kmeanss3: An R Package for K-Means Clustering

Friedrich Jaime Vogt, 4856443

02.04.2019

Contents

1 Introduction

The fast paced development of information technologies and the growth of their respective applications such as Internet search engines, digital imaging, and E-Commerce platforms has resulted in the creation of large databases with a variety of high-volume and multi-dimensional data sets (Crone et al., 2006). The volume of big data in worldwide data center storage is predicted to more than double from 179 exabytes at the current point of time to 403 exabytes in 2021 (Statista, 2019). Due to such data being stored digitally, for example in data warehouses or clouds, the analysis of said data is imperative to derive useful information and potential competitive advantages (Provost and Fawcett, 2013). While large amounts of data are gathered through different sources, such as web-trackers or transactions, the complexity of analyzing it is not to be underestimated due to most of this data being of unstructured nature (Jain, 2010).

Therefore, statistical models and analysis techniques are necessary to gain useful insights and realize said potentials. Many of these techniques are types of machine learning algorithms, such as regression and clustering models and can be used to recognize patterns or derive other inferences from the data (Buck et al., 2008). While regression models are part of supervised learning, which are methods to predict the values of outputs with a known dataset that includes response values (i.e. training sample), cluster analysis is a method of unsupervised learning (Friedman et al., 2001). The goal of unsupervised learning techniques is to draw inferences from a dataset that does not include labeled responses. Cluster analysis is used for exploratory data analysis and aims to group a collection of objects into subsets, natural groups or so-called clusters, such that those within each cluster are more closely related to each other than to objects of different clusters (Jain, 2010; Friedman et al., 2001). An object is described either by a set of measurements or by relationships between the object and other objects (Jain and Dubes, 1988). Cluster analysis can be used in market research to partition consumers into market segments, which may help with the selection of competitive markets, and product development strategies. (Sarstedt and Mooi, 2014). Another application is the grouping of search results based on topological features offered by systems such as Clusty and Lingo3G. These are used to cluster search results by using folder labels, for example borough, city, or state and may create a more relevant set of results for the end-users (Scaiella et al., 2012).

A large number of clustering algorithms exist, such as Expectation-Maximization Clustering, DBSCAN, and k-means (Xu and Wunsch, 2005). While each clustering method offers its own advantages and disadvantages, this paper will solely focus on the implementation of the k-means clustering algorithm in R. In the following chapter, the statistical and theoretical background of k-means will be introduced to demonstrate the functionality and methodology of the algorithm. Building on these foundations, the development and implementation of the R-package **kmeanss3** is going to be illustrated. Following this, the results of cluster analysis of the **IRIS** dataset using **kmeanss3** will be shown and compared with the results of *kmeans()* contained in the **stats** package of **CRAN**. Lastly, the implemented package will be discussed to underline both the contributions and limitations.

2 Background

In this section the theortical as well as statistical background of the k-means algorithm that is implented in the package **kmeanss3** is presented.

2.1 Theoretical Background

The term "k-means algorithm" was first coined by MacQueen (1967) for a sequential version of the algorithm. Data points x_i are processed in a sequential order $s = 1, 2, ...$ using the first k data points as "singleton classes", also known as centroids or centers. Data points are then assigned to the closest center from step s. In the following step, the centers are updated after each assignment (MacQueen, 1967). Despite MacQueen first coining the name, the algorithm has been proposed by several researchers in different forms and under different assumptions (Anderberg, 1973; Bock, 1974; Späth, 1975).

In the research fields of computer science and pattern recognition, k-means is often referred to as Lloyd's algorithm (Bock, 2007). Lloyd (1982) considers the sum of squares as clustering criterion in the context of pulse-code modulation. By minimizing the sum of squares, Lloyd is able to derive the optimality of centroids. He refers to his version of the k-means algorithm as "Method I", which despite only being publicly published in 1982, was first proposed in 1957 (Bock, 2007). In the implementation of **kmeanss3** Lloyd's algorithm plays a central role and is further detailed in Chapter 2.3.

