# apply transformation
ordernorm <- orderNorm(dfwithout$amot)
dfwithout$amot.t <- ordernorm$x.t

# check raw data with normal density curve
ggdensity(dfwithout,
          x = "amot.t",
          fill = "#2F3C4D",
          title = "AMOT (OQT-transformed)") +
  stat_overlay_normal_density(color = "red", linetype = "dashed")
```

Data content

Matrix including data distributions, scatter plots and correlations like 1.7

```
chart.Correlation(dfwithout, histogram=TRUE, pch=19)
```

Standardization

```
# scale the dataframe
dfwithout.sc <- scale(dfwithout,scale=TRUE)

# as it is saved as a matrix, convert to data frame
dfwithout.sc <- as.data.frame(dfwithout.sc)
```

Density heatmap

See figure 1.8

```
densityHeatmap(dfwithout.sc, title=NULL)
```

C.2 Chapter 2: K-Means Cluster Analysis

C.2.1 (Dis)similarity measurement

See figure 2.2

```
# make distance matrix based on euclidean distance
dismatrix <- get_dist(dfwithout.sc,
                      method= "euclidean")

# show total distance matrix
dismatrix

# show only the first 3 rows and 3 columns
round(as.matrix(dismatrix)[1:3, 1:3], 1)

# visualize distance matrix in colors
dismatrix.color <- get_dist(dfwithout.sc,
                            method= "euclidean")
fviz_dist(
  dismatrix.color,
  order = TRUE,
  show_labels = TRUE,
  lab_size = NULL,
  gradient = list(low = "red",
                  mid = "white",
                  high = "blue"))
```

C.2.2 Intervention of Hierarchical Clustering

Studying the different linkage methods like figure 2.3

```
# calculate all linkage methods based on the distance matrix
linkage.average  <- hclust(d= dismatrix, method = "average")
linkage.single   <- hclust(d= dismatrix, method = "single")
linkage.complete <- hclust(d= dismatrix, method = "complete")
linkage.ward     <- hclust(d= dismatrix, method = "ward.D2")

# visualize them
dend.av <- fviz_dend(linkage.average,
                     cex=0.5, main = "Average linkage")
dend.si <- fviz_dend(linkage.single,
                     cex=0.5, main = "Single linkage")
dend.co <- fviz_dend(linkage.complete,
                     cex=0.5, main = "Complete linkage")
```

```
dend.wa <- fviz_dend(linkage.ward,
                     cex=0.5, main = "Ward's method")

# function to plot figures side by side in 2 rows
cowplot::plot_grid(dend.av,dend.si,dend.co,dend.wa, nrow = 2)
```

C.2.3 Validation and choosing the number of clusters

Generate the correlations between the linkage mathods and the distance matrix like table 2.1

```
# calculate cophenetic distances for each linkage method
coph.average <- cophenetic(linkage.average)
coph.single <- cophenetic(linkage.single)
coph.complete <- cophenetic(linkage.complete)
coph.ward <- cophenetic(linkage.ward)

# calculate correlations
table.corr.coph <- round(as.data.frame(c(
                        cor(coph.average, dismatrix),
                        cor(coph.single, dismatrix),
                        cor(coph.complete, dismatrix),
                        cor(coph.ward, dismatrix)
                    )),3)

# make table
table.corr.coph <- cbind(c('Average',
                           'Single',
                           'Complete',
                           'Ward'),
                         table.corr.coph)
colnames(table.corr.coph) <- c('Linkage method','Correlations')
table.corr.coph
```

Generate the agglomerative coefficients like table 2.2

```
# define AC function
m <- c( "average", "single", "complete", "ward" )
names(m) <- c( "average", "single", "complete", "ward")

ac <- function(x) {agnes(dfwithout.sc, method = x)$ac}

# perform the function
ac.table <- round(as.data.frame(map_dbl(m, ac)),3)
```

```
# make table
ac.tablenames <- rownames(ac.table)
ac.table <- cbind(ac.tablenames,ac.table)
rownames(ac.table) <- NULL
colnames(ac.table) <- c("Linkage method", "ac")
ac.table
```

Perform the stopping rule like table 2.3

```
# calculate duda index for range of 5 clusters
duda <- NbClust(dfwithout.sc,
                distance = "euclidean",
                method = "ward.D2",
                max.nc = 9,
                index = "duda")

# calculate pseudo index for range of 5 clusters
pseudot2 <- NbClust(dfwithout.sc,
                    distance = "euclidean",
                    method = "ward.D2",
                    max.nc = 9,
                    index = "pseudot2")

# make table for output
table.duda.pseudo <- rbind(duda$All.index,pseudot2$All.index)
rownames(table.duda.pseudo) <- c('Duda-Hart index','Pseudo T2')

# return optimal number of clusters
duda$Best.nc
```

Colorize the dendogram based on the final decision like figure 2.4

```
# visualize dendogram
## make different dataframe for the figure
dfwithout.sc.HC <- dfwithout.sc

## define linkage method and number of groups
dfwithout.sc.HC$grps <- cutree(linkage.ward, k=3)
dfwithout.sc.HC$grps <- as.factor(dfwithout.sc.HC$grps)

# make figure
fviz_dend(linkage.ward,
          k = 3,
          cex = 0.5,
```

```
          k_colors = c("#2f3c4d",
                       "#ad131b",
                       "#cc6a0e"), # colors surfaces
          color_labels_by_k = TRUE,
          rect = TRUE,
          rect_border = c("#2f3c4d",
                          "#ad131b",
                          "#cc6a0e"), # colors borders
          rect_fill = TRUE)
```

Generate the centroids of the hierarchical clustering like table 2.4

```
# generate hierarchically clustered centroids
centroids <- dfwithout.sc.HC %>%
             group_by(grps) %>%
             summarise_all("mean")

# select columns and save it in the objects 'centroids'
centroids <- round(as.data.frame(centroids[,2:4]),2)
```

###K-Means clustering algorithms

Perform the K-Means clustering algorithms based on the centroids of the hierarchical clustering procedure and visualize a 2D PCA plot like figure 2.5

```
# perform all k-means clustering methods

km.Lloyd <- kmeans(dfwithout.sc, centroids,
                   iter.max=10, nstart=1,
             algorithm = "Lloyd")

km.Forgy <- kmeans(dfwithout.sc, centroids,
                   iter.max=10, nstart=1,
             algorithm = "Forgy")

km.MQ    <- kmeans(dfwithout.sc, centroids,
                   iter.max=10, nstart=1,
             algorithm ="MacQueen")

km.HW    <- kmeans(dfwithout.sc, centroids,
                   iter.max=10, nstart=1,
             algorithm = "Hartigan-Wong")

