clustMixType: User-Friendly Clustering of Mixed-Type Data in R

by Gero Szepannek

Abstract Clustering algorithms are designed to identify groups in data where traditionally the emphasis in research has been on numeric data and in consequence most traditional algorithms are devoted to this kind of data. A gap has uprised from industrial praxis by the fact that in most business applications the variables are of both types: numeric and categorical. In recent past an increasing number of algorithms for clustering of mixed-type data has been proposed. Many of these are based on the idea of Huang's k-prototypes algorithm. This paper describes the R package **clustMixType** which provides an implementation of this algorithm in R.

Introduction

Clustering algorithms are designed to identify groups in data where traditionally the emphasis in research has been on numeric data and in consequence most traditional algorithms are devoted to this kind of data. A gap has uprised from industrial praxis by the fact that in most business applications the variables are of both types: numeric and categorical (for an example in the context of credit scoring, see e.g. Szepannek, 2017). A standard way to tackle mixed-type data clustering problems in R is to use Gower distance (Gower, 1971), e.g. using the packages **gower** (van der Loo, 2017) or the function `daisy(method = "gower")` of the **cluster** package (Maechler et al., 2018) together with hierarchical clustering which can be done e.g. by `hclust()` or **cluster**'s `agnes()`. The package **CluMix** (Hummel et al., 2017) combines both together with some functions for visualization. As this approach requires computation of distances between any two observations it is not feasible for large data sets. The package **flexclust** (Leisch, 2006) offers a flexible framework for k-centroids clustering by the function `kcca()` which allows for an extension by arbitrary distance functions but among the currently preimplemented kccaFamilies there is no distance for mixed-type data. Alternative approaches based on expectation-maximization are given by the function `flexmixedruns()` of the **fpc** package (Hennig, 2018) and the package **clustMD** (McParland, 2017). Both require the variables in the data set to be ordered according to their data type as well a preprocessing of the categorical variables to be coded by integers (1, 2,...). The clustMD algorithm (McParland and Gormley, 2016) further allows to take into account for ordinal variables but is quite computationally intensive. The **kamila** package (Foss and Markatou, 2018) implements the KAMILA clustering algorithm which uses a kernel density estimation technique in the continuous domain and a multinomial model in the categorical domain as well as an approach of Modha-Spangler weighting of variables where categorical variables have to be transformed into dummies in advance (Modha and Spangler, 2003).

In recent past an increasing number of algorithms for clustering of mixed-type data has been proposed in literature (Amir and Dey, 2007; Dutta et al., 2012; Foss et al., 2016; He et al., 2005; HajKacem et al., 2016; Ji et al., 2012, 2013, 2015; Lim et al., 2012; Liu et al., 2017; Pham et al., 2011). Many of these are based on the idea of Huang's k-prototypes algorithm (Huang, 1998). The rest of this paper describes the R package **clustMixType** (Szepannek, 2018) which provides up to the author's knowledge the first implementation of this algorithm in R. The k-modes algorithm (Huang, 1997a) has been implemented in the package **klaR** (Weihs et al., 2005; Roever et al., 2018) for data frames of categorical variables only. The rest of the paper is organized as follows: A brief description of the algorithm is followed by the functions that are provided by the **clustMixType** package. Afterwards, some extensions to the original algorithm are discussed and an example application is given.

k-prototypes clustering

The k-prototypes algorithm belongs to the family of partitional cluster algorithms. Its objective function is given by:

$$E = \sum_{i=1}^{n} \sum_{j=1}^{k} u_{ij} d\left(x_i, \mu_j\right), \tag{1}$$

where $x_i, i = 1, \ldots, n$ are the observations in the sample, $\mu_j, j = 1, \ldots, k$ are the cluster prototype observations and u_{ij} are the elements of the binary partition matrix $U_{n \times k}$ satisfying $\sum_{j=1}^{k} u_{ij} = 1, \forall\, i$.