2.2 Statistical Background

K-means is the best-known squared error criterion-based clustering algorithm and is used for variables of quantitative type (Xu and Wunsch, 2005; Bock, 2007). Let $X = (X_1, ..., X_n)$ be a set of n-dimensional points to be clustered into a set of K clusters $C = (C_1, ..., C_k)$. The algorithm finds a partition such that the squared error between the mean of a cluster and the points in the cluster is minimized (Jain, 2010). For this, the squared Euclidean distance is used as the distance measure. It is denoted as (Friedman et al., 2001):

$$d(x_i, x_i') = \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 = ||x_i - x_i'||^2 \ . \tag{1}$$

The within-cluster sum of squares (WSS) is given by:

$$WSS(C_k) = \sum_{x_i \in C_k} ||x_i - \mu_k||^2 \tag{2}$$

where $\mu_k = (\mu_{1k}, ..., \mu_{pk})$ is the mean vector of the kth cluster (Friedman et al, 2001). The WSS is to be minimized for each cluster C_k by assigning the nearest n observations to it. The WSS for a cluster C_k therefore represents the squared Euclidean distance from the center μ_k to each data point x_i within the cluster. In other words, WSS measures how closely related the objects within a cluster are, also known as cluster cohesion. Summarizing the WSS of each cluster returns the total-WSS.

Another important metric is the between cluster sum of squares (BSS). This metric measures the cluster separation, i.e. how distinct or well-seperated the created clusters are (Liu et al., 2010). The BSS is denoted as follows, with a weight n_k that represents the number of elements in the kth cluster:

$$BSS = \sum_{k=1}^{K} n_k ||\mu_k - \mu_i||^2 \tag{3}$$

Naturally, the BSS is maximized, when the WSS is minimized. This is due to the sum of the BSS and WSS being equal to the total sum of squares (TSS) (Liu et al., 2010). The TSS depicts the squared distance between each data point and the overall mean. Due to this, the TSS is a constant for every data set: it does not change when the number of centers or the initial centers are changed.

$$TSS = \sum_{i=1}^{N} ||x_i - \mu_i||^2 = WSS + BSS \tag{4}$$

The goodness of fit is measured by the ratio BSS/TSS and ranges between zero and one, with one representing the best possible fit. Furthermore, in an ideal scenario clusters have the properties of both internal cohesion and external seperation (Mehar et al., 2013). With this in mind, the goal of the k-means algorithm can be denoted as follows, where the goal is to assign each object to a cluster, so that the above can be achieved (Xu and Wunsch, 2005):

$$min \sum_{i=1}^{N} \sum_{k=1}^{K} \gamma_{ik} ||x_i - \mu_k||^2 \tag{5}$$

where

$$\gamma_{ik} = \begin{cases} 1, & \text{if } x_i \in C_k \\ 0, & \text{otherwise} \end{cases} . \tag{6}$$

Finding a solution for the algorithm is NP-hard, meaning that there is no known polynomial algorithm that can solve the problem, resulting in the computational time of said solution to grow exponentially with the size of the problem (Dasgupta and Freund, 2009). Additionally, k-means is a greedy algorithm, which may result in the solution not being a global optimum (Hartigan and Wong, 1979; Drineas et al., 2004). Despite this, k-means is popular due to its simplicity and fast local optimum convergence.

2.3 Algorithms

A fair amount of algorithms for k-means clustering exist, which heuristically attempt to create the "best" clusters. Commonly used ones are the previously mentioned algorithms of Lloyd (1982) and MacQueen (1967), as well as Hartigan and Wong's (1979) method. The main difference between the two latter and Lloyd's method is the calculation of centers. For both MacQueen as well as Hartigan and Wong's algorithm, the centers are updated every time a data point is moved to or from a cluster. With Lloyd's algorithm the cluster centers are only updated once all data points have been assigned, which is known as batch mode (Morissette and Chartier, 2013). Furthermore, Lloyd's method only iterates if a cluster has a point closer to some other cluster's center, while Hartigan and Wong's method takes the change of all centers resulting from a reassignment into account. This implies, that a data point may be reassigned despite already being assigned to the closest center (Telgarsky and Vattani, 2010). Despite a less efficient clustering process, Lloyd's algorithm provides satisfactory results in most cases. Therefore, to implement the k-means algorithm, Lloyd's method will be used. Jain (2010) summarizes the procedure of Lloyd's algorithm in four steps:

1. Initialize k-partition randomly or based on prior knowledge.
2. Assign each object in the data set to the nearest cluster.
3. Recalculate the centers.
4. Repeat 2. and 3. until clusters do not change.