# generate 2D PCA plots to compare K-means algorithms
p.lloyd <- fviz_cluster(km.Lloyd, data=dfwithout.sc,
```

```
                        palette = c("#2f3c4d",
                                    "#ad131b",
                                    "#cc6a0e"),
             ellipse.type="convex",
             star.plot=TRUE,
             repel=TRUE,
             main = "Lloyd algorithm",
             ggtheme = theme_minimal(),geom = "point")

p.forgy <- fviz_cluster(km.Forgy, data=dfwithout.sc,
                        palette = c("#2f3c4d",
                                    "#ad131b",
                                    "#cc6a0e"),
             ellipse.type="convex",
             star.plot=TRUE,
             repel=TRUE,
             main = "Forgy algorithm",
             ggtheme = theme_minimal(),geom = "point")

p.mq <- fviz_cluster(km.MQ, data=dfwithout.sc,
                     palette = c("#2f3c4d",
                                 "#ad131b",
                                 "#cc6a0e"),
            ellipse.type="convex",
             star.plot=TRUE,
             repel=TRUE,
             main = "MacQueen algorithm",
             ggtheme = theme_minimal(),geom = "point")

p.hw <- fviz_cluster(km.HW, data=dfwithout.sc,
                     palette = c("#2f3c4d",
                                 "#ad131b",
                                 "#cc6a0e"),
             ellipse.type="convex",
             star.plot=TRUE,
             repel=TRUE,
             main = "Hartigan & Wong algorithm",
             ggtheme = theme_minimal(),geom = "point")

cowplot::plot_grid(p.lloyd,p.forgy,p.mq,p.hw, nrow = 2)
```

After choosing which algorithm to use, we can obtain the K-Means based centroids like 2.7

```
# get centroids of K-Means clustering
km.means <- as.data.frame((aggregate(dfwithout.sc,
                          by=list(cluster=km.HW$cluster),
                          mean)))
km.meanstable <- round(t(km.means[,2:4]),2)
colnames(km.meanstable) <- c('cluster 1', 'cluster 2', 'cluster 3')
km.meanstable
```

In this chapter, we performed the procedure of Hierarchical K-Mean clustering step by step. All these steps can be performed using only one function.

```
hkm <- hkmeans(dfwithout.sc,                          # dataset
               k = 3,                                 # number of clusters
               hc.metric="euclidian",                 # distance measurement
               hc.method='ward.D2',                   # linkage method
               iter.max = 10,                         # iterations
               km.algorithm = "Hartigan-Wong")        # K-Means algorithm
```

C.2.4 Analytical considerations

Clustering Tendency: compare the original dataset with a random dataset. This can be done by performing PCA like figure 2.6 and figure 2.7

```
# generate random df
random_df <- as.data.frame(apply(dfwithout.sc,2,
                                 function(x){runif(length(x),
                                                   min(x),
                                                   max(x))}
                                 ))

# calculate PCA for original dataset
## visualization on the level of cases
pca.ind.o <- fviz_pca_ind(prcomp(dfwithout.sc),
             title = "PCA original data",
             subtitle="cases",
             ggtheme = theme_minimal(),
             geom = "point",
             lab_sze = FALSE,
             labelsize = 3)

## visualization on the level of variables
pca.var.o <- fviz_pca_var(prcomp(dfwithout.sc),
             subtitle = "variables",
             title= ' ',
```

```
              ggtheme = theme_minimal(),
              lab_sze = FALSE,
              labelsize = 3,
              ylim=c(-1,1),
              xlim=c(0,1.5))

cowplot::plot_grid(pca.ind.o,pca.var.o,nrow = 1)

# calculate PCA for random dataset
## visualization on the level of cases
pca.ind.r <- fviz_pca_ind(prcomp(random_df),
              title = "PCA random data",
              subtitle="cases",
              ggtheme = theme_minimal(),
              geom = "point",
              lab_sze = FALSE,
              labelsize = 3)

## visualization on the level of variables
pca.var.r <- fviz_pca_var(prcomp(random_df),
              subtitle = "variables",
              ggtheme = theme_minimal(),
              lab_sze = FALSE,
              labelsize = 3,
               title= ' ')

cowplot::plot_grid(pca.ind.r,pca.var.r,nrow = 1)
```

Compare visualization of the Hierarchical K-Means procedure like figure 2.8

```
# HKM for original dataset
km.pca.o <- hkmeans(dfwithout.sc,
                    k = 3,
                    hc.metric="euclidian",
                    hc.method='ward.D2',
                    iter.max = 10,
                    km.algorithm = "Hartigan-Wong")

# plot 2D PCA figure where colors specify cluster surfaces
kmean.pca.o <- fviz_cluster(km.pca.o, data=dfwithout.sc,
                   palette = c("#2f3c4d","#ad131b","#cc6a0e"),
                   ellipse.type="convex",
                   star.plot=TRUE,
                   repel=TRUE,
```

```
                  main = "K means original dataset",
                  ggtheme = theme_minimal(),
                  geom = "point")

# HKM vor random dataset
km.pca.r <- hkmeans(random_df,
                    k = 3,
                    hc.metric="euclidian",
                    hc.method='ward.D2',
                    iter.max = 10,
                    km.algorithm = "Hartigan-Wong")

# plot 2D PCA figure where colors specify cluster surfaces
kmean.pca.r <- fviz_cluster(km.pca.r, data=random_df,
                 palette = c("#2f3c4d","#ad131b","#cc6a0e"),
                 ellipse.type="convex",
                 star.plot=TRUE,
                 repel=TRUE,
                 main = "K means random dataset",
                  ggtheme = theme_minimal(),
                  geom = "point")

cowplot::plot_grid(kmean.pca.o,kmean.pca.r,nrow = 1)
```

The Hopkins statistic: check result for the original and random dataset like table 2.8

```
# calculate Hopkins statistic for the original dataset
res.o <- get_clust_tendency(dfwithout.sc,
                            n = nrow(dfwithout.sc)-1,
                            graph=FALSE)

H_dataset <- round(res.o$hopkins_stat,3)

# calculate Hopkins statistic for the random dataset
res.r <- get_clust_tendency(random_df, n =
                            nrow(random_df)-1,
                            graph=FALSE)
H_random <- round(res.r$hopkins_stat,3)

# set up a table to compare both results
table_H <- cbind(H_dataset,H_random)
colnames(table_H) <- c('dfwithout.sc','random dataset')
rownames(table_H) <- 'Hopkin statistic'
table_H
```

C.2.5 Determining the number of clusters

Elbow method: compute within-cluster sum of square and generate elbow plot like figure 2.9 (left).

```
# function to compute total within-cluster sum of square
# for a range of k clusters
# just run it
wss <- function(k) {
hkmeans(dfwithout.sc, k,
               hc.metric="euclidian",
               hc.method='ward.D2',
               iter.max = 10,
               km.algorithm = "Hartigan-Wong")$tot.withinss
}

# Specify range of clusters
k.values <- 1:15

# extract wss for 2-15 clusters
wss_values <- map_dbl(k.values, wss)
elbowmethod <- as.data.frame(t(rbind(k.values,wss_values)))

ggplot(data=elbowmethod, aes(x=k.values, y=wss_values)) +
         geom_line(stat="identity",width = 0.7,
                   position=position_dodge(),color='black')+
         geom_point()+
         theme(
                legend.position = 'right',
                legend.spacing.x = unit(1, 'mm'),
                axis.title.x = element_blank(),
                plot.caption = element_text(color = "black"))+
         scale_x_continuous(name="K clusters", breaks=c(1:15))+
         theme_minimal()+
         ylab("WSS")+
         labs(subtitle = "Elbow method") +
         geom_vline(xintercept=3,linetype='dashed',color='red')
```

Within versus Between variance: code to generate the both the between and within variance like figure 2.9 (right).

