Journal of Statistical Software

October 2016, Volume 74, Issue 9. doi: 10.18637/jss.v074.i09

ClickClust: An R Package for Model-Based Clustering of Categorical Sequences

Volodymyr Melnykov
The University of Alabama

Abstract

The R package **ClickClust** is a new piece of software devoted to finite mixture modeling and model-based clustering of categorical sequences. As a special kind of time series, categorical sequences, also known as categorical time series, exhibit a time-dependent nature and are traditionally modeled by means of Markov chains. Clustering categorical sequences is an important problem with multiple applications, but grouping sequences of sites or web-pages, also known as clickstreams, is one of the most well-known problems that helps discover common navigation patterns and routes taken by users. This popular application is recognized in the package title **ClickClust**. The paper discusses methodological and algorithmic foundations of the package based on finite mixtures of Markov models. The number of Markov chain states can often be large leading to high-dimensional transition probability matrices. The high number of model parameters can affect clustering performance severely. As a remedy to this problem, backward and forward selection algorithms are proposed for grouping states. This extends the original clustering problem to a biclustering framework. Among other capabilities of **ClickClust**, there are the estimation of the variance-covariance matrix corresponding to model parameter estimates, prediction of future states visited, and the construction of a display named click-plot that helps illustrate the obtained clustering solutions. All available functions and the utility of the package are thoroughly discussed and illustrated on multiple examples.

Keywords: categorical sequences, model-based cluster analysis, finite mixture models, Markov models, biclustering, click-plot, R.

1. Introduction

The objective of cluster analysis is to group data in such a way that each cluster contains data points with similar features while clusters are relatively distinct from one another. This area of machine learning is popular in statistics and computer science due to a large variety of applications. There are many clustering approaches considered in literature. Among them there are the well-known k-means (Forgy 1965; MacQueen 1967), k-medoids (Kaufman and Rousseuw 1990), and various hierarchical clustering algorithms (Sorensen 1948; Sneath 1957; Ward 1963). Model-based clustering (Banfield and Raftery 1993; Fraley and Raftery 2002; Melnykov 2013a) is a standalone technique of grouping data which assumes that each data cluster can be seen as a sample from some probability distribution. In case there are multiple data groups, several distributions are needed and finite mixture models (McLachlan and Peel 2000) should be employed. Then the task of grouping data is tied to the problem of finding the best fitting mixture model.

Model-based clustering is known for remarkably good performance in grouping complex objects. Among many challenging applications addressed by this technique, there are the analysis of social networks (Handcock, Raftery, and Tantrum 2007), mass spectrometry data (Melnykov 2013b), and text classification (Nigam, McCallum, Thrun, and Mitchell 2000). Some work has been done in the direction of model-based clustering of time series (Liao 2005) and regression time series (Chen and Maitra 2011; Melnykov 2012). Despite a high number of interesting applications that can be addressed by grouping categorical sequences, this area has been given very limited attention in the literature on model-based clustering. One such application is grouping sequences of sites or web-pages visited by customers. These sequences reflect the navigation behavior of users and are commonly referred to as clickstreams. The name of the software package **ClickClust**, which is considered in this paper, reflects the importance of this application. As pointed out by Cadez, Heckerman, Meek, Smyth, and White (2003), the main challenge in the cluster analysis of clickstreams is their categorical nature and dynamic behavior. The authors remark that these issues can be addressed with the use of finite mixtures with Markov model components. A recent paper by Melnykov (2016a) extends their work, provides a variability assessment procedure, and remarks that the number of parameters can be excessive when the number of states in a Markov model is high. The author develops backward and forward state selection algorithms that allow to cluster states along with clickstreams, thus transforming the problem into a biclustering framework.

As a result of limited attention to the analysis of categorical sequences, there is also a lack of software devoted to addressing this problem. The only piece of software with a specific focus on the analysis of categorical sequences by means of transition probabilities that the author of this paper is aware of is the recent contribution in the form of the R package **clickstream** (Scholz 2016a,b). The package offers the functionality for estimating the overall transition probability matrix as well as clustering sequences by means of the k-means algorithm applied to transition matrices constructed for all sequences. While clearly useful in many situations, k-means clustering implemented in **clickstream** is often oversimplistic due to the constraint imposed by the use of the Euclidean metric for measuring distance between cluster centers and data points. Other contributions important in the considered framework are R packages **TraMineR** (Gabadinho, Ritschard, Müller, and Studer 2011) and **WeightedCluster** (Studer 2013). These tools aim at the analysis of categorical sequences through distance (dissimilarity) measures rather than modeling state transition patterns. Some methods of assessing similarity in data rely on distances calculated based on the longest common prefix, suffix, and common subsequence as well as optimal matching. After the dissimilarity between sequences has been calculated, the user can apply various clustering procedures such as k-medoids or hierarchical clustering algorithms mentioned in the beginning of this section. Besides the above-mentioned R packages, tools for clustering categorical sequences through distances and dissimilarities such as **SQ** (Brzinsky-Fay, Kohler, and Luniak 2006) and **SADI** (Halpin 2014) are also available in Stata (StataCorp. 2015).

In this paper, we present an R (R Core Team 2016) package called **ClickClust** (Melnykov 2016b) that allows grouping of categorical sequences and running state selection procedures by means of finite mixture modeling and model-based clustering. Despite the fact that there is a variety of R packages devoted to modeling Markov processes (e.g., **depmixS4**, Visser and Speekenbrink 2010; **HiddenMarkov**, Harte 2016), they do not provide clustering capabilities similar to those of **ClickClust**. The core of **ClickClust** is written in C and the package is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=ClickClust`.

The paper is organized as follows. Section 2 briefly discusses the methodological aspects of finite mixtures with Markov model components, corresponding estimation and variability assessment procedures, state selection algorithms, state prediction, and visual representation of obtained results by means of click-plots. Section 3 provides a comprehensive description of functions available in **ClickClust** and illustrates their use on multiple demo examples included in the package. Section 4 focuses on demonstrating the utility of the package via a comprehensive analysis of a challenging simulated dataset. Section 5 is devoted to the analysis of a real-life clickstream dataset. The paper concludes with a brief summary provided in Section 6.