3 Developement

In this section the development approach and implementation of **kmeanss3** is discussed.

3.1 Approach

For the implementation of the k-means algorithm a top-down approach was used. Firstly, a pseudo-code was created which is based on the previously mentioned four step approach of Jain (2010). In the fourth step, two measures are used to decide whether the clustering process has been completed or not. These are the maximum number of allowed iterations, and convergence, i.e. when the mean squared difference between old centers and newly assigned centers is smaller than the previously defined stop criterion. The pseudo-code was then used to develop the package contents in the R-Studio environment and is displayed below.

```
input:  data    x (matrix)
        centers k (number or matrix)
        iter.max
        stop.criterion

for inputs:
        assert numeric
        assert no NA

if k is numeric:
        if length of k < rows in x
          create k centers using sample()
else:
        if k < rows in x
          take input k as centers

iter = 0
converged = F

while iter < max iterations & converged = F & stop criterion < threshold:
  assign object to cluster where distance minimal
  update centers and current threshhold
  do until convergence (stop criterion > treshold)
```

3.2 Implementation

The implementation of **kmeanss3** is based on the background presented in Chapter 2. The previously mentioned statistical concepts as well as the algorithm were implemented in the programming language R and compiled as part of the package **kmeanss3**. For this, the S3-System, which is an object-oriented system within R (Wickham, 2015), was used to implement a print, summary, and plot method. Additionally, sanity checks were implemented to control both the input data as well as generated data, e.g. missing values, non-numeric values, more centers than data points, etc. Using control structures, in this case if- functions, the package offers the possibility of both specifying initial centers as well as the number of randomly generated centers. Additionally, it is possible to define the maximum number of iterations as well as the stop criterion. To assign points to the initial clusters as well as update the centers, a while-loop was implemented. The loop checks whether the criteria mentioned in 3.1 have been reached and furthermore uses the *clusterAssignment()* function to assign points to the nearest cluster based on the squared Euclidean distance. Lastly, using for-loops the metrics WSS and TSS are calculated. The BSS is derived from subtracting the WSS from TSS, for which the statistical background is mentioned in 2.2.

In this chapter, the undertaken approach as well as implementation process were demonstrated. By using the literature-review based foundations a pseudo- code for the k-means algorithm was created. The pseudo-code was presented and served as a basis to create the fundaments of the R code. The package **kmeanss3** uses control structures, functions as well as the S3-System, which also allows for visualization of the analysis results.

4 kmeanss3

This section explains how to obtain the package **kmeanss3**. Additionally, the functionalities of the package are presented.

4.1 Obtaining the software

The following will demonstrate how to obtain **kmeanss3**. First, the package must be downloaded and then installed using the *install.packages()* command. Additionally, **kmeanss3** requires the packages **scales** (Wickham, 2018). It can then be loaded into the environment using the command *library()* in the console or script window.

```
# install necessary package kmeanss3
install.packages("~/Desktop/Uni/Master/R-Seminar/kmeanss3/kmeanss3_1.0.tgz",
                 repos = NULL, type = .Platform$pkgType)

# load the package
library("kmeanss3")
```

4.2 Functionality

The **kmeanss3** package is used to cluster numeric data. The clustering method is k-means and Llodyd's algorithm is used to assign objects to clusters. The main function can be called using **myMeans()**, which also offers a help page. To demonstrate available compomenents, it is assumed that the results are saved under *results* (by using *results <- myMeans()*).