```
minnumber <- 1  # choose minimum number of k clusters
maxnumber <- 15 # choose maximum number of k clusters
dataset <- dfwithout.sc # dataset

# for loop generating both between and within variance
# just run it
uncertaintytable<-c()
for (i in minnumber:maxnumber) {
  km.res <- hkmeans(dataset, i,
                hc.metric="euclidian",
                hc.method='ward.D2',
                iter.max = 10,
                km.algorithm = "Hartigan-Wong")

  sum_i <- km.res$betweenss + km.res$tot.withinss
  number_i <- i
  ss_i <- cbind(km.res$tot.withinss, km.res$betweenss,sum_i,number_i)
  ss_i <- as.data.frame(ss_i)
  ss_i <- round(ss_i,2)
  colnames(ss_i) <- c('within','between','sum',"clusters")
  uncertaintytable <- rbind(uncertaintytable,ss_i)
}

# making a dataset in preparation of the figure. Just run it
uncertaintytable$within <- (uncertaintytable[,1]/
                              uncertaintytable[,3])*100
uncertaintytable$between <- (uncertaintytable[,2]/
                               uncertaintytable[,3])*100
uncertaintytablee <- uncertaintytable[,c("clusters",
                                         'within',
                                         'between')]
uncertaintytablee <- round(uncertaintytablee,2)
data_longtable <- gather(uncertaintytablee, type,
                         variance, c('within','between'),
                         factor_key=TRUE)

# figure based on for loop and table
ggplot(data=data_longtable,
          aes(x=clusters, y=variance, fill=type)) +
          geom_bar(stat="identity",width = 0.7,
                    position=position_dodge(),color='black')+
          scale_fill_manual("Type of variance",
                            values = c("within" = "#2f3c4d",
                                       "between" = "#ad131b"))+
```

```
          theme(legend.spacing.x = unit(1, 'mm'),
                axis.title.x = element_blank(),
                plot.caption = element_text(color = "black"))+
          scale_y_continuous(labels = function(x) paste0(x, "%"))+
          scale_x_continuous(name="Clusters", breaks=c(1:15))+
          theme_minimal()+
          theme(legend.position = 'top')+
          labs(subtitle = "Between- and Within variance")
```

Silhouette coefficients by cases: generate silhouette coefficients in ordered fashion by cluster like figure 2.10 (left).

```
# run HKM
hkm.res <- hkmeans(dfwithout.sc, k=3,
               hc.metric="euclidian",
               hc.method='ward.D2',
               iter.max = 10,
               km.algorithm = "Hartigan-Wong")

# generate silhouette values for each case and each cluster
ss <- silhouette(hkm.res$cluster,
                 dist(dfwithout.sc, method = "euclidean"))
ss <- cbind(ss[,1],ss[,2],ss[,3])
ss <- as.data.frame(ss)
colnames(ss) <- c('cluster','neighbor','silhouette')
ss <- with(ss, ss[order(cluster, silhouette,decreasing = TRUE),])
ss$cluster <- as.factor(ss$cluster)
ss$ppnr <- 1:dim(ss)[1]
ss <- ss[order(match(ss$cluster, c("1", "2", "3"))),]

# figure a plot where cases are ordered
# by decreasing silhouette and by cluster
ggplot(data=ss, aes(x=ppnr,
               y=silhouette,
               fill=cluster,
               order=cluster)) +
       geom_bar(stat="identity",width = 0.7,
               position=position_dodge())+
       scale_fill_manual("Cluster",
                         values = c("1"="#2f3c4d",
                                    "2"="#ad131b",
                                    "3"="#cc6a0e"))+
        theme(
           legend.position = 'right',
           legend.spacing.x = unit(1, 'mm'),
```

```
          axis.title.x = element_blank(),
          plot.caption = element_text(color = "black"))+
        theme_minimal()+
        ylab("Silhouette width")+
        labs(subtitle = "Silhouette coefficients by cases")+
        xlab("Cases")
```

Average silhouette method: generate figure like 2.10 (right)

```
# function generating silhouettes for a range of k clusters
avg_sil <- function(k) {
  km.res <- hkmeans(dfwithout.sc, k,
                hc.metric="euclidian",
                hc.method='ward.D2',
                iter.max = 10,
                km.algorithm = "Hartigan-Wong")
  ss <- silhouette(km.res$cluster, dist(dfwithout.sc,
                                        method = "euclidean"))
  mean(ss[, 3])
}

# Specify range of clusters
k.values <- 2:15

# extract average silhouette for 2-15 clusters
avg_sil_values <- map_dbl(k.values, avg_sil)

averagesilh <- as.data.frame(t(rbind(k.values,avg_sil_values)))

# plot the values
ggplot(data=averagesilh,
       aes(x=k.values, y=avg_sil_values))+
       geom_line(stat="identity",width = 0.7,
                 position=position_dodge(),color='black')+
       geom_point()+
       theme(
          legend.position = 'right',
          legend.spacing.x = unit(1, 'mm'),
          axis.title.x = element_blank(),
          plot.caption = element_text(color = "black"))+
       scale_x_continuous(name="K clusters",
                          breaks=c(1:15))+
       theme_minimal()+
       ylab("Average Silhouettes")+
       labs(subtitle = "Average silhouette method") +
```

```
      geom_vline(xintercept=3,
                 linetype='dashed',
                 color='red')
```

Gap statistic: generate figure like 2.11

```
fviz_nbclust(dfwithout.sc,hkmeans,
             linecolor = "black",
             method="gap_stat",  #gap method
             nboot=50) +
             labs(title=NULL,
                      subtitle = "Gap statistic",
                      xlab='K clusters') +
             geom_vline(xintercept=3,
                        linetype='dashed',
                        color='red')+
             theme_minimal()
```

Calculate and extract cases with negative silhouette widths like table 2.9

```
ss[which(ss$silhouette <= 0),1:3]
```

Summary of 30 indices: plot frequencies of indices for each number of clusters like figure 2.12

```
# calculate 30 indices
otherind <- NbClust(dfwithout.sc,
                    distance = "euclidean",
                    min.nc = 2,
                    max.nc = 10,
                    method = "ward.D2",
                    index ="all")

# extract frequencies in dataframe
bestnc <- as.data.frame(t(otherind$Best.nc))
bestnc$Number_clusters <- as.factor(bestnc$Number_clusters)
summ.bestnc <- as.data.frame(summary(bestnc$Number_clusters))
clusters <- rownames(summ.bestnc)
summ.bestnc <- cbind(summ.bestnc,clusters)
colnames(summ.bestnc) <- c('Frequency',"clusters")
summ.bestnc$clusters <- as.numeric(summ.bestnc$clusters)

# use this dataframe to visualize the frequencies
ggplot(data=summ.bestnc, aes(x=clusters, y=Frequency)) +
```

```
        geom_bar(stat="identity",width = 0.7,
                 position=position_dodge(),fill="#2f3c4d")+
        theme(
           legend.position = 'right',
           legend.spacing.x = unit(1, 'mm'),
           axis.title.x = element_blank(),
           plot.caption = element_text(color = "black"))+
        theme_minimal()+
        ylab("Frequency of indices")+
        scale_y_continuous(breaks = seq(0,
                           max(summ.bestnc$Frequency),
                           by = 1))+
        scale_x_continuous(name="Clusters",
                           breaks=c(0:max(summ.bestnc$clusters,
                           na.rm=TRUE)))+
        xlab("Clusters")+
        ggtitle('Summary frequency 30 indices')
```

Cluster overlap: visualize data coverage on 2D PCA plot like figure 2.13

```
# for instance, generate HKM for specific number of clusters
hkm2 <- hkmeans(dfwithout.sc, k = 2,
                hc.metric="euclidian",
                hc.method='ward.D2',
                iter.max = 10,
                km.algorithm = "Hartigan-Wong")

fviz_cluster(hkm2,
             ellipse.type="convex",
             palette = c("#2f3c4d","#ad131b"), # colors
             repel=FALSE,
             ggtheme=theme_minimal(),
             geom=NULL) +
             ggtitle("k = 2")
```

C.2.6 Cluster characteristics

Cluster stability: to perform the double split cross-validation procedure as in Cluster stability