The distance function is given by:

$$d(x_i, \mu_j) = \sum_{m=1}^{q} \left(x_i^m - \mu_j^m\right)^2 + \lambda \sum_{m=q+1}^{p} \delta\left(x_i^m, \mu_j^m\right). \qquad (2)$$

where m is an index over all variables of the data set with the first q variables of metric type and the remaining $p - q$ ones categorical and $\delta(a, b) = 0$ for $a = b$ and $\delta(a, b) = 1$ for $a \neq b$. $d()$ corresponds to weighted sum of Euclidean distance between two points in the metric space and simple matching distance (i.e. the count of mismatches) for categorical variables. The trade off between both terms can be controlled by the parameter λ which has to be prespecified in advance as well as the number of clusters k: for larger values of λ the impact of the categorical values increases and for $\lambda = 0$ it vanishes and only numeric variables are taken into account just as for traditional k-means clustering.

The algorithm iterates similar to the k-means algorithm (MacQueen, 1967) where for the numeric variables the mean and for the categorical the mode minimizes the total within cluster distance.

The steps of the algorithm are:

1. Initialization with random cluster prototypes.
2. For each observation do:
   (a) Assign observations to its closest prototype according to $d()$.
   (b) Update cluster prototypes by cluster-specific means/modes for all variables.
3. As long as any observations have swapped their cluster assignment in 2 or the maximum number of iterations has not been reached: repeat from 2.

k-prototypes in R

An implementation of the k-prototypes algorithm is given by the function

```
kproto(x, k, lambda = NULL, iter.max = 100, nstart = 1, na.rm = TRUE)
```

where `x` is a data frame with both numerics and factors. As opposed to many other packages presented in the introduction no preprocessing of the factor variables has to be done in advance and the order of the variables does not matter.

`k` is the number of clusters which has to be pre-specified. Alternatively, it can also be a vector of observation indices or a data frame of prototypes with the same columns as `x`. If ever at the initialization or during the iteration process identical prototypes do occur the number of clusters will be reduced accordingly. `lambda` > 0 is a real valued parameter that controls the trade off between Euclidean distance for numeric variables and simple matching distance for factor variables for cluster assignment. If no λ is specified the parameter is set automatically based on the data and a heuristic using the function `lambdaest()`. Alternatively, also a vector of length `ncol(x)` can be passed to `lambda` (cf. discussion of extensions).

The algorithm stops if no observations swap their clusters anymore. Alternatively, just as in `kmeans()` a maximum number of iterations can be specified by the argument `iter.max`. Similar to k-means, the result of k-prototypes depends on its initialization. For this reason, the argument `nstart` can be set to a value > 1 in order to run k-prototypes multiple times. In this case the best solution (i.e. the one that minimizes E) is returned. Generally, the algorithm can deal with missing data but as a default `NA`s are removed by `na.rm = TRUE`.

By two additional arguments: `verbose` and `keep.data` it can be controlled whether information on missing values in the data should be printed as well as whether the original data should be stored in the output object (`keep.data = TRUE`). The latter is necessary for the default call of the `summary()` function but in case of large x it can be avoided to save memory.

The output is an object of class `"kproto"`. For convenience the elements are designed to be compatible with those of class `"kmeans"`: `cluster` is an integer vector of cluster assignments. The output element `centers` stores the prototypes in a data frame and `size` are the corresponding cluster sizes respectively. `withinss` returns the sum over all within cluster distances to the prototype for each cluster and `tot.withinss` is their sum over all clusters which corresponds to the objective function E. In addition, `dists` returns a matrix of all observations' distances to all prototypes in order to investigate the crispness of the clustering. The elements `lambda` and `iter` store the specified arguments of the function call. Finally, `trace` lists the objective function E as well as the number of swapped observations during the iteration process.

Unlike `kmeans` the `"kproto"` class is accompanied by both a corresponding predict and summary method. The `predict.kproto` method can be used to assign clusters to new data. Like many of its cousins is called by

```
predict(object, newdata)
```

where the output again consists of two elements: a vector `cluster` of cluster assignments and a matrix `dists` of all observations' distances to all prototypes.

The investigation of the result from a cluster analysis typically consists in identifying the differences between the clusters, i.e. in this specific case those of the k prototypes. For practical applications besides the cluster sizes it is further of interest to take into account the homogeneity of the clusters. For numeric variables this can e.g. be done by calling the R function `summary()`. For categorical variables the representativity of the prototypes is given their frequency distribution as e.g. obtained by `prop.table()`. The `summary.kproto()` method applies these functions to the variables conditional to the resulting clusters and returns a comparative results table of the clusters for each variable.