1. *myMeans()* - main function with inputs x, k, and optionally *maxIteration* and *stopCrit*.
2. *results$clusters* - returns a cluster vector which shows the assignment of data point i to cluster k.
3. *results$centers* - returns the centers (means) for each cluster k.
4. *results$totss* - returns the total sum of squares.
5. *results$withinss* - returns the within-cluster sum of squares for each cluster k.
6. *results$tot.withinss* - returns the sum of within-cluster sum of squares.
7. *results$betweenss* - returns the between cluster sum of squares.
8. *results$size* - returns the size of each cluster k.
9. *results$iter* - returns the number of iterations before convergence was reached.
10. *results$data* - returns the unclustered data input x.
11. *plot(results)* - returns an x-y plot in which data points are colored by cluster assignment and centers are highlighted. If input data has more than two dimensions, one can select which columns of the data are to be graphed.
12. *print(results)* - prints the key components of the cluster analysis.
13. *summary(results)* - returns a standard summary of input object.

5 Example: IRIS dataset

In this section the functionalities of **kmeanss3** will be demonstrated using the **IRIS** dataset. The results will be compared with the **CRAN** *kmeans()*.

5.1 Loading and description of data

For the example analysis the **IRIS** dataset from the **UCI Machine Learning Repository** (https://archive.ics.uci.edu/ml/datasets/iris) is used. The dataset is commonly used for pattern recognition and is well suited for cluster analysis. After obtaining the dataset it is loaded into the environment.

```
# loading the data
irisData <- read.csv("/Users/friedrichvogt/Desktop/Uni/Master/R-Seminar/iris.csv")
```

The structure of the dataset is as follows:

```
# data structure
str(irisData)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ sepal.length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
##  $ sepal.width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
##  $ petal.length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
##  $ petal.width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
##  $ class       : Factor w/ 3 levels "Iris-setosa",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
summary(irisData)
```

```
##   sepal.length    sepal.width     petal.length    petal.width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.054   Mean   :3.759   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##              class
##  Iris-setosa    :50
##  Iris-versicolor:50
##  Iris-virginica :50
##
##
##
```

The dataset contains 150 observations (n = 150) in three classes that represent different types of iris plants. For each of the classes (Iris-setosa, Iris-veriscolour, and Iris-virginica) 50 observations are recorded. The sepals and petals of the plants are each documented by width and length length in cm. As k-means clusters numeric values, the class will be excluded from analysis. Furthermore, to ensure two dimensions only the sepals will be clustered based on their documented width and length. The mean value for *sepal.length* is 5.84, and for *sepal.width* 3.05. With medians of 5.8 and and 3.0 for the latter.

5.2 K-means clustering and analysis of data

As previously mentioned, the first two columns of the **IRIS** dataset are analyzed. The analysis results are compared with the **CRAN** kmeans. *kmeans()* offers four algorithms, which have been previously discussed. As **kmeanss3** uses Lloyd's algorithm, this must also be specified in *kmeans()* function as an argument. Lastly, the results of **kmeanss3** are plotted with the built-in S3-method. To ensure that all results are reproducible *set.seed(2019)* is used.

```
# using kmeanss3
set.seed(2019)
useMyMeans <- myMeans(irisData[1:2], 4)
useMyMeans
```

```
##
## K-means clustering with 4 clusters of sizes 43 25 31 51
##
## Cluster means:
##      sepal.length sepal.width
## [1,]     6.853488    3.100000
## [2,]     4.772000    2.900000
## [3,]     5.196774    3.638710
## [4,]     5.909804    2.735294
##
## Clustering vector:
##   [1] 3 2 2 2 3 3 2 3 2 2 3 3 2 2 3 3 3 3 3 3 3 3 3 3 2 3 3 3 2 2 3 3 3 2
##  [36] 2 3 2 2 3 3 2 2 3 3 2 3 2 3 3 1 1 1 4 1 4 1 2 1 2 2 4 4 4 4 1 4 4 4 4
##  [71] 4 4 4 4 1 1 1 1 4 4 4 4 4 4 4 4 1 4 4 4 4 4 4 2 4 4 4 4 2 4 1 4 1 4 1
## [106] 1 2 1 1 1 1 4 1 4 4 1 1 1 1 4 1 4 1 4 1 1 4 4 4 1 1 1 4 4 4 1 1 1 4 1
## [141] 1 1 4 1 1 1 4 1 1 4
##
##
## Within cluster sum of squares by cluster:
## [1] 11.426977  4.510400  4.543226  7.481569
##
## (between_SS / total_SS = 78.5%)
##
## Available components:
## [1] "clusters"     "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "data"
```

A total of four clusters with sizes 43, 25, 31, and 51 have been created. The initial selection of clusters was randomized. The output presents the cluster means, a cluster vector, the WSS as well as goodness of fit. Output values will not be rounded, to ensure fair comparison with the output of *kmeans()*.