```
# make new dataset 'subset'
subset <- dfwithout.sc

# assign original clustering as a variable 'hkm'
```

```
subset$hkm <- as.factor(hkm$cluster)

# start position for random choosing
set.seed(7)

# make two new subsamples A and B having 50% of the dataset
ss <- sample(1:2,size=nrow(subset),replace=TRUE,prob=c(0.5,0.5))
subsetA <- subset[ss==1,]
subsetB <- subset[ss==2,]

# perform HKM to both subsets,
# selecting the study variables
hkmA <- hkmeans(subsetA[,1:3], k = 3,
                hc.metric="euclidian",
                hc.method='ward.D2',
                iter.max = 10,
                km.algorithm = "Hartigan-Wong")

hkmB <- hkmeans(subsetB[,1:3], k = 3,
                hc.metric="euclidian",
                hc.method='ward.D2',
                iter.max = 10,
                km.algorithm = "Hartigan-Wong")


# use the centroids of HKM procedures from the others subset
# as initial values for a K-Mean algorithm
kmeanAB <- kmeans(subsetA[,1:3],
                  hkmB$centers,  # initial values subset B
                  iter.max=10, nstart=1,
                  algorithm = "Hartigan-Wong")

subsetA$AB <- as.factor(kmeanAB$cluster) # add result to subset

kmeanBA <- kmeans(subsetB[,1:3],
                  hkmA$centers,  # initial values subset A
                  iter.max=10, nstart=1,
                  algorithm = "Hartigan-Wong")

subsetB$BA <- as.factor(kmeanBA$cluster) # add result to subset

# make contingency tables
mytableAB <- with(subsetA, table(hkm,AB))
mytableBA <- with(subsetB, table(hkm,BA))
```

```
# Compute kappa indices
kappa.ab <- Kappa(mytableAB)
kappa.ab

kappa.ba <- Kappa(mytableBA)
kappa.ba
```

Heatmap: visualize cases by color representing the z-score on the variables, sorted by cluster like figure 2.14

```
Heatmap(dfwithout.sc, name ="Z-score", km = 3,
        column_names_gp = gpar(fontsize = 8),
        row_names_gp = gpar(fontsize = 6))
```

Bidimensional plot: plotting final 2D PCA plot including data surface and data points with colors representing the clusters like figure 2.15

```
fviz_cluster(hkm,
             ellipse.type="convex",
             palette = c("#2f3c4d","#ad131b","#cc6a0e"),
             repel=FALSE,
             pointsize = 1,
             ggtheme=theme_minimal(),
             geom=c("point"))+
             labs(title=NULL, subtitle = NULL)
```

Barplot: calculating and visualizing a barplot with the standardized variables as a function of Hierarchical K-Means clusters like figure 2.16

```
# make new final dataset  including the original dataset
finaldata <- dfwithout.sc
finaldata$Kmean <- as.factor(hkm$cluster) # add HKM clusters
levels(finaldata$Kmean) <- c('Good','Bad','Low') # name clusters

# Check cluster sized in percentages
round(prop.table(summary(finaldata$Kmean)),2)

# Get centroids
hkmdata <- as.data.frame((aggregate(
           dfwithout.sc,
           by=list(cluster=hkm$cluster), mean)))

# Convert from wide to long data format
data_long <- tidyr::gather(hkmdata, type, measurement,
```

```
                    c('auto_mot','control_mot','amot'),
                    factor_key=TRUE)
colnames(data_long) <- c('Clusters','Type','Standardized value')
data_long$Clusters <- as.numeric(data_long$Clusters)

# Label variables
data_long$Type <- factor(data_long$Type,
                         labels=c("Autonomous motivation",
                                  "Controlled motivation",
                                  "Amotivation"))

# Make barplot
ggplot(data=data_long,
       aes(x=Clusters,
           y=`Standardized value`,
           fill=Type)) +
       geom_bar(stat="identity",width = 0.7,
                position=position_dodge(),color='black')+
       scale_fill_manual("Type of Motivation",
                         values = c("Autonomous motivation" = "#2f3c4d",
                                    "Controlled motivation" = "#ad131b",
                                    "Amotivation" = "#cc6a0e"))+
       theme(
           legend.position = 'right',
           legend.spacing.x = unit(1, 'mm'),
           axis.title.x = element_blank(),
           plot.caption = element_text(color = "black"))+
       scale_y_continuous(limits = c(-1.1,1.1))+
       scale_x_continuous(name="Cluster", breaks=c(1:3),
                          labels= c("Cluster 1\n35%",
                                    "Cluster 2\n30%",
                                    "Cluster 3\n35%"))+
       theme_minimal()
```

Test between-cluster difference: for interpretation, differences between clusters can be tested by a MANOVA and ANOVA for each (unstandardized) study variable like 2.12. The same code can be used for LPA like table 3.7.

```
# function to round only the numeric variables in dataset to 2 digits
round_df <- function(x, digits) {
    numeric_columns <- sapply(x, mode) == 'numeric'
    x[numeric_columns] <-  round(x[numeric_columns], digits)
    x
}
```

```
# perform MANOVA
## for LPA: replace 'Kmean' variable by LPA variable
manova.km <- manova(cbind(auto_mot, control_mot, amot) ~ Kmean,
                    data = finaldata)

# check results for Wilks' lambda
summary(manova.km, test="Wilks")

# ANOVA, for example autonomous motivation
aum <- lm(auto_mot ~ Kmean, data = finaldata)
summary(aum)
anova(aum)
```

C.3 Chapter 3: Latent Profile Analysis

Before performing LPA, one can explore the multivariate density plots from top view like figure 3.2 (left) or in (fancy) 3D like figure 3.2 (right).

```
# bivariate density plot with top view
topdensityplot <- ggplot(dfwithout.sc,
                         aes(x = auto_mot, y = control_mot)) +
                         stat_density2d(aes(fill = ..density..),
                                 geom = 'tile', contour = F) +
                         scale_fill_viridis()+
                 theme(
                         legend.position = 'none',
                         legend.spacing.x = unit(1, 'mm'),
                         axis.title.x = element_blank(),
                         plot.caption = element_text(color = "black")
                      )+
                 theme_minimal()
topdensityplot

# 3D bivariate density plot

dens <- kde2d(dfwithout.sc$auto_mot, dfwithout.sc$control_mot)
plot_ly(x = dens$x,
            y = dens$y,
            z = dens$z) %>% add_surface() %>%
        layout(
            scene = list(
            xaxis = list(title = "auto_mot"),
            yaxis = list(title = "control_mot"),
```

```
            zaxis = list(title = "Density")
          ))
```

C.3.1 Mixture modeling

By following function of the `mclust` package, one can perform mixture modeling. Remark no number of clusters is specified or no model is specified.