2. Methodological and algorithmic details

This section is devoted to describing methodological developments in model-based clustering of categorical sequences. It outlines the EM algorithm, state selection approaches, a variability assessment procedure, prediction of future locations and introduces a graphical display called click-plot.

2.1. Model-based clustering of categorical sequences

The fundamental assumption of model-based clustering is the ability to model each data group by means of a distribution of some pre-specified functional form. The probability distribution corresponding to a finite mixture model is given by

$$f(\boldsymbol{y}|\boldsymbol{\vartheta}) = \sum_{k=1}^{K} \alpha_k f_k(\boldsymbol{y}|\boldsymbol{\vartheta}_k), \tag{1}$$

where K is the total number of component distributions $f_k(\cdot|\boldsymbol{\vartheta}_k)$ with corresponding parameter vectors $\boldsymbol{\vartheta}_k$ and $\alpha_1, \ldots, \alpha_K$ are mixing proportions, subject to the restrictions $\alpha_k > 0$ and $\sum_{k=1}^{K} \alpha_k = 1$. $\boldsymbol{\vartheta} = (\alpha_1, \ldots, \alpha_{K-1}, \boldsymbol{\vartheta}_1^\top, \ldots, \boldsymbol{\vartheta}_K^\top)^\top$ represents the entire parameter vector that has to be estimated. The estimation of $\boldsymbol{\vartheta}$ is usually carried out by a two-stage iterative procedure called the expectation-maximization (EM) algorithm (Dempster, Laird, and Rubin 1977). The EM algorithm aims to incorporate missing or unavailable information. In the mixture modeling framework, it is assumed that each observation i comes from one of K component distributions and that its true membership label t_i is missing. The first stage of the EM algorithm is called the expectation step and aims to find the conditional expectation of the complete-data log-likelihood function given observed data. At the maximization step, this expectation has to be maximized with respect to the parameter vector $\boldsymbol{\vartheta}$. Upon convergence in parameters or log-likelihood values, the EM algorithm provides the maximum likelihood estimate (MLE) $\hat{\boldsymbol{\vartheta}}$ and the matrix of estimated posterior probabilities $(\hat{z}_{ik})_{n\times K}$, where n is the sample size and $\hat{z}_{ik}$ is the probability that the i-th observation $\boldsymbol{y}_i$ belongs to the k-th component distribution.

Let $\boldsymbol{Y}_i = (Y_{i1}, \ldots, Y_{iS_i})^\top$ represent the i-th categorical sequence of length S_i following the first-order Markov model with p unique states. Then, we can write $\mathsf{P}(\boldsymbol{Y}_i = \boldsymbol{y}_i) = \mathsf{P}(Y_{i1} = y_{i1}) \prod_{s=2}^{S_i} \mathsf{P}(Y_{is} = y_{is}|Y_{i(s-1)} = y_{i(s-1)})$, where y_{is}, $s = 1, \ldots, S_i$, takes values in $\{1, \ldots, p\}$ and represents the state observed in the s-th position of $\boldsymbol{y}_i$. To simplify our notation, we denote the initial state probability as $\beta_j = \mathsf{P}(Y_{i1} = j)$ and transition probability as $\gamma_{jj'} = \mathsf{P}(Y_{is} = j'|Y_{i(s-1)} = j)$, subject to the restrictions $\sum_{j=1}^p \beta_j = 1$ and $\sum_{j'=1}^p \gamma_{jj'} = 1$ for $j = 1, \ldots, p$. Now, noting that

$$\mathsf{P}(\boldsymbol{Y}_i = \boldsymbol{y}_i) = \mathsf{P}(Y_{i1} = y_{i1}) \prod_{s=2}^{S_i} \mathsf{P}(Y_{is} = y_{is}|Y_{i(s-1)} = y_{i(s-1)}) = \prod_{j=1}^{p} \beta_j^{I(y_{i1}=j)} \prod_{j=1}^{p}\prod_{j'=1}^{p} \gamma_{jj'}^{x_{ijj'}}, \quad (2)$$

where $I(\cdot)$ is the indicator function and $x_{ijj'}$ represents the frequency of transitions from state j to state j' within the i-th sequence, and assuming that each categorical sequence originates from one of the K components, Equation 1 can be written as

$$f(y_{i1}, \boldsymbol{X}_i|\boldsymbol{\vartheta}) = \sum_{k=1}^{K} \alpha_k \prod_{j=1}^{p} \beta_{kj}^{I(y_{i1}=j)} \prod_{j=1}^{p}\prod_{j'=1}^{p} \gamma_{kjj'}^{x_{ijj'}} \quad (3)$$

for a $p \times p$ matrix $\boldsymbol{X}_i$ with elements $x_{ijj'}$. As we can see from Equation 3, the information about the i-th sequence is summarized in terms of the first state observed and the transition frequency matrix, i.e., a pair $(y_{i1}, \boldsymbol{X}_i)$. While the information about the order of states is not preserved under such data representation, the pair $(y_{i1}, \boldsymbol{X}_i)$ is a minimal sufficient statistic for estimating parameters of the model given in (2).