The summary can further be computed for new data by calling `summary(object,data)` where `data` are new data that will be predicted first. If no new data is specified (default: `data = NULL`) the function requires the `"kproto"` object to contain the original data (argument `keep.data = TRUE`). In addition a function

```
clprofiles(object, x, vars = NULL)
```

supports the analysis of the clusters by visualization of cluster profiles based on an `object` of class kproto and data `x`. Note that the latter may also have different variables compared to the original data e.g. for profiling of variables that were not used for clustering. As opposed to `summary.kproto` no new prediction is done but the cluster assignments of the kproto object given by `object$cluster` are used. For this reason, the observations in `x` must be the same as in the original data. Via the `vars` argument a subset of variables can be specified either by a vector of indices or variable names. Note that `clprofiles()` is not restricted to objects of class `"kproto"` but can also be applied to other cluster objects as long as they are of a kmeans-like structure with elements `cluster` and `size`.

Extensions to the original algorithm

Typical for unsupervised clustering problems the user has to specify the impact of the specific variables on the desired cluster solution which is controlled by the parameter λ. Generally, small values $\lambda \sim 0$ emphasize numeric variables and will lead to results similar to standard k-means whereas larger values of λ lead to an increased influence of categorical variables. In Huang (1997b) the average standard deviation σ of the numeric variables is suggested for the choice of λ to be based on: in a simulation study values of $\frac{1}{3}\sigma \leq \lambda \leq \frac{2}{3}\sigma$ are used. The function

```
lambdaest(x, num.method = 1, fac.method = 1, outtype = "numeric")
```

provides different data based heuristics for the choice of λ: The average variance σ^2 (`num.method = 1`) or standard deviation σ (`num.method = 2`) over all numeric variables is related to the average concentration h_{cat} of all categorical variables which is computed by the averaging either $h_m = 1 - \sum_c p_{mc}^2$ (`cat.method = 1`) or $h_m = 1 - \max_c p_{mc}$ (`cat.method = 2`) over all variables m where c are the categories of the factor variables. $\lambda = \frac{\sigma^t}{h_{cat}}, t \in \{1, 2\}$ can be used as a user-friendly automatic choice of the parameter in order to prevent from over-emphasizing either numeric or categorical variables. If `kproto()` is called without specification of `lambda` the parameter is automatically set using `num.method = cat.method = 1`. Nevertheless, this should be considered rather as a starting point for further analysis and the explicit choice of λ should be done carefully based on the application context.

Originally, λ is real-valued, but in order to up- or downweight the relevance of single variables in a specific application context the function `kproto()` is extended to accept also vectors as input where each element corresponds to a variable specific weight λ_m. The formula for distance computation changes into:

$$d(x_i, p_j) = \sum_{m=1}^{q} \lambda_m \left(x_i^m - \mu_j^m\right)^2 + \sum_{m=q+1}^{p} \lambda_m \delta \left(x_i^m, \mu_j^m\right). \tag{3}$$

Note that the choice of λ only affects the assignment step for the observations but not the computation of the prototype given a cluster of observations. By changing the `outtype` argument into `"vector"` the function `lambdaest()` returns a vector of λ_ms. In order to support a user-specific definition of λ based on the variables' variabilities `outtype = "variation"` returns a vector of original values of variability for all variables in terms of the quantities described above.

cluster	numeric	categorical
1	+	+
2	+	-
3	-	+
4	-	-

Table 1: Separation of clusters in the example.

An issue of great practical relevance is the ability of an algorithm to deal with missing values which can be solved by an intuitive extension of k-prototypes. During the iterations cluster prototypes can be updated by ignoring `NA`s: Both means for numeric variables as well as modes for factors can be computed based on the available data. Similarly, for cluster assignment distances can be computed for each observation based on the available variables only. This not only allows to assign clusters to observations with missing values but already to take these observations into account when the clusters are formed. Using `kproto()` this can be obtained by setting the argument `na.rm = FALSE`. Note that in practice this option should be handled with care unless the number of missings is very small, e.g. the representativeness of a prototype might become questionable if its means and modes do not represent the major part of the cluster′s observations.

A modification of the original algorithm (cf. above) has been made allowing for vector-wise computation of the iteration steps and thus for an reduction in computation time. The update of the prototypes is not done after each new cluster assignment but once every time the whole data have been reassigned. The modified k-prototypes algorithm consists of the following steps:

1. Initialization with random cluster prototypes.
2. Assign all observations to its closest prototype according to $d()$.
3. Update cluster prototypes.
4. As long as any observations have swapped their cluster assignment in 2 or the maximum number of iterations has not been reached: repeat from 2.