```
mod <- Mclust(dfwithout.sc,
            #   G = 3,
            #  modelNames='EEE',
               initialization =
               list(
                 hcPairs = hc(dfwithout.sc),
                 use = "SVD",
                 subset = NULL)
               )
mod
```

C.3.2 Model selection

BIC: calculating BIC values for a specified number of k components and models from which results are showed in a figure like figure 3.3

```
# specify name of the dataset
data <- dfwithout.sc

# specify number of maximum tested clusters
number <- 5

# names of mclust covariance structures
modelnames <- c('EII', 'VII', 'EEI', 'VEI', 'EVI', 'VVI', 'EEE',
                'EVE', 'VEE', 'VVE', 'EEV', 'VEV', 'EVV', 'VVV')

# for loop extracting BIC values for each cluster and model
out <- c()
for (i in 1:number){
  for (j in modelnames){
  IC_i <- Mclust(data=data, G=i, modelNames = j,
                 initialization = list(hcPairs = hc(data), use = "SVD"))
  bic_i_output<- as.numeric(((IC_i$df)*(log(1046))) - (2*IC_i[[8]]))
  if(length(bic_i_output)>0){
    bic_i<- as.numeric(((IC_i$df)*(log(1046))) - (2*IC_i[[8]]))
  }else{
```

```
    bic_i <- "NA"
  }
  out <- c(out, bic_i, i,j)
  }
}

# generate dataframe based on BIC values, clusters and models
m1bic <- matrix(out, ncol=3, byrow=TRUE)
d1bic <- as.data.frame(m1bic, stringsAsFactors=FALSE)
colnames(d1bic) <- c('BIC','number','Model')
d1bic$BIC <- as.numeric(d1bic$BIC)
d1bic$BICneg <- (d1bic$BIC)*-1
d1bic$number <- as.numeric(d1bic$number)
d1bic$Model <- as.factor(d1bic$Model)

# visualize the result in a line figure
ggplot(d1bic,
       aes(x=number, y=BIC,
           group=Model, color=Model,
           shape=Model)) +
      scale_shape_manual(values=1:nlevels(d1bic$Model))+
      geom_line(aes(linetype=Model),size=1)+
      geom_point(aes(shape=Model),size=3)+
      theme(
          legend.position = 'right',
          legend.spacing.x = unit(1, 'mm'),
          axis.title.x = element_blank(),
          plot.caption = element_text(color = "black"))+
      theme_minimal()+
      scale_x_continuous(name="Components", breaks=c(1:5))+
      ylab("BIC")
```

To extract the lowest BIC value and model given a specified number of components like 3.3

```
table_widebic <- tidyr::spread(d1bic, number, BIC)

# for example
k <- '3' # specify number of components

## generate name of model
minc3bic <- table_widebic[which.min(table_widebic[,k]),'Model']
minc3bic
```

```
## generate BIC value
minc3bicval <- table_widebic[which.min(table_widebic[,k]),k]
minc3bicval
```

Likelihood Ratio Test: perform bootstrapped LRT like table 3.2

```
mclustBootstrapLRT(dfwithout.sc,      # specify dataset
                   modelName = "VEV") #specify model
```

AIC: calculating AIC values for a specified number of k components and models from which results are showed in a figure like figure 3.5

```
data <- dfwithout.sc # specify dataset

number <- 5 # maximum number of components

modelnames <- c('EII', 'VII', 'EEI', 'VEI', 'EVI', 'VVI', 'EEE',
                'EVE', 'VEE', 'VVE', 'EEV', 'VEV', 'EVV', 'VVV')

# for loop to calculate AIC values
out <- c()
for (i in 1:number){
  for (j in modelnames){
  IC_i <- Mclust(data=data, G=i, modelNames = j,
                 initialization = list(
                   hcPairs = hc(data),
                   use = "SVD"))
  aic_i_output<- as.numeric((2*IC_i$df) - (2*IC_i[[8]]))
  if(length(aic_i_output)>0){
    aic_i<- (2*IC_i$df) - (2*IC_i[[8]])
  }else{
    aic_i <- "NA"
  }
  out <- c(out, aic_i, i,j)
  }
}

# extracting AIC values
m1aic <- matrix(out, ncol=3, byrow=TRUE)
d1aic <- as.data.frame(m1aic, stringsAsFactors=FALSE)
colnames(d1aic) <- c('AIC','number','Model')
d1aic$AIC <- as.numeric(d1aic$AIC)
d1aic$AICneg <- (d1aic$AIC)*-1
d1aic$number <- as.numeric(d1aic$number)
d1aic$Model <- as.factor(d1aic$Model)
```

```
# plotting
ggplot(d1aic,
       aes(x=number, y=AIC,
           group=Model, color=Model,
           shape=Model)) +
       scale_shape_manual(values=1:nlevels(d1aic$Model))+
       geom_line(aes(linetype=Model),size=1)+
       geom_point(aes(shape=Model),size=3)+
       theme(
           legend.position = 'right',
           legend.spacing.x = unit(1, 'mm'),
           axis.title.x = element_blank(),
           plot.caption = element_text(color = "black"))+
       theme_minimal()+
       scale_x_continuous(name="Components", breaks=c(1:5))+
       ylab("AIC")
```

To extract the lowest AIC value and model given a specified number of components like 3.3

```
table_wideaic <- tidyr::spread(d1bic, number, AIC)

# for example
k <- '3' # specify number of components

## generate name of model
minc3aic <- table_wideaic[which.min(table_wideaic[,'3']),'Model']
minc3aic

## generate AIC value
minc3aicval <- table_wideaic[which.min(table_wideaic[,'3']),'3']
minc3aicval
```

Entropy: calculating relative Entropy values for a specified number of k components and models from which results are showed in a figure like figure 3.6

```
data <- dfwithout.sc # specify dataset

number <- 5 #number of maximum tested clusters

modelnames <- c('EII', 'VII', 'EEI', 'VEI', 'EVI', 'VVI', 'EEE',
                'EVE', 'VEE', 'VVE', 'EEV', 'VEV', 'EVV', 'VVV')

# for loop to calculate relative entropy
```

```
out <- c()
for (i in 1:number){
  for (j in modelnames){
  clustcom_i <- Mclust(data=data, G=i, modelNames = j,
                 initialization = list(
                   hcPairs = hc(data),
                   use = "SVD"))

  z_output_i<- clustcom_i$z
  if(length(z_output_i)>0){
    probs_i <- as.data.frame(clustcom_i$z)
    datameanprobs_i <- probs_i %>%
                         summarise_all("mean")
    datameanprobs_i <- as.vector(datameanprobs_i)

    entropy<-function (p) sum(-p*log(p))
    error_prior_i <- entropy(datameanprobs_i) # Class proportions
    error_post_i <- mean(apply(probs_i, 1, entropy))
    R2_entropy_i <- (error_prior_i - error_post_i) / error_prior_i
  }else{
    R2_entropy_i <- "NA"
  }
  out <- c(out, R2_entropy_i, i,j)
  }
}

# make dataframe out of for loop results with colnames
m1ent <- matrix(out, ncol=3, byrow=TRUE)
d1ent <- as.data.frame(m1ent, stringsAsFactors=FALSE)
colnames(d1ent) <- c('Rel_entropy','number','Model')
d1ent$Rel_entropy <- as.numeric(d1ent$Rel_entropy)
d1ent$number <- as.numeric(d1ent$number)
d1ent$Model <- as.factor(d1ent$Model)

# Plot relative entropy values
ggplot(d1ent,
        aes(x=number, y=Rel_entropy,
            group=Model, color=Model,
            shape=Model)) +
        scale_shape_manual(values=1:nlevels(d1ent$Model))+
        geom_line(aes(linetype=Model),size=1)+
        geom_point(aes(shape=Model),size=3)+
        theme(
            legend.position = 'right',
```