It can be shown that the expectation step in this setting reduces to the calculation of posterior probabilities at the m-th iteration by

$$z_{ik}^{(m)} = \frac{\alpha_k^{(m-1)} \prod_{j=1}^{p} (\beta_{kj}^{(m-1)})^{I(y_{i1}=j)} \prod_{j=1}^{p}\prod_{j'=1}^{p} (\gamma_{kjj'}^{(m-1)})^{x_{ijj'}}}{\sum_{k'=1}^{K} \alpha_{k'}^{(m-1)} \prod_{j=1}^{p} (\beta_{k'j}^{(m-1)})^{I(y_{i1}=j)} \prod_{j=1}^{p}\prod_{j'=1}^{p} (\gamma_{k'jj'}^{(m-1)})^{x_{ijj'}}}$$

and the maximization step involves updating parameter estimates by

$$\alpha_k^{(m)} = \frac{1}{n}\sum_{i=1}^{n} z_{ik}^{(m)}, \qquad \beta_{kj}^{(m)} = \frac{\sum_{i=1}^{n} z_{ik}^{(m)} I(y_{i1} = j)}{\sum_{i=1}^{n} z_{ik}^{(m)}}, \qquad \gamma_{kjj'}^{(m)} = \frac{\sum_{i=1}^{n} z_{ik}^{(m)} x_{ijj'}}{\sum_{i=1}^{n} z_{ik}^{(m)} \sum_{r'=1}^{p} x_{ijr'}}.$$

In many applications, the mixture model order is unknown and thus K has to be estimated. Traditionally, the best K is detected by minimizing the Bayesian information criterion (BIC) (Schwarz 1978). In the considered setting, BIC is calculated as $-2\sum_{i=1}^{n} \log f(y_{i1}, \boldsymbol{X}_i|\hat{\boldsymbol{\vartheta}}) + N \log n$, where n represents the number of clickstream sequences and the number of model parameters is given by $N = Kp^2 - 1$. Melnykov (2016a) argued that some clusters obtained can reflect the initial state preference rather than navigation patterns associated with transitions. For example, a separate cluster "homepage" is likely to be constructed in the analysis of clickstreams. Therefore, when the focus of an application is on studying transitions, the initial state effect should be eliminated by assuming fixed probabilities $\beta_{kj} = 1/p$ for $j = 1, \ldots, p$ and $k = 1, \ldots, K$. In this case, the number of parameters is given by $N = Kp^2 - Kp + K - 1$.

A model-based clustering solution is obtained via the Bayes decision rule that assigns observations to the class with the highest posterior probability, i.e., according to the rule $\hat{t}_i = \text{argmax}_k\{\hat{z}_{ik}\}$. The function `click.EM()` from the package **ClickClust** is responsible for running the outlined EM algorithm. The description and use of the function are provided in Section 3.2.

2.2. Variability assessment procedure

If it is needed to construct confidence intervals or conduct tests for model parameters and functions involving them, the variability in parameter estimates has to be assessed. Melnykov (2016a) showed that the observed information matrix corresponding to Equation 1 can be calculated by

$$\boldsymbol{I}(\hat{\boldsymbol{\vartheta}}) = \sum_{i=1}^{n} \boldsymbol{\nabla} \log f(y_{i1}, \boldsymbol{X}_i|\hat{\boldsymbol{\vartheta}}) \boldsymbol{\nabla} \log f(y_{i1}, \boldsymbol{X}_i|\hat{\boldsymbol{\vartheta}})^\top,$$

where $\boldsymbol{\nabla} \log f(y_{i1}, \boldsymbol{X}_i|\boldsymbol{\vartheta})$ is the gradient vector containing the $K - 1$ elements

$$\frac{\partial \log f(y_{i1}, \boldsymbol{X}_i|\boldsymbol{\vartheta})}{\partial \alpha_k} = \frac{z_{ik}}{\alpha_k} - \frac{z_{iK}}{\alpha_K}, \qquad k = 1, \ldots, K-1,$$

the $K(p-1)$ elements

$$\frac{\partial \log f(y_{i1}, \boldsymbol{X}_i|\boldsymbol{\vartheta})}{\partial \beta_{kj}} = z_{ik}\left(\frac{I(y_{i1} = j)}{\beta_{kj}} - \frac{I(y_{i1} = p)}{\beta_{kp}}\right), \qquad k = 1, \ldots, K, \quad j = 1, \ldots, p-1,$$

and the $Kp(p-1)$ elements

$$\frac{\partial \log f(y_{i1}, \boldsymbol{X}_i|\boldsymbol{\vartheta})}{\partial \gamma_{kjj'}} = z_{ik}\left(\frac{x_{ijj'}}{\gamma_{kjj'}} - \frac{x_{ijp}}{\gamma_{kjp}}\right), \quad k = 1, \ldots, K, \quad j = 1, \ldots, p, \quad j' = 1, \ldots, p-1.$$

An estimated variance-covariance matrix for the MLE can be obtained by computing the inverse of the observed information matrix $\boldsymbol{I}(\hat{\boldsymbol{\vartheta}})$. The function `click.var()` calculates the estimated variance-covariance matrix. The description of the function is provided in Section 3.4.

2.3. State selection for merging or splitting

In the previous subsections we observed that the number of model parameters is equal to $Kp^2 - 1$, and thus it is a quadratic function of the number of states p. Therefore, the model might suffer from overparameterization and, as a result, the performance of the clustering procedure might degrade severely. Melnykov (2016a) conducted a simulation study and showed that the mixture model order is often underestimated under such a scenario. To address this issue, backward and forward state selection algorithms that allow combining states together are proposed. These algorithms compare reduced and full models. Under the reduced model, it is assumed that there are similar (or equivalent) states, i.e., states where their corresponding transition probabilities are the same row-wise and column-wise. Such states are combined in so-called equivalence blocks. Statistically, this idea can be formulated as restrictions $\gamma_{kj1} = \gamma_{kj'1}, \ldots, \gamma_{kjp} = \gamma_{kj'p}$ and $\gamma_{k1j} = \gamma_{k1j'}, \ldots, \gamma_{kpj} = \gamma_{kpj'}$ for all j and j' within the same equivalence block. States that do not get combined into equivalence blocks form singleton blocks. This prevents mixing states with blocks and unifies introduced terminology.

The assumption of similar states is often realistic. For example, in the analysis of clickstream sequences collected at an internet-based store, states representing different models of the same brand can often be so closely related that the behavior of customers with regard to these states will follow the same patterns. Since the definition of equivalence blocks relies entirely on the equality of corresponding rows and columns in transition probability matrices, some states without immediate and vivid relationship can also be declared similar due to the closeness of their estimated transition probability distributions. For example, rarely visited states can often be included in the same equivalence block. In general, the usefulness of the proposed tool depends on the specific problem. However, in many applications such an idea can help provide better results.

Forward and backward state selection algorithms can be proposed as the three-stage procedure outlined below.