Example

As an example data `x` with two numeric and two categorical variables are simulated according to the documentation ?`kproto` (using `set.seed(42)`): Four clusters (subscript $j = 1, \ldots, 4$) are designed such that two pairs can be separated only by the numeric variables and the other two pairs only by the categorical variables: The numeric variables are generated as normally distributed random variables with cluster specific means $\mu_1 = \mu_2 = -\mu_3 = -\mu_4 = \Phi^{-1}(0.9)$ and the categorical variables have two levels (A and B) each of them with a cluster specific probability $p_1(A) = p_3(A) = 1 - p_2(A) = 1 - p_4(A) = 0.9$. Table 1 summarizes the clusters. It can be seen that both numeric and categorical variables are needed in order to identify all four clusters.

Given the knowledge that there are four clusters in the data a straight forward call for k-prototypes clustering of the data will be:

```
kpres <- kproto(x = x, k = 4)
kpres                  # output 1
summary(kpres)         # output 2
library(wesanderson)
par(mfrow=c(2,2))
clprofiles(kpres, x, col = wes_palette("Royal1", 4, type = "continuous"))  # figure 1
```

The resulting output is of the form:

```
# Output 1:

Numeric predictors: 2
Categorical predictors: 2
Lambda: 5.52477

Number of Clusters: 4
Cluster sizes: 100 95 101 104
Within cluster error: 332.909 267.1121 279.2863 312.7261

Cluster prototypes:
    x1 x2         x3        x4
92   A  A  1.4283725  1.585553
54   A  A -1.3067973 -1.091794
205  B  B -1.4912422 -1.654389
272  B  B  0.9112826  1.133724

# Output 2 (only for variable x1 and x3):

x1

cluster     A     B
      1 0.890 0.110
      2 0.905 0.095
      3 0.069 0.931
      4 0.144 0.856

-----------------------------------------------------------------

x3
     Min. 1st Qu.  Median    Mean 3rd Qu.   Max.
1 -0.7263  0.9314  1.4080  1.4280  2.1280 4.5110
2 -3.4200 -1.9480 -1.3170 -1.3070 -0.6157 2.2450
3 -4.2990 -2.0820 -1.4600 -1.4910 -0.7178 0.2825
4 -1.5300  0.2788  0.9296  0.9113  1.5000 3.1480

-----------------------------------------------------------------
```

The first two (as well as the last two) cluster prototypes share the same mode in the factor variables but they can be distinguished by their location with respect to the numeric variables. For clusters 1 & 4 (as opposed to 2 & 3) it is vice versa. Note that of course the order of the identified clusters is not necessarily the same as in cluster generation. Calling the `summary` and `clprofiles` provides further information on the homogeneity of the clusters. A color palette can be passed to represent the clusters across the different variables for the plot, here it is taken from the package **wesanderson** (Ram and Wickham, 2018).

By construction, taking into account either only numeric (k-means) or only factor variables (k-modes) will not use all information necessary to identify the underlying cluster structure without further preprocessing of the data in this example. A performance comparison using Rand index (Rand, 1971) as computed by the package **clusteval** (Ramey, 2012) results in rand indices of 0.728 (k-means) and 0.733 (k-modes). As already seen above, the prototypes as identified by **clustMixType** do represent the true clusters quite well: the corresponding Rand index improves to 0.870.

```
library(klaR)
library(clusteval)

kmres  <- kmeans(x[,3:4], 4) # kmeans using numerics only
kmores <- kmodes(x[,1:2], 4) # kmodes using factors only

cluster_similarity(kpres$cluster, clusid, similarity = ”rand”)
cluster_similarity(kmres$cluster, clusid, similarity = ”rand”)
cluster_similarity(kmores$cluster, clusid, similarity = ”rand”)
```

The runtime of `kproto` is linear in the number of observations (Huang, 1997b) and thus it is also applicable to large data sets. Figure 2 (left) shows the behaviour of the runtime for 50 repeated