The results of the analysis show that the created clusters are good, with 78.5% goodness of fit. The cluster means for *sepal.length* are 6.85, 4.77, 5.19, and 5.90, while the cluster means for *sepal.width* are 3.10, 2.90, 3.63, and 2.73. Considering the mean values of 5.84 for *sepal.length* and 3.0 for *sepal.width*, the clusters are close to the means. This is also underlined by the WSS values of 11.42, 4.51, 4.54 and 7.48 in order of clusters. Despite this, one can see a dispersion in the values of WSS, especially when comparing the first and last cluster with the second and third. This is due to the cluster means for the first cluster being further apart (6.85 and 3.10) than e.g. the cluster means of the second cluster (4.77 and 2.90).

Thus, clusters two and three show less dispersion than clusters one and four and could therefore be considered "better", as the goal of k-means is to minimize the distance.

The results of *kmeans()* are presented below.

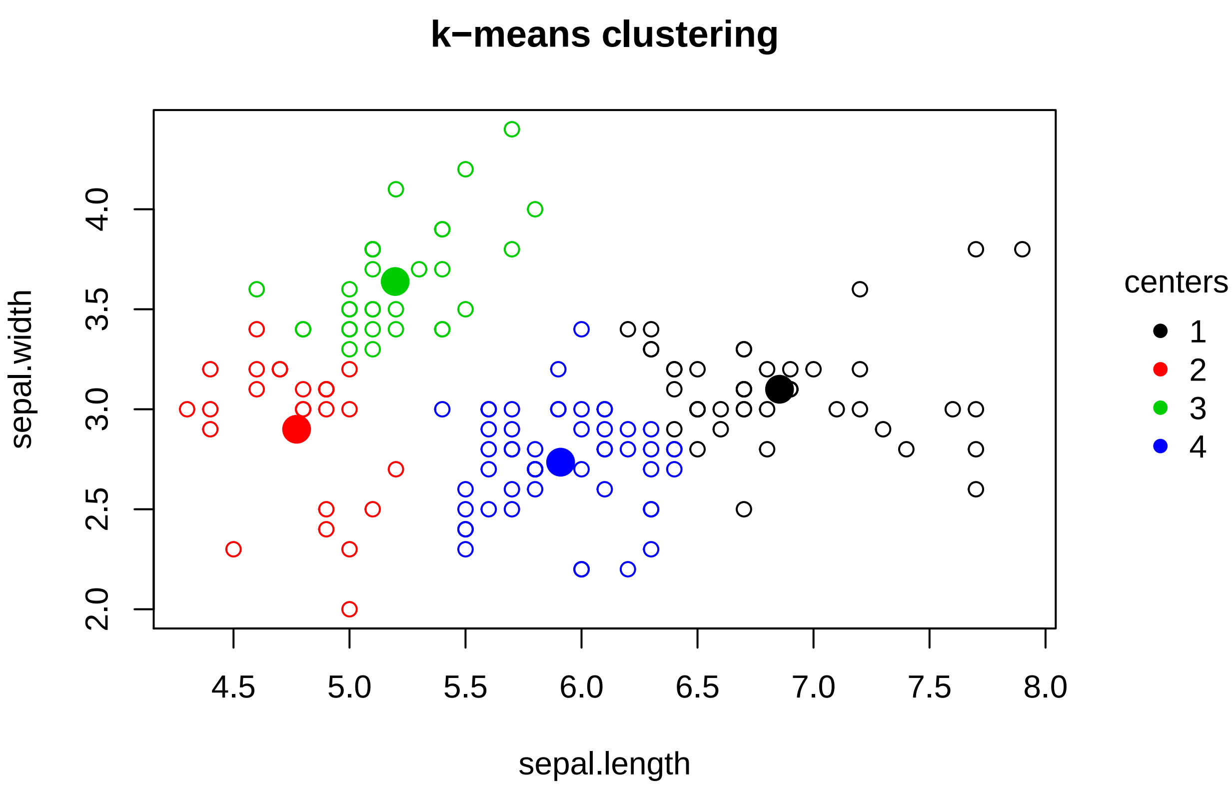
```r
# using base kmeans
set.seed(2019)
kMeansCRAN <- kmeans(irisData[1:2], 4, algorithm = "Lloyd")
kMeansCRAN
```

```
## K-means clustering with 4 clusters of sizes 43, 25, 31, 51
##
## Cluster means:
##   sepal.length sepal.width
## 1     6.853488    3.100000
```

```
## 2     4.772000    2.900000
## 3     5.196774    3.638710
## 4     5.909804    2.735294
##
## Clustering vector:
##   [1] 3 2 2 2 3 3 2 3 2 2 3 3 2 2 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 2 2 3 3 3 2
##  [36] 2 3 2 2 3 3 2 2 3 3 2 3 2 3 3 1 1 1 4 1 4 1 2 1 2 2 4 4 4 4 1 4 4 4 4
##  [71] 4 4 4 4 1 1 1 1 4 4 4 4 4 4 4 4 1 4 4 4 4 4 4 2 4 4 4 4 2 4 1 4 1 4 1
## [106] 1 2 1 1 1 1 4 1 4 4 1 1 1 1 4 1 4 1 4 1 1 4 4 4 1 1 1 4 4 4 1 1 1 4 1
## [141] 1 1 4 1 1 1 4 1 1 4
##
## Within cluster sum of squares by cluster:
## [1] 11.426977  4.510400  4.543226  7.481569
##  (between_SS / total_SS =  78.5 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"         "iter"
## [9] "ifault"
```