1. *Initialization.* Propose an initial model by considering all states combined into one equivalence block (forward selection) or by treating each state as individual equivalence block (backward selection). Calculate BIC for the initial model.

2. *Splitting/merging.* Consider all possible models obtained by separating one state into a new block (forward selection) or by merging two equivalence blocks together (backward selection). Choose the best model in terms of BIC. Stop if no improvement is reached, otherwise proceed to Step 3.

3. *Rearrangement.* Consider all possible models with the current number of equivalence blocks by reassigning states among blocks and choose the best model in terms of BIC. Proceed to Step 2.

The outlined algorithm is implemented in **ClickClust** through the functions `click.forward()` and `click.backward()` discussed and illustrated in Section 3.5.

2.4. Prediction of future states visited

Given the set of transition probability matrices $\mathbf{\Gamma}_1, \ldots, \mathbf{\Gamma}_K$ and a probability distribution $\pi_1, \ldots, \pi_K$ associated with the mixture components, the M-step transition probability matrix can be found by

$$\mathbf{\Gamma}^M = \sum_{k=1}^{K} \pi_k \mathbf{\Gamma}_k^M,$$

where $\mathbf{\Gamma}_k^M$ represents the matrix $\mathbf{\Gamma}_k$ raised to the power M. For example, $\mathbf{\Gamma}_k^3 = \mathbf{\Gamma}_k\mathbf{\Gamma}_k\mathbf{\Gamma}_k$. The choice of the distribution $\pi_1, \ldots, \pi_K$ depends on the particular application, but the two immediate candidates are the vector of estimated mixing proportions (i.e., $\hat{\alpha}_1, \ldots, \hat{\alpha}_K$) and the vector of estimated posterior probabilities associated with a specific sequence $\boldsymbol{y}_i$ (i.e., $\hat{z}_{i1}, \ldots, \hat{z}_{iK}$). The function `click.predict()` from the package **ClickClust** provides the functionality outlined in this section. Corresponding examples along with the description of available parameters can be found in Section 3.6.

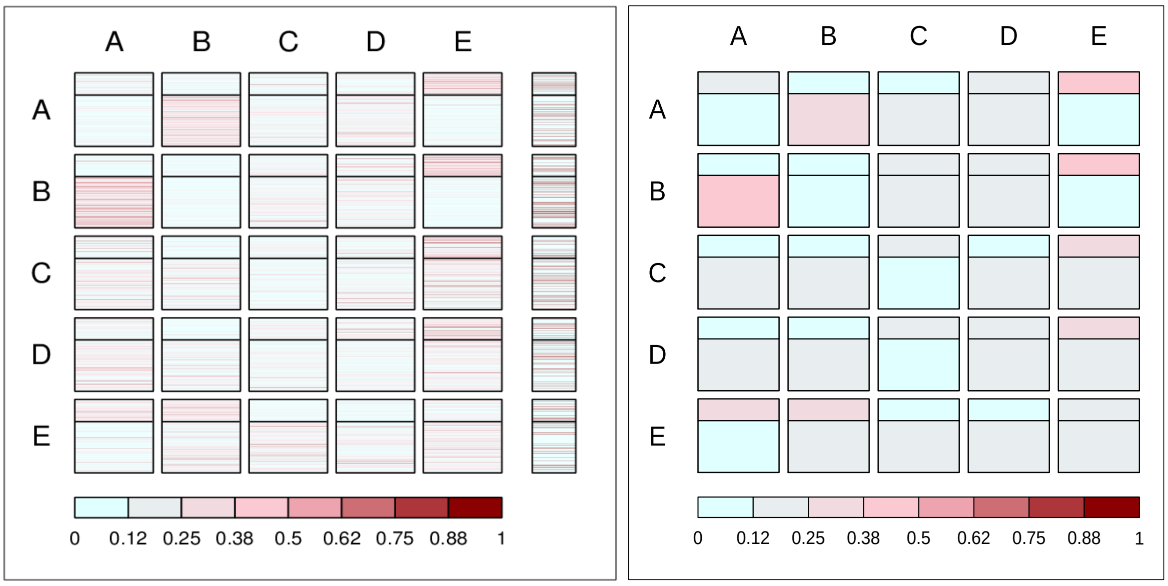

Figure 1: Click-plots representing clustering solutions obtained by the EM algorithm (a) with and (b) without estimation of initial state probabilities.

2.5. Graphical display click-plot

After finding a model-based clustering solution, it is important to present the obtained results in a form that is easy to understand and interpret. Traditional graphical tools are not readily effective in the considered framework. Melnykov (2016a) proposed constructing a graphical display that we call a click-plot. Two examples of such plots are provided in Figure 1. In plot (a) we can see 25 cells associated with a 5×5 transition probability matrix. Each cell is split by a black horizontal separator into two segments representing the first and second cluster. Within each segment, we can observe a number of thin horizontal lines of different colors. Each line corresponds to a particular observation with the color marking the magnitude of the relative frequency of transitions associated with the specific cell. The colors are summarized in the legend provided at the bottom of the plot. According to the chosen color scheme, the darker the red color is, the larger the value of the relative transition frequency is. Thus, the high number of reddish horizontal lines in the top segment of the last cell in the first row implies that there are many observations in the first cluster with relatively large proportions of transitions from state A to state E. On the contrary, the majority of light cyan hues in the bottom segment of the same cell indicates low transitional activity from state A to state E associated with the second cluster. If the initial state probabilities are among the estimated parameters, an additional column of cells can be provided within a click-plot along with the relative transition frequency matrix as shown in Figure 1 (a). The interpretation here is similar, but there are only two colors involved. The dark red horizontal line highlights observations that start in the particular state and the cyan line implies that the sequence begins with some other state. This column of cells can be helpful in the analysis of partitions from the initial state standpoint.