```
            legend.spacing.x = unit(1, 'mm'),
            axis.title.x = element_blank(),
            plot.caption = element_text(color = "black"))+
        theme_minimal()+
        scale_x_continuous(name="Components", breaks=c(1:10))+
        ylab("Relative entropy")
```

To extract the highest Entropy value and model given a specified number of components like 3.3

```
table_wideent <- tidyr::spread(d1bic, number, Rel_entropy)

# for example
k <- '3' # specify number of components

## generate name of model
minc3ent <- table_wideent[which.min(table_wideent[,k]),'Model']
minc3ent

## generate entropy
minc3entval <- table_wideent[which.min(table_wideent[,k]),k]
minc3entval
```

ICL: extracting ICL values and visualize them for k components and models from which results are showed in a figure like figure 3.7

```
# fun ICL function for range of clusters
ICLoutput <- mclustICL(dfwithout.sc, G=1:5,
                       initialization = list(
                         hcPairs = hc(dfwithout.sc),
                         use = "SVD"))

# extra ICL values and make dataframe
ICL <- as.numeric(ICLoutput)
Model <- rep(attr(ICLoutput, "modelNames"), each = 5)
number <- rep(seq(1:5),14)
d1icl <- as.data.frame(cbind(ICL,number,Model))
d1icl$ICL <- as.numeric(d1icl$ICL)
d1icl$ICLneg <- (d1icl$ICL)*-1
d1icl$number <- as.factor(d1icl$number)
d1icl$Model <- as.factor(d1icl$Model)

ggplot(d1icl,
       aes(x=as.numeric(number), y=ICL,
           group=Model,color=Model,
```

```
        shape=Model)) +
    scale_shape_manual(values=1:nlevels(d1ent$Model))+
    geom_line(aes(linetype=Model),size=1)+
    geom_point(aes(shape=Model),size=3)+
    theme(
        legend.position = 'right',
        legend.spacing.x = unit(1, 'mm'),
        axis.title.x = element_blank(),
        plot.caption = element_text(color = "black"))+
    theme_minimal()+
    scale_x_continuous(name="Components", breaks=c(1:10))+
    ylab("ICL")
```

To extract the lowest BIC values and model given a specified number of components like 3.3

```
table_wideicl <- tidyr::spread(d1bic, number, ICLneg)

# for example
k <- '3' # specify number of components

## generate name of model
minc3icl <- table_wideicl[which.min(table_wideicl[,k]),'Model']
minc3icl

## generate ICL value
minc3iclval <- table_wideicl[which.min(table_wideicl[,k]),k]
minc3iclval
```

C.3.3 Analytical considerations

Model uncertainty: calculate mixture weights (probabilities) for each case and each k number of components, sort in increasing order and extract quantiles like table 3.4. The R-code includes a for loop so it can be applied to a range of components.

```
# specify range of components
minnumber <- 3
maxnumber <- 3

# specify modelname
modelname <- 'EVE'

# refer to dataset
dataset <- dfwithout.sc
```

```
# for loop extracting the uncertainties for each case
uncertaintytable<-c()
for (i in minnumber:maxnumber) {
  moduncertaint2 <- Mclust(dataset,
              G = i,
              modelnames=modelname,
              initialization =
              list(
                hcPairs = hc(dataset),
                use = "SVD",
                subset = NULL)
              )
    uncer_i <- 1 - apply(moduncertaint2$z, 1, max)
    uncer_i <-round(quantile(uncer_i),2)
    uncer_i <- as.data.frame(uncer_i)
    quant <- rownames(uncer_i)
    number <- rep(i, 5)
    uncer_i <- cbind(number, quant, uncer_i)
    colnames(uncer_i) <- c('components','quantiles','uncertainty')
    rownames(uncer_i) <- NULL
    uncer_i <- as.data.frame(uncer_i)
    uncertaintytable <- rbind(uncertaintytable,uncer_i)
    table_wide3eve <- tidyr::spread(uncertaintytable,
                                    components,
                                    uncertainty)
    table_wide3eve$quantiles <- as.factor(table_wide3eve$quantiles)
    table_wide3eve <- table_wide3eve[order(match(
      table_wide3eve$quantiles, c("0%","25%","50%","75%","100%"))),]
}

# constructing the table of quantiles
table_wide3eve <- as.data.frame(t(table_wide3eve))
colnames(table_wide3eve) <- table_wide3eve[1,]
table_wide3eve <- table_wide3eve[2:6,]
Components <- '3 EVE'
table_wide3eve <- cbind(Components,table_wide3eve)
table_wide3eve <- na.omit(table_wide3eve)
row.names(table_wide3eve)<-NULL
table_wide3eve
```

Visualize the general uncertainties like figure 3.8.

```
# run the mixture model
finmod <- Mclust(dfwithout.sc,
                 G = 3,
                 modelNames = 'EVE',
                 initialization = list(
                         hcPairs = hc(dfwithout.sc),
                         use = "SVD"))

uncerPlot(finmod$z)
```

Visualize the uncertainties by component like figure 3.9.

```
# extract mixture weights
probabilities <- as.data.frame(finmod$z)
colnames(probabilities) <- paste0('C', 1:3) # specify colnames

# convert from wide to long format
probabilitiesq <- probabilities %>%
  as.data.frame() %>%
  dplyr::mutate(id = row_number()) %>%
  tidyr::gather(cluster, probability, -id)

# make plot
ggplot(probabilitiesq, aes(probability)) +
       geom_histogram(fill="#2f3c4d")+
       facet_wrap(~ cluster, nrow = 1)+
       theme(
          legend.position = 'right',
          legend.spacing.x = unit(1, 'mm'),
          axis.title.x = element_blank(),
          plot.caption = element_text(color = "black"))+
       theme_minimal()+
       labs(subtitle='3-EVE')+
       ylab("Count")+
       xlab("Probability")
```

Cluster overlap: equivalent to K-Means clustering, we can visualize the data surface based on the mixture model like figure 3.11 (right).

```
fviz_mclust(finmod, "classification",
            geom = NULL,pointsize = 1.5,
            ellipse.type="convex",
            palette = c("#2f3c4d","#ad131b","#cc6a0e"))+
            labs(title=NULL, subtitle =  NULL) +
            ggtitle("Components 3-EVE model")
```

C.3.4 LPA model characteristics

Multivariate scatterplot: plot multivariate scatterplot where the colors specify the final components like figure 3.12

```
plot(finmod, what = "classification",
     col = c("#2f3c4d","#ad131b","#cc6a0e"))
```

Barplot: equivalent to K-Means clustering, a barplot including the standardized scores as a function of variable and cluster can be constructed like figure 3.13

```
# add result of LPA to final data (also used in K-Means clustering)
# by adding this, the finaldata dataset can be used for comparison
finaldata$LPA <- as.factor(finmod$classification)
finaldata$LPA <- as.factor(finaldata$LPA)
levels(finaldata$LPA) <- c('Bad','Good','Amotivated')

# select relevant variables for figure
finaldataLPA <- finaldata[,c('auto_mot','control_mot','amot','LPA')]

# check cluster sizes in proportion
round(prop.table(summary(finaldata$LPA)),2)

# generate calculated centroids of the clusters
datamean <- finaldataLPA %>%
    group_by(LPA) %>%
    summarise_all("mean")

# from wide to long dataset
data_long <- tidyr::gather(datamean, type, measurement,
                    c('auto_mot','control_mot','amot'),
                    factor_key=TRUE)
colnames(data_long) <- c('Clusters','Type','Standardized value')
data_long$Clusters <- as.numeric(data_long$Clusters)
data_long$Clusters <- rep(c(2,1,3),3)
data_long$Type <- factor(data_long$Type,
                         labels=c("Autonomous motivation",
                                  "Controlled motivation",
                                  "Amotivation"))