The results align with the results of **kmeanss3**, specifically the *myMeans()* function.

```r
# plotting the results
plot(useMyMeans)
```



The results of the k-means cluster analysis using **kmeanss3** can be plotted as seen above. The plot visualizes the results, which show that four distinct clusters have been generated. The cluster means are highlighted by enlarged points. From the plot one can see, that the colored data points are scattered closely around their cluster's center. The previously presented analysis results are further underlined. While a larger dispersion of data points around cluster one can be seen, the points assigned to other clusters are less scatted around the centers of their respective clusters.

This section of the paper presented how to obtain the package as well as the functionalities of **kmeanss3**. The package was used to group the two variables *sepal.length* and *sepal.width* of the **IRIS** dataset into four clusters. The results were discussed and then compared with the **CRAN** *kmeans()* to show equality. These were then visualized using the built-in plot method of **kmeanss3**.

6 Discussion

In this paper the developement as well as functionalities of the R-package **kmeanss3** were presented. Based on statistical and theoretical foundations derived from an extensive literature-review, the implementation was described. The package can be used to perform k-means clustering based on Lloyd's (1982) algorithm. It is possible to visualize the results using *plot()*. The package uses functions, control structres, and S3-system methods. The functionality of the package was presented using a free publicly-available dataset. The presented results aligned with the benchmark set by the **CRAN** *kmeans()* algorithm.

In future development the package could be extended to provide further functionalities. Firstly, more clustering algorithms could be added to offer more variability regarding the clustering process, as is done in *kmeans()*. Second, a random initialization of centers is done when k is a number. A commonly used method is to do n random initializations and select the centers with the smallest within-cluster sum of squares. *kmeans()* offers the argument *n.iter*, which can be used to provide better results based on n iterations of the initial centers. For future changes, this could be implemented in **kmeanss3** to provide better clusters.

7 References

ANDBERG, M.R. (1973): Cluster analysis for applications. Academic Press, New York.