To improve the visibility of the plot, the user has an option to replace all color hues within

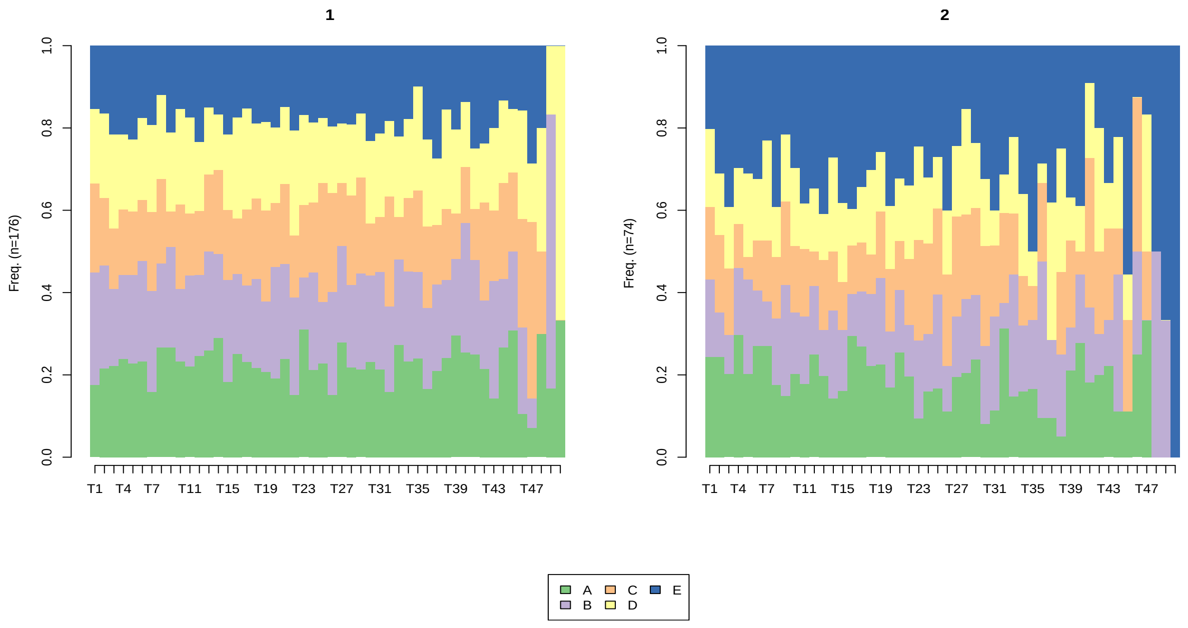

Figure 2: Chronograms of the clustering solution presented in Figure 1 (b).

each cell segment with the corresponding median color. Figure 1 (b) illustrates such an option. As we can see, the distinction in colors is sharper and more vivid compared to that in display (a). For example, we can clearly see that the top cluster of the smaller size has higher relative transition frequencies in cells A–E, B–E, and E–A, while the other cluster is primarily characterized by high proportions of transitions between states A and B.

To emphasize the effectiveness of click-plots in visualizing clustering results based on transition patterns, we compare them with a popular display called chronogram. Chronograms represent state distributions and are often useful in studying the properties of obtained partitions. Chronograms presented in Figure 2 are constructed by means of the R package **TraMineR** (Gabadinho *et al.* 2011) and illustrate the same clustering solution as that presented in Figure 1 (b). Indeed, some useful information can be obtained by the comparison of the plots on the left (cluster 1) and right (cluster 2). We can notice, that state E given in blue color is visited more frequently in observations assigned to the second group. All other states seem to be slightly more frequent in observations associated with the first group. While chronograms provide a great summary of the relative frequencies of state visits over time, they are not capable of reflecting the transitional behavior of sequences within detected groups. Some other interesting display options worth mentioning in the context of this discussion are index and transition pattern plots. The former is similar to chronograms but plots each sequence as a line. The latter is somewhat similar to our click-plots as it also provides a $p \times p$ plot to display transition rates. However, within each cell of such a plot, transition rates are plotted against time. While useful in many applications where data are recorded by days or months, such a tool might be confusing in situations where the notion of time is not clearly defined (for example, in clickstream applications).

Thus, we can conclude that the proposed colorful representation helps assess the quality of the illustrated clustering solution and discover important navigation patterns in the detected partition. The function `click.plot()` available in the package is discussed in Section 3.3.

3. Package description and illustrative examples

This section provides a comprehensive coverage of the **ClickClust** functionality. Every feature discussed below is thoroughly illustrated through examples, also included in the package in the form of demos. The following list summarizes the main capabilities of the package:

- running the EM algorithm for finite mixture models with Markov model components;
- model-based clustering by means of the Bayes decision rule;
- estimating the variance-covariance matrix for model parameter estimates;
- illustrating clustering results by means of click-plots;
- identifying equivalence blocks by backward and forward state selection procedures;
- predicting future locations after M transitions;
- simulating categorical sequences according to specified model parameters.

All functions available in the package **ClickClust** are listed in Table 1. Their use is illustrated via corresponding examples that can be run by the function `demo()`. All demo examples available in the package are provided in Table 2.

Function	Description
`click.read()`	Transforms categorical sequences into the frequency matrix form.
`click.EM()`	Runs the EM algorithm for a mixture of Markov models.
`click.var()`	Estimates the covariance matrix for the obtained MLEs.
`click.plot()`	Constructs a click-plot to display clustering results.
`click.forward()`	Runs a forward state selection procedure.
`click.backward()`	Runs a backward state selection procedure.
`click.predict()`	Predicts future states visited in M transition steps.
`click.sim()`	Simulates categorical sequences given model parameters.

Table 1: Summary of functions implemented in **ClickClust**.

Section	Demo names
Section 3.1	`"ReadData"`
Section 3.2	`"EMalgorithm1"`, `"EMalgorithm2"`
Section 3.3	`"ClickPlot1"`, `"ClickPlot2"`
Section 3.4	`"ConfidenceIntervals"`
Section 3.5	`"ForwardSelection"`, `"BackwardSelection"`
Section 3.6	`"StatePrediction"`
Section 3.7	`"DataSimulation"`
Section 4	`"utility"`
Section 5	`"msnbc323"`

Table 2: Summary of demo examples included in **ClickClust**.

Arguments	Description
`S`	List of categorical sequences.