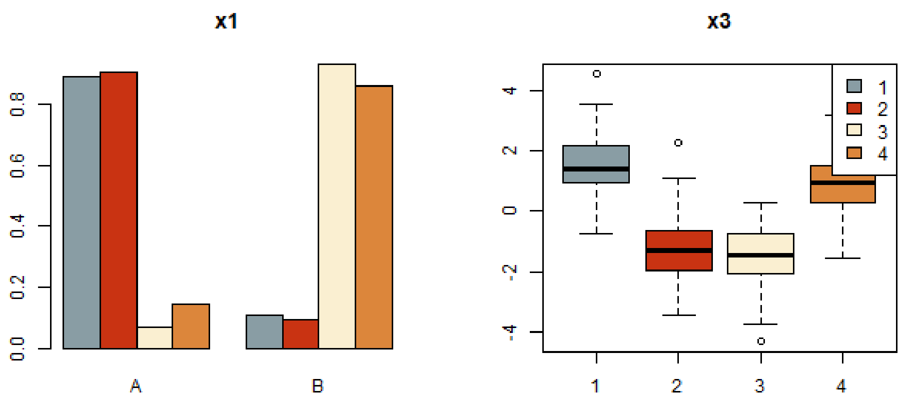

Figure 1: Cluster profiles for variables x1 and x3.

simulations of the example data with an increased number of variables (half of them numeric and half of them categorical): It is possible to run `kproto()` for more than hundred variables which is far beyond most practical applications where human interpretation of the resulting clusters is desired. Note that **clustMixType** is written in R and currently no C++ code is used to speed up computations which could be a subject of future work.

In order to determine the appropriate number of clusters for a data set we can use the standard scree test: In this case the objective function E is given by the output's `tot.withinss` element. `kproto()` is run multiple times for varying numbers of clusters (but fixed λ) and the number of clusters is chosen as minimum k from whereon no strong improvements of E are possible. In Figure 2 (right) an elbow is visible at the correct number of four clusters on the example data. Note that from a practitioner's point of view an appropriate solution requires clusters that are well represented by their prototypes. For this reason the choice of the number of clusters should further take into account a homogeneity analysis of the clusters as returned by `summary()`, `clprofiles()` or the `withinss` element of the `kproto` object.

```
Es <- numeric(10)
for(i in 1:10){
        kpres <- kproto(x, k = i, nstart = 5)
        Es[i] <- kpres$tot.withinss
}
plot(1:10, Es, type = "b", ylab = "Objective Function", xlab = "# Clusters",
     main = "Scree Plot")  # figure 2
```

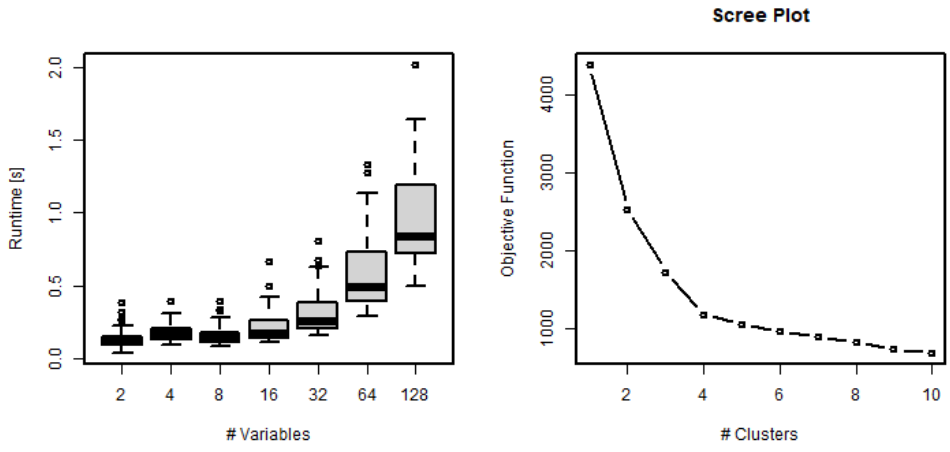

Figure 2: Runtime for increasing number of variables (left) and screeplot (right).