BOCK, H. H. 1974. Automatische Klassifikation: theoretische und praktische Methoden zur Gruppierung und Strukturierung von Daten (Cluster-Analyse), Vandenhoeck & Ruprecht.

BUCK, C., GASS, T., HANNIG, A., HOSANG, J., JONAS, S., PETER, J.-T., STEINGRUBE, P. & ZIEGELDORF, J. H. 2008. Data-Mining-Cup 2007. Informatik-Spektrum, 31, 591-599.

CRONE, S. F., LESSMANN, S. & STAHLBOCK, R. 2006. The impact of preprocessing on data mining: An evaluation of classifier sensitivity in direct marketing. European Journal of Operational Research, 173, 781-800.

DASGUPTA, S. & FREUND, Y. 2009. Random projection trees for vector quantization. IEEE Transactions on Information Theory, 55, 3229-3242.

DRINEAS, P., FRIEZE, A., KANNAN, R., VEMPALA, S. & VINAY, V. 2004. Clustering large graphs via the singular value decomposition. Machine learning, 56, 9-33.

FRIEDMAN, J., HASTIE, T. & TIBSHIRANI, R. 2001. The elements of statistical learning, Springer series in statistics New York.

HANS-HERMANN, B. 2008. Origins and extensions of the k-means algorithm in cluster analysis. Journal Electronique d'Histoire des Probabilités et de la Statistique Electronic Journal for History of Probability and Statistics, 4, 48-49.

HARTIGAN, J. A. & WONG, M. A. 1979. Algorithm AS 136: A k-means clustering algorithm. Journal of the Royal Statistical Society. Series C (Applied Statistics), 28, 100-108.

JAIN, A. K. 2010. Data clustering: 50 years beyond K-means. Pattern recognition letters, 31, 651-666.

JAIN, A. K. & DUBES, R. C. 1988. Algorithms for clustering data.

LIU, Y., LI, Z., XIONG, H., GAO, X. & WU, J. Understanding of internal clustering validation measures. 2010 IEEE International Conference on Data Mining, 2010. IEEE, 911-916.

LLOYD, S. 1982. Least squares quantization in PCM. IEEE transactions on information theory, 28, 129-137.

MACQUEEN, J. Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley symposium on mathematical statistics and probability, 1967. Oakland, CA, USA, 281-297.

MEHAR, A. M., MATAWIE, K. & MAEDER, A. Determining an optimal value of K in K-means clustering. 2013 IEEE International Conference on Bioinformatics and Biomedicine, 2013. IEEE, 51-55.

MORISSETTE, L. & CHARTIER, S. 2013. The k-means clustering technique: General considerations and implementation in Mathematica. Tutorials in Quantitative Methods for Psychology, 9, 15-24.

PROVOST, F. & FAWCETT, T. 2013. Data science and its relationship to big data and data-driven decision making. Big data, 1, 51-59.

UCI. 2019. Iris Data Set [Online]. Available: https://archive.ics.uci.edu/ml/datasets/iris [Accessed 16.03.2019].

SARSTEDT, M. & MOOI, E. A concise guide to market research.

SCAIELLA, U., FERRAGINA, P., MARINO, A. & CIARAMITA, M. Topical clustering of search results. Proceedings of the fifth ACM international conference on Web search and data mining, 2012. ACM, 223-232.

SPÄTH, H. 1975. Cluster-Analyse-Algorithmen zur Objektklassifizierung und Datenreduktion. Verfahren der Datenverarbeitung, Muenchen: R. Oldenbourg, 1975.

STATISTA. 2019. Volume of big data in data center storage worldwide from 2015 to 2021 (in exabytes) [Online]. Available: https://www.statista.com/statistics/638621/worldwide-data-center-storage-used-by-big-data/ [Accessed 15.03.2019].

TELGARSKY, M. & VATTANI, A. Hartigan's method: k-means clustering without voronoi. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, 2010. 820-827.

WICKHAM, H. 2015. R packages: organize, test, document, and share your code, " O'Reilly Media, Inc.".

WICKHAM, H. 2018. Package 'scales'.

XU, R. & WUNSCH, D. C. 2005. Survey of clustering algorithms.