Returned values	Description
`$X`	Data array of transition frequencies.
`$y`	Vector containing the initial states of the sequences.

Table 3: Summary of arguments and values returned by the function `click.read()`.

3.1. Reading categorical sequences with ClickClust

We start the review of the package with a discussion of how to organize user data in the format used by **ClickClust**. The main function considered in this section is `click.read()`. It reads the list of numeric sequences and returns an array of transition frequencies for all states along with the initial state vector. This format of data is required by the functions `click.EM()`, `click.backward()`, and `click.forward()` considered in future sections. The function call is as follows below:

```
click.read(S)
```

The argument and returned values related to `click.read()` are summarized in Table 3. The parameter `S` represents the list of categorical sequences with states provided in integer form. The function returns the array `$X` of dimensions $p \times p \times n$ containing n transition frequency matrices and the vector `$y` with the initial states for all n sequences.

The dataset `synth`, available in the package, contains a vector of 250 simulated sequences (`$data`) along with the vector of true membership labels (`$id`). The sequences involve 5 states (denoted as A, B, C, D, and E) and have lengths varying between 10 and 50 elements. As a result, some sequences are rather short and do not include all five states. There are 8 sequences that do not contain state C and 7 sequences that do not contain state D. Two of these sequences (#50 and #69) do not have both states. The following example available through the demo `"ReadData"` demonstrates how the dataset `synth$data` can be read and converted into the form standard in **ClickClust**.

```
R> data("synth", package = "ClickClust")
R> synth$data

[1] "E A C C E A D A C E C A E A E E E A C E A A E B C C"

...

[250] "C A B A A B A B A D E B C E C D A B C A D D B A C B B A A"

R> repl.levs <- function(x, ch.lev) {
+    for (j in 1:length(ch.lev)) x <- gsub(ch.levs[j], j, x)
+    return(x)
+  }
R> d <- paste(synth$data, collapse = " ")
```

```
R> d <- strsplit(d, " ")[[1]]
R> ch.levs <- levels(as.factor(d))
R> S <- strsplit(synth$data, " ")
R> S <- sapply(S, repl.levs, ch.levs)
R> S <- sapply(S, as.numeric); S

S
[[1]]
 [1] 5 1 3 3 5 1 4 1 3 5 3 1 5 1 5 5 5 1 3 5 1 1 5 2 3 3

...

[[250]]
 [1] 3 1 2 1 1 2 1 2 1 4 5 2 3 5 3 4 1 2 3 1 4 4 2 1 3 2 2 1 1

R> C <- click.read(S)
R> C

$X
, , 1
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    3    1    3
[2,]    0    0    1    0    0
[3,]    1    0    2    0    3
[4,]    1    0    0    0    0
[5,]    5    1    1    0    2

...

, , 250
     [,1] [,2] [,3] [,4] [,5]
[1,]    2    4    1    2    0
[2,]    5    1    2    0    0
[3,]    2    1    0    1    1
[4,]    1    1    0    1    1
[5,]    0    1    1    0    0

$y
  [1] 5 4 4 5 3 3 2 5 3 4 3 5 3 5 5 1 4 2 3 1 2 1 5 4 2 1 2 3 1 2 1 3 4 1 1 5
 [37] 1 2 2 1 4 4 5 1 5 4 2 5 2 1 3 4 5 5 1 4 4 4 3 1 3 2 5 3 2 2 1 1 2 1 2 5
 [73] 2 3 2 1 2 4 5 3 5 2 2 4 1 2 5 3 5 1 4 5 1 5 3 4 5 3 3 3 4 1 1 2 3 2 2 3
[109] 2 5 4 1 2 3 1 3 2 2 2 3 2 5 4 1 2 2 2 2 3 1 2 2 4 3 5 5 2 2 4 3 2 5 1 5
[145] 4 4 4 4 2 3 2 1 5 5 2 2 2 5 2 3 4 1 1 3 1 4 5 3 4 3 5 3 3 1 1 2 2 3 1 1
[181] 4 3 4 3 1 2 4 4 3 4 1 4 2 3 3 4 2 1 1 4 2 5 5 4 1 2 3 5 1 5 3 2 3 1 4 4
[217] 2 1 2 3 4 3 3 2 1 3 2 2 5 4 2 5 2 2 3 5 4 3 4 3 2 1 4 5 5 4 1 1 1 3
```

In the example given above, all states are identified and saved into the variable `ch.levs` first. The function `repl.levs()` replaces states `ch.levs` given in character format with the states

Arguments	Description
`X`	Data array of transition frequencies.
`y`	Data vector of initial states visited.
`K`	Number of mixture components.
`eps`	Tolerance level.
`r`	Number of random starts for the initialization of the EM algorithm.
`iter`	Number of iterations for each EM run at the initialization stage.
`min.beta`	Lower bound for initial state probabilities.
`min.gamma`	Lower bound for transition probabilities.
`scale.const`	Scaling constant for handling numerical issues.

Returned values	Description
`$z`	Matrix of posterior probabilities.
`$id`	Classification vector.
`$alpha`	Vector of mixing proportions.
`$beta`	Matrix of initial state probabilities.
`$gamma`	Array of transition probability matrices.
`$logl`	Maximized log-likelihood value.
`$BIC`	Bayesian information criterion.

Table 4: Summary of arguments and values returned by the function `click.EM()`.

provided in the form of integer values. The list of numeric vectors with integers representing particular states is denoted as `S`. This format is convenient for managing sequences and expected by the function `click.read()`. In the output produced by the code, some printout is intentionally omitted to save space as indicated where necessary.