Summary

The **clustMixType** package provides a user-friendly way for clustering mixed-type data in R given by the k-prototypes algorithm. As opposed to other packages no preprocessing of the data is necessary and in contrast to standard hierarchical approaches it is not restricted to moderate data sizes. Its application requires the specification of two hyperparameters: the number of clusters k as well as a second parameter λ that controls the interplay of the different data types for distance computation. As an extension to the original algorithm the presented implementation allows for a variable-specific choice of λ and can further deal with missing data. Furthermore, with regard to business purposes functions for profiling a cluster solution are presented. The paper is based on **clustMixType** version 0.1-36. Future work may be devoted to the development of further guidance on the choice of the parameter λ, e.g. using stability considerations (cf. e.g. Hennig, 2007) or the investigation of possible improvements in computation time by the integration of **Rcpp** (Eddelbuettel et al., 2018; Eddelbuettel and François, 2011).

Bibliography

A. Amir and L. Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data and Knowledge Engineering*, 63:88–96, 2007. URL `https://doi.org/10.1016/j.datak.2007.03.016`. [p1]

D. Dutta, P. Dutta, and J. Sil. Data clustering with mixed features by multi objective genetic algorithm. In *12th International Conference on Hybrid Intelligent Systems (HIS)*, pages 336 – 341, 2012. URL `https://doi.org/10.1109/his.2012.6421357`. [p1]

D. Eddelbuettel and R. François. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL `https://doi.org/10.18637/jss.v040.i08`. [p7]

D. Eddelbuettel, R. Francois, J. Allaire, K. Ushey, Q. Kou, N. Russell, D. Bates, and J. Chambers. *Rcpp: Seamless R and C++ Integration*, 2018. URL `https://CRAN.R-project.org/package=Rcpp`. R package version 0.12.18. [p7]

A. Foss and M. Markatou. *Kamila: Methods for Clustering Mixed-Type Data*, 2018. URL `https://CRAN.R-project.org/package=kamila`. R package version 0.1.1.2. [p1]

A. Foss, M. Markatou, B. Ray, and A. Heching. A semiparametric method for clustering mixed data. *Machine Learning*, 105(3):419–458, 2016. URL `https://doi.org/10.1007/s10994-016-5575-7`. [p1]

J. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27:857–874, 1971. URL `https://doi.org/10.2307/2528823`. [p1]

M. A. B. HajKacem, C.-E. B. N'cir, and N. Essoussi. An accelerated mapreduce-based k-prototypes for big data. In P. Milazzo, D. Varro, and M. Wimmer, editors, *Software Technologies: Applications and Foundations 2016, Springer LNCS 9946*, pages 13–25, 2016. URL `https://doi.org/10.1007/978-3-319-50230-4_2`. [p1]

Z. He, X. Xu, and S. Deng. A cluster ensemble method for clustering categorical data. *Information Fusion*, 6, 2005. URL `https://doi.org/10.1016/j.inffus.2004.03.001`. [p1]

C. Hennig. Cluster-wise assessment of cluster stability. *Computational Statistics & Data Analysis*, 52: 258–271, 2007. URL `https://doi.org/10.1016/j.csda.2006.11.025`. [p7]

C. Hennig. *Fpc: Flexible Procedures for Clustering*, 2018. URL `https://CRAN.R-project.org/package=fpc`. R package version 2.1-11.1. [p1]

Z. Huang. A fast clustering algorithm to cluster very large categorical data sets in data mining. In *Proceedings of the SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery*, pages 1 – 8, 1997a. [p1]

Z. Huang. Clustering large data sets with mixed numeric and categorical values. In *Proceedings of the First Pacific Asia Knowledge Discovery and Data Mining Conference, Singapore*, pages 21 – 34, 1997b. [p3, 5]

Z. Huang. Extensions to the k-means algorithm for clustering large data sets with categorical variables. *Data Mining and Knowledge Discovery*, 2:283–304, 1998. URL `https://doi.org/10.1023/a:1009769707641`. [p1]

M. Hummel, D. Edelmann, and A. Kopp-Schneider. *CluMix: Clustering and Visualization of Mixed-Type Data*, 2017. URL https://CRAN.R-project.org/package=CluMix. R package version 2.1. [p1]

J. Ji, W. Pang, C. Zhou, X. Han, and Z. Wang. A fuzzy k-prototype clustering algorithm for mixed numeric and categorical data. *Knowledge-Based Systems*, 30:129–135, 2012. URL https://doi.org/10.1016/j.knosys.2012.01.006. [p1]