# Visualize in barplot
ggplot(data=data_long,
       aes(x=Clusters, y=`Standardized value`, fill=Type)) +
       geom_bar(stat="identity",width = 0.7,
                position=position_dodge(),color='black')+
```

```
      scale_fill_manual("Type of Motivation",
                        values = c("Autonomous motivation" = "#2f3c4d",
                                   "Controlled motivation" = "#ad131b",
                                   "Amotivation" = "#cc6a0e"))+
     theme(
         legend.position = 'right',
         legend.spacing.x = unit(1, 'mm'),
         axis.title.x = element_blank(),
         plot.caption = element_text(color = "black"))+
      scale_y_continuous(limits = c(-1.1,1.1))+
      scale_x_continuous(name="Cluster", breaks=c(1:3),
      labels= c("Cluster 1\n36%","Cluster 2\n43%","Cluster 3\n21%"))+
      theme_minimal()
```

Between-cluster differences: same code can be used like K-Mean clustering

C.4 Chapter 4: a Considerative Comparison

Contingency table: based on the `finaldata` dataset, contingency tables can be generated to compare both results as in table 4.1.

```
# make 2 way table
con.table <- with(finaldata, table(Kmean,LPA))

# in case the LPA variable is not in the same order
# of the K-Mean clustering, the order can be changed.
# This is only relevant when both methods become the same clusters.
con.table <- con.table[ , c("Good", "Bad", "Low")]

# row percentages
rows <- round(prop.table(con.table, 1),2)

# column percentages
columns <- round(prop.table(con.table, 2),2)
```

Chi-squared: perform Chi-squared test as in [Chi-squared].

```
chisq.test(con.table)
```

Cohen's kappa: when both clustering methods became the same clusters, a Kappa-index can be calculated to test their agreement as in [Cohen's kappa].

```
Kappa(con.table)
```

Biplot: run a CA to generate a symmetrical biplot like figure 4.2