3.2. Finite mixture modeling with ClickClust

`click.EM()` is the core function of the package that runs the EM algorithm for finite mixtures with first-order Markov model components. The command has the following syntax:

```
click.EM(X, y = NULL, K, eps = 1e-10, r = 100, iter = 5, min.beta = 1e-3,
  min.gamma = 1e-3, scale.const = 1)
```

The parameters of the function and objects returned by it are listed in Table 4. The description of the array `X` and vector `y` produced by the function `click.read()` was given in the previous section. In many applications, the primary focus is on the analysis of transitions, as we discussed in Section 2.1. In such cases, the user should not provide the initial state vector `y`. Then, probabilities $\boldsymbol{\beta}_k$ will not be estimated and all states will be assumed equally likely as starting points. The parameter `K` specifies the order of the mixture model and takes on positive integers. The default tolerance level `eps` is set at the level of 10^{-10}. The choice of initialization strategy for the EM algorithm is important for finding the correct solution. Biernacki, Celeux, and Govaert (2003) developed a two-stage approach called *emEM* that is incorporated in `click.EM()`. The *emEM* algorithm runs multiple EM algorithms from random initial points for some pre-specified number of iterations. Then, the best intermediate solution obtained this way is used to initialize the final EM algorithm. The parameters `r` and `iter` of the function `click.EM()` specify the number of initial EM algorithm runs and the number of iterations for such runs, respectively. By default, `r` is set equal to 100 and `iter` is equal to 5. Common sense suggests that Markov chains considered in our framework are usually irreducible. This implies that all transition probabilities must be greater than zero and the user should be able to set the smallest probability value. The arguments `min.beta` and `min.gamma` provide lower bounds for initial state probabilities and transition probabilities, respectively. The final parameter `scale.const` is devoted to handling numerical issues that might occur in the course of the EM algorithm due to raising low probabilities to high powers. The default value is 1, which represents the situation with no scaling applied. In the majority of cases, the specific value of this parameter is not important. However, increasing `scale.const` can help to overcome some numerical issues.

The first object returned by the function `click.EM()` is the $n \times K$ matrix of posterior probabilities `$z`. An estimated classification vector obtained by the Bayes decision rule is provided in the vector `$id`. Objects `$alpha`, `$beta`, and `$gamma` represent estimates of mixing proportions, initial state probabilities, and transitional probabilities, respectively. `$alpha` is a vector of length K, `$beta` is a matrix consisting of K initial state probability vectors of length p, and `$gamma` is an array containing K $p \times p$ transition probability matrices. Finally, the parameters `$logl` and `$BIC` provide the maximized log-likelihood and corresponding BIC values.

We illustrate the utility of the function `click.EM()` on two examples. In both cases, we use the dataset `synth$data`. The corresponding demos are called `"EMalgorithm1"` and `"EMalgorithm2"`.

```
R> set.seed(123)
R> N2 <- click.EM(X = C$X, K = 2)
R> N2

K = 2, p = 5, logl = -11688.35, BIC = 23603.09

Cluster sizes:
  1   2
176  74

id:
  [1] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [37] 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2
 [73] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[109] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[145] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[181] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[217] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1

alpha:
[1] 0.6949289 0.3050711

Use $ to access:
        -transition probability matrices (gamma)
        -posterior probabilities (z)
```

As we can see from the vector `id` in the output provided above, the first group of data is denoted as cluster 2 and followed by observations assigned to cluster 1. This is a well-documented phenomenon known as label switching that does not cause serious practical issues in the considered framework.

In the second example, which is run on the same simulated dataset, it is assumed that initial state probabilities have to be estimated along with the other parameters. This is achieved by specifying the vector of initial states visited, `C$y`.

```
R> set.seed(123)
R> M2 <- click.EM(X = C$X, y = C$y, K = 2)
R> M2

K = 2, p = 5, logl = -11684.21, BIC = 23638.98

Cluster sizes:
  1   2
174  76

id:
  [1] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [37] 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2
 [73] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[109] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[145] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[181] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 2
[217] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1

alpha:
[1] 0.6958358 0.3041642

beta:
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.1715453 0.2763954 0.2148031 0.1809289 0.1563273
[2,] 0.2519450 0.1830399 0.1792858 0.1910258 0.1947035

Use $ to access:
        -transition probability matrices (gamma)
        -posterior probabilities (z)
```

As we can see from the output for both demos, the maximized log-likelihood value is equal to $-11,688.35$ in the first case versus the value of $-11,684.21$ in the second one. Expectedly, the model with initial state probabilities included yields a higher log-likelihood value as it fits the data slightly better. The difference is rather marginal and the model without initial state probabilities is preferred over the model with them based on the BIC value of 23,603.09 versus BIC equal to 23,638.98. The preferred model also demonstrates slightly better performance in terms of the number of correctly classified observations. It yields 8 misclassifications (rate 0.032) versus 10 misclassifications (rate 0.040) obtained in the other case. The relatively low misclassification rates can be explained by considerable differences in the true transition probability matrices associated with the two components. These can be easily observed by means of click-plots constructed and thoroughly discussed for these two examples in Section 3.3.

3.3. Constructing a click-plot with ClickClust

After a clustering result is obtained, a click-plot can be constructed by means of the function `click.plot()`. It produces the plot either in a graphical window or saves it into a PDF file. The function has the following syntax:

```
click.plot(X, y = NULL, file = NULL, id, states = NULL, marg = 1,
  font.cex = 2, font.col = "black", cell.cex = 1, cell.lwd = 1.3,
  cell.col = "black", sep.lwd = 1.3, sep.col = "black", obs.lwd = NULL,
  colors = c("lightcyan", "pink", "darkred"), col.levels = 8, legend = TRUE,
  leg.cex = 1.3, top.srt = 0, frame = TRUE)
```