J. Ji, T. Bai, C. Zhou, C. Maa, and Z. Wang. An improved k-prototypes clustering algorithm for mixed numeric and categorical data. *Neurocomputing*, 120:590–596, 2013. URL https://doi.org/10.1016/j.neucom.2013.04.011. [p1]

J. Ji, W. Pang, Y. Zheng, Z. Wang, Z. Ma, and L. Zhang. A novel cluster center initialization method for the k-prototypes algorithms using centrality and distance. *Applied Mathematics and Information Sciences*, 9:2933–2942, 2015. [p1]

F. Leisch. A toolbox for k-centroids cluster analysis. *Computational Statistics & Data Analysis*, 51(2): 526–544, 2006. URL https://doi.org/10.1016/j.csda.2005.10.006. [p1]

J. Lim, J. Jun, S. H. Kim, and D. McLeod. A framework for clustering mixed attribute type datasets. In *Proceeding of the Fourth International Conference on Emerging Databases (EDB)*, 2012. [p1]

S.-H. Liu, B. Zhou, D. Huang, and L. Shen. Clustering mixed data by fast search and find of density peaks. *Mathematical Problems in Engineering*, 2017:1–7, 2017. [p1]

J. MacQueen. Some methods for classification and analysis of multivariate observations. In L. Le Cam and J. Neyman, editors, *Proc. 5th Berkeley Symp. Math Stat and Prob*, pages 281 – 297, 1967. [p2]

M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik. *Cluster: Cluster Analysis Basics and Extensions*, 2018. URL https://CRAN.R-project.org/package=cluster. R package version 2.0.7-1. [p1]

D. McParland. *clustMD: Model Based Clustering for Mixed Data*, 2017. URL https://CRAN.R-project.org/package=clustMD. R package version 1.2.1. [p1]

D. McParland and I. Gormley. Model based clustering for mixed data: clustmd. *Advances in Data Analysis and Classification*, 10(2):155–170, 2016. URL https://doi.org/10.1007/s11634-016-0238-x. [p1]

D. Modha and S. Spangler. Feature weighting in k-means clustering. *Machine Learning*, 52(3):217–237, 2003. URL https://doi.org/10.1023/a:1024016609528. [p1]

D. Pham, M. Suarez-Alvarez, and Y. I. Prostov. Random search with k-prototypes algorithm for clustering mixed datasets. *Proceedings of The Royal Society A: Mathematical, Physical and Engineering Sciences*, 467:2387–2403, 2011. URL https://doi.org/10.1098/rspa.2010.0594. [p1]

K. Ram and H. Wickham. *Wesanderson: A Wes Anderson Palette Generator*, 2018. URL https://CRAN.R-project.org/package=wesanderson. R package version 0.3.6. [p5]

J. A. Ramey. *Clusteval: Evaluation of Clustering Algorithms*, 2012. URL https://CRAN.R-project.org/package=clusteval. R package version 0.1. [p5]

W. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66:846–850, 1971. URL https://doi.org/10.2307/2284239. [p5]

C. Roever, N. Raabe, K. Luebke, U. Ligges, G. Szepannek, and M. Zentgraf. *klaR: Classification and Visualization*, 2018. URL https://CRAN.R-project.org/package=klaR. R package version 0.6-14. [p1]

G. Szepannek. On the practical relevance of modern machine learning algorithms for credit scoring applications. *WIAS Report Series*, 29:88–96, 2017. URL https://doi.org/10.20347/wias.report.29. [p1]

G. Szepannek. *clustMixType: k-Prototypes Clustering for Mixed Variable-Type Data*, 2018. URL https://CRAN.R-project.org/package=clustMixType. R package version 0.1-36. [p1]

M. van der Loo. *Gower: Gower's Distance*, 2017. URL https://CRAN.R-project.org/package=gower. R package version 0.1.2. [p1]

C. Weihs, U. Ligges, K. Luebke, and N. Raabe. klar analyzing german business cycles. In D. Baier, R. Decker, and L. Schmidt-Thieme, editors, *Data Analysis and Decision Support*, pages 335–343, Berlin, 2005. Springer-Verlag. URL https://doi.org/10.1007/3-540-28397-8_36. [p1]

Gero Szepannek
Stralsund University of Applied Sciences
Zur Schwedenschanze 15
18435 Stralsund
Germany
gero.szepannek@hochschule-stralsund.de