```
fit <- ca(con.table)
fviz_ca_biplot(fit, repel = TRUE)+labs(title=NULL)
```

References

Akaike, H. (1973). Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki (Eds.), *Second international symposium on information theory* (pp. 267-281). Budapest: Academiai Kiado.

Allen, J. B. (2003). Social Motivation in Youth Sport. *Journal of Sport and Exercise Psychology, 25*(4), 551–567.

Arai, K., & Barakbah, A., R. (2007). Hierarchical K-means: an algorithm for centroids initialization for K-means. *Reports of the Faculty of Science and Engineering 36*(1), 25–31.

Assor, A., Vansteenkiste, M., & Kaplan, A. (2009). Identified versus introjected approach and introjected avoidance motivations in school and in sports: The limited benefits of self-worth strivings. *Journal of Educational Psychology, 101*(2), 482–497.

Bakker, M., & Wicherts, J. M. (2014). Outlier Removal and the Relation with Reporting Errors and Quality of Psychological Research. *PLoS ONE 9*(7), e103360.

Banfiel, J., & Raftery, A. (1992). Ice floe identification in satellite images using mathematical morphology and clustering about principal curves. *Journal of the American Stastistical Association, 87*, 7-16.

Bauer, D. J., & Curran, P. J. (2003). Distributional assumptions of growth mixture models: Implications for overextraction of latent trajectory classes. *Psychological Methods, 8*, 338–363.

Bechter, B. E., Dimmock, J. A., Howard, J. L., Whipp, P. R., & Jackson, B. (2018). Student motivation in high school physical education: A latent profile analysis approach.* Journal of Sport & Exercise Psychology, 40*(4), 206–216.

Bergman, L. R., & Wangby, M. (2014). The person-oriented approach: A short theoretical and practical guide. *Eesti Har*

Biernacki, C., Celeux, G., Govaert, G., (2000). Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 22*(7), 719–725.

Bradley, P., S., & Fayyad, U. (1998). "Refining Initial Points for K-Means Clustering", Proc. 15th International Conf. Machine Learning. Morgan Kaufmann.

Breckenridge, J. N. (2000). Validating cluster analysis: Consistent replication and symmetry. *Multivariate Behavioral Research, 35*, 261–285.

Calinski, T. and Harabasz, J. (1974) A Dendrite Method for Cluster Analysis Communications in Statistics. *Theory and Methods, 3*, 1-27.

Celeux, G., & Govaert, G. (1995). Gaussian parsimonious clustering models. *Pattern Recogn 28*(5), 781–793.

Chang, E., Lee, A., Byeon, E., & Lee, S.M. (2015). Role of motivation in the relation between perfectionism and academic burnout in Korean students. *Personality and Individual Differences, 82,* 221–226.

Charrad, M., Ghazzali, N., Boiteau, V., & Niknafs, A. (2014). NbClust Package for Determining the Best Number of Clusters. *R package version 2.0.3*, URL `http://CRAN.R-project.org/` package=NbClust.

Cheung, Y. M. (2003). K-Means: A new generalized k-means clustering algorithm. *Pattern Recognition Lett. 24,* 2883-2893.

Cohen, J. (1968). Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin, 70*(4), 213–220.

Cousineau, D., & Chartier, S. (2010). Outliers detection and treatment: A review. *International Journal of Psychological Research, 3*(1), 58–67.

Deci, E. L., & Ryan, R. M. (1985). *Intrinsic motivation and self-determination in human behavior.* New York, NY: Plenum.

Deci, E. L., Olafsen, A. H., & Ryan, R. M. (2017). Self-determination theory in work organizations: The state of a science. *Annual Review of Organizational Psychology and Organizational Behavior, 4,* 19–43.

Dempster, A. P., Laird, N., M., & Rubin, D., B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm, *Journal of the Royal Statistical Society, Series B., 29*, 1–37.

Duda, R. O., & Hart, P., E. (1973). *Pattern Classification and Scene analysis.* John Wiley and Sons, NY.

Efron, B. (1979). Bootstrap methods: another look at the jackknife. *Annals of Statistics, 7*, 1-26.

Elliot, A. J., Murayama, K., & Pekrun, R. (2011). A 3 × 2 achievement goal model. *Journal of Educational Psychology, 103*(3), 632–648.

Forgy, E. (1965). Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics, 21*(3), 768–769.

Growe, G. A. (1999). *Comparing Algorithms and Clustering Data: Components of The Data Mining Process*, thesis, department of Computer Science and Information Systems, Grand Valley State University.

Haerens, L., Kirk, D., Cardon, G. et al. (2010). Motivational Profiles for Secondary School Physical Education and Its Relationship to the Adoption of a Physically Active Lifestyle among University Students. *European Physical Education Review, 16*, 117-139.

Hair, J., F., Hult, G., T., M., Ringle, C., M., & Sarstedt, M. (2017). *A Primer on Partial Least Squares Structural Equation Modeling (PLS-SEM).* 2nd Ed. Thousand Oaks, CA: Sage

Hampel, F., R. (1974). The influence curve and its role in robust estimation. *Journal of the American Statistical Association, 69*(346), 383–393.

Hartigan, J., A., & M. A. Wong (1979). "Algorithm AS 136: A k-means clustering algorithm". *In: Applied Statistics 28.1*, pp. 100–108.

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

Hautamäki, V., Cherednichenko, S., Kärkkäinen, I., Kinnunen, T. & Fränti, P.

(2005). *Improving kmeans by outlier removal.* In H. Kalviainen et al. (Eds.): SCIA 2005, LNCS 3540, pp. 978–987.

Howard, J., Gagné, M., Morin, A. J. S., & Van den Broeck, A. (2016). Motivation profiles at work: A self-determination theory approach.*Journal of Vocational Behavior*, 95-96.

Kaufman, L., & P.J. Rousseeuw (1990). *Finding Groups in Data.* John Wiley & Sons, New York.

Kövesi, B., Boucher, J. M., & Saoudi, S. (2001).Stochastic K-means algorithm for vector quantization. *Pattern Recognition Letters, 22*, pp. 603–610.

Lawson, Richard G., & Peter C. J. (1990). New Index for Clustering Tendency and Its Application to Chemical Problems. *Journal of Chemical Information and Computer Sciences 30*(1), 36–41.

Lazarsfeld, P., F., & Henry, N., W. (1968). *Latent Structure Analysis.* Houghton Mifflin, New York.

Lee, S., X, & McLachlan, G. J. (2013) On mixtures of skew-normal and skew tdistributions. *Advances in Data Analysis and Classification* DOI 10.1007/ s11634-013-0132-8, preprint arXiv:1211.3602

Leys, C., Delacre, M., Mora, Y. L., Lakens, D., & Ley, C. (2019). How to Classify, Detect, and Manage Univariate and Multivariate Outliers, With Emphasis on Pre-Registration. *International Review of Social Psychology, 32*(1), 5.

Leys, C., et al. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology, 49,* (4), 764–766.

Lin, T., Lee, J., & Yen, S. (2007). Finite modelling using the skew normal distribution. *Statistica Sinica, 17*(3), 909-927. Retrieved August 23, 2020, from http://www.jstor.org/stable/24307705/par Liu, W. C., Wang, C., K., J., Kee, Y., H., Koh, C., Lim, B., S., C., & Chua, L. (2014). College students' motivation and learning strategies profiles and academic achievement: a self-determination theory approach. *Educational Psychology: An international journal of experimental educational psychology, 34,* (3), 338-353.

Lloyd, S. P. (1957). Least squares quantization in PCM.*Technical Report RR-5497, Bell Lab.*

Lloyd, S., P. (1982). Least squares quantization in pcm. *IEEE Transactions on Information Theory, 28,*(2), 129–137.

Lo Siou, G, et al. (2011). Exploring statistical approaches to diminish subjectivity of cluster analysis to derive dietary patterns: the tomorrow project. *Am J Epidemiol., 173*, (8), 956–967.

MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. In L. M. Le Cam & J. Neyman (Eds.), *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, 1, (pp. 281–297). California: University of California Press.

MacQueen, J., B. (1967). Some methods for classification and analysis of multivariate observations. In L. M. Le Cam and J. Neyman (Eds). *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability* (pp. 281–297). University of California Press.

Mahalanobis, P. C. (1930). On tests and measures of groups divergence, theoretical formulae. *International Journal of the Asiatic Society of Bengal, 26*, 541–588.

Maitra, R., & Melnykov, V. (2010) Simulating data to study performance of finite mixture modeling and clustering algorithms. *The Journal of Computational and Graphical Statistics, 2:19*, 354- 376.

Maslow, A. H. (1943). A theory of human motivation. *Psychological Review, 50*(4), 370–396.

McLachlan, G. J. (1987). On Bootstrapping the Likelihood Ratio Test Statistic for the Number of Components in a Normal Mixture. *Applied Statistics-Journal of the Royal Statistical Society Series C, 36,*(3) 318-324.

McLachlan, G. J., & Krishnan, K., (1997). *The EM Algorithm*. Wiley, New York.

Miller, J. (1991). Reaction time analysis with outlier exclusion: Bias varies with sample size. *The Quarterly Journal of Experimental Psychology, 43*(4), 907–912.

Milligan, G. W., & Cooper, M. C. (1985). An examination of procedures for determining the number of clusters in a data set. *Psychometrika*, 50, 159–179.

Moran, M., Diefendorff, J., Kim, T., & Liu, Z. (2012). A profile approach to self-determination theory motivations at work. *Journal of Vocational Behavior, 81*, 354–363.

Muthén, B. (2003). Statistical and Substantive Checking in Growth Mixture Modeling: Comment on Bauer and Curran (2003). *Psychological Methods, 8*(3), 369–377.

Peterson, R. A., & Cavanaugh, J., E. (2019). Ordered quantile normalization: a semiparametric transformation built for the cross-validation era. *Journal of Applied Statistics*, 1-16.

Robert, T., Guenther, W. R & Trevor, H. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society Series B, Royal Statistical Society, vol. 63*(2), 411-423.

Ryan, R. M., & Connell, J. P. (1989). Perceived locus of causality and internalization: Examining reasons for acting in two domains. *Journal of Personality and Social Psychology, 57*(5), 749–761.

Ryan, R. M., & Deci, E. L. (2017). *Self-determination theory: Basic psychological needs in motivation, development, and wellness*. Guilford Press.

Sander, J., Ester, M., Kriegel, H.-P. & Xu, X. (1998). Density-Based Clustering in Spatial Databases: The Algorithm GDBSCAN and Its Applications. *Data Mining and Knowledge Discovery, 2*, 169-194.

Schork, N. , & Schork, M. (1988). Skewness and mixtures of normal distributions. Communications in Statistics: *Theory and Methods, 17*, 3951-3970.

Schwarz, G. (1978) Estimating the Dimension of a Model. *Annals of Statistics, 6*, 461-464.

Scrucca, L. & Raftery, A.E. (2015). Improved initialisation of model-based clustering using a Gaussian hierarchical partition. *Advances in Data Analysis and Classification, 9*, 447-460.

Skinner, B. F. (1938). *The behavior of organisms: an experimental analysis*. Appleton-Century.

Symons, M. (1981). Clustering criteria and multivariate normal mixtures. *Bio-*

metrics, 37, 35-43.

Vansteenkiste, M., Sierens, E., Soenens, B., Luyckx, K., & Lens, W. (2009). Motivational profiles from a self-determination perspective: The quality of motivation matters. *Journal of Educational Psychology, 101*, 671-688.

Vermunt, J. K. (2011). K-means may perform as well as mixture model clustering but may also be much worse: Comment on Steinley and Brusco (2011). *Psychological Methods, 16*, 82-88.

Wang, J., C., K., Morin, A., J., S., Ryan, R., M., & Liu, W. C. (2016). Students' motivational profiles in the physical education context. *Journal of Sport & Exercise Psychology, 38*(6), 612–630.

Waterschoot. J., Koekelkoren, J., Soenens, & Vansteenkiste, M. (in progress). *A Taxonomical Research on Goals to participate a 100 km walk.*

Waterschoot, J., Vansteenkiste, M. & Soenens, B. (2019). The Effects of Experimentally Induced Choice on Elementary School Children's Intrinsic Motivation: The Moderating Role of Indecisiveness and Teacher-Student Relatedness. *Journal of Experimentally Child Psychology, 188*, 104692.

Wu, C., F., J. (1983). On the convergence properties of the EM algorithm. *Annals of Statistics, 11*,95–103.

Zhao, Q., M. Xu, & P. Franti, (2008). *Knee point detection on bayesian information criterion. Tools with Artificial Intelligence.* ICTAI '08. 20th IEEE International Conference, 2, 431–438.