All available function arguments with brief descriptions are listed in Table 5. The first two parameters of the function are `X` and `y`, representing the array of transition frequencies and the vector of initial states visited, respectively. While `X` is a required parameter, `y` is not. If `y` is not provided (default setting), initial state probabilities will not be shown in the plot. The argument `file` specifies the name of the PDF file for output. If the name is not provided, the function constructs the plot, but does not save it into a file. The vector `id` contains the proposed partitioning vector. The rest of the parameters provide graphical capabilities for the constructed click-plot. `states` contains the names of states. If the names are not included, all states will be denoted by integers starting from 1. `marg` is responsible for setting the size of the left and top plot margins. `font.cex` and `font.col` specify the magnification and color of row and column labels. The latter is also responsible for the color of legend labels, if applicable. Parameters `cell.cex`, `cell.lwd`, and `cell.col` specify the size of the intercell interval, the width of the cell frame, and frame color, respectively. Similarly, the parameters `sep.lwd` and `sep.col` are responsible for the width and color of horizontal lines separating cell segments. `obs.lwd` specifies the width of lines representing individual observations. If this width is not provided (default state), the color of each cell segment is chosen based on the corresponding median. The argument `colors` specifies base colors for the color interpolation with `col.levels` number of hues produced. The default number of the color levels is 8. The next two arguments are responsible for presenting the color hues through a legend. `legend` is an indicator specifying whether the legend should be presented or not while `leg.cex` provides the magnification of the font for legend labels. `top.srt` allows to rotate the state names given in the top of the plot. This feature is helpful when state names are lengthy and cannot be provided horizontally. Finally, the parameter `frame` controls whether a frame around the plot is constructed.

In the following we construct click-plots for the results obtained in the two illustrations from Section 3.2.

```
R> click.plot(X = C$X, y = C$y, id = M2$id, states = ch.levs, obs.lwd = 0.3)
R> click.plot(X = C$X, id = N2$id, states = ch.levs)
```

The examples can be run by commands `demo("ClickPlot1", package = "ClickClust")` and `demo("ClickPlot2", package = "ClickClust")`, respectively. Figure 1 displays the

Arguments	Description
`X`	Data array of transition frequencies.
`y`	Data vector of initial states visited.
`file`	Name of the output PDF file.
`id`	Classification vector.
`states`	Names of states.
`marg`	Left and top plot margin.
`font.cex`	Font scaling.
`font.col`	Font color.
`cell.cex`	Cell scaling.
`cell.lwd`	Width of the cell frame.
`cell.col`	Color of the cell frame.
`sep.lwd`	Width of the group separator line.
`sep.col`	Color of the group separator line.
`obs.lwd`	Width of observation lines.
`colors`	Color scheme.
`col.levels`	Number of colors.
`legend`	Inclusion of legend.
`leg.cex`	Legend font scaling.
`top.srt`	Rotation of state labels in the top.
`frame`	Plot frame.

Table 5: Summary of arguments of the function `click.plot()`.

click-plots constructed in these two examples. The detailed discussion of the plots was already provided in Section 2.5. Both displays effectively illustrate obtained clustering solutions, suggesting that the distinction between the two clusters is mostly due to the difference in transitions A–B, A–E, B–A, B–E, and E–A. For the cells associated with these transitions, the color discrepancy between the corresponding segments reaches its highest levels. The described solution characteristics agree well with the original parameter values of this simulated dataset that can be found in Section 3.7. From plot (a), we can also conclude that initial state probabilities do not contribute to separating the two clusters.

3.4. Variability assessment with ClickClust

The function `click.var()` employs the algorithm for assessing the variability of obtained estimates as outlined in Section 2.2. The function has the following syntax:

```
click.var(X, y = NULL, alpha, beta = NULL, gamma, z)
```

All available parameters are listed in Table 6. The arguments have the same meaning as before. `z` represents the $n \times K$ matrix of posterior probabilities. The only object returned by the function is the estimated variance-covariance matrix. The parameters associated with this matrix are arranged in the following order: mixing proportions ($K-1$ elements), initial state probabilities (if specified, $K(p-1)$ elements) listed by rows, and transition probabilities ($Kp(p-1)$ elements) listed by rows for all transition probability matrices.

The following example illustrates how 95% confidence intervals can be calculated for the

Arguments	Description
`X`	Data array of transition frequencies.
`y`	Data vector of initial states visited.
`alpha`	Vector of mixing proportions.
`beta`	Matrix of initial state probabilities.
`gamma`	Array of transition probability matrices.
`z`	Matrix of posterior probabilities.

Table 6: Summary of arguments for the function `click.var()`.

parameter estimates obtained in the course of the first example considered in Section 3.2. The corresponding demo can be run by the command `demo("ConfidenceIntervals", package = "ClickClust")`.

```
R> V <- click.var(X = C$X, alpha = N2$alpha, gamma = N2$gamma, z = N2$z)
R> st.err <- sqrt(diag(V))
R> Estimates <- c(N2$alpha[-2], as.vector(apply(N2$gamma[, -5, ], 3, t)))
R> Lower <- Estimates - qnorm(0.975) * st.err
R> Upper <- Estimates + qnorm(0.975) * st.err
R> cbind(Estimates, Lower, Upper)

[1,] 0.69492889 0.63146870 0.7583891
[2,] 0.12356408 0.10330503 0.1438231
[3,] 0.35616335 0.32698788 0.3853388
[4,] 0.19211304 0.16586742 0.2183587

...

[38,] 0.29587636 0.25465414 0.3370986
[39,] 0.30146575 0.26325962 0.3396719
[40,] 0.09661424 0.07061489 0.1226136
[41,] 0.10930597 0.08249654 0.1361154
```

The first column of the produced output contains obtained estimates while the second and third columns represent the lower and upper bounds of confidence intervals, respectively. Although there are many parameters, they are estimated quite accurately.

3.5. State merging and splitting with ClickClust

As we discussed in Section 2.3, the performance of a clustering procedure can be severely affected by model overparameterization. Therefore, especially when the number of states is high, it is important to identify states with similar estimated transition probability distributions. This might help reduce the number of parameters and improve the performance of the model-based clustering procedure. The package **ClickClust** includes two functions responsible for the selection of equivalent states. The function `click.forward()` conducts the forward state selection and the function `click.backward()` is responsible for the backward search.

Arguments	Description
`X`	Data array of transition frequencies.
`K`	Number of mixture components.
`eps`	Tolerance level.
`r`	Number of random starts for the initialization of the EM algorithm.
`iter`	Number of iterations for each EM run at the initialization stage.
`bic`	Flag indicating which information criterion is used, BIC or AIC.
`min.gamma`	Lower bound for transition probabilities.
`scale.const`	Scaling constant for handling numerical issues.
`silent`	Output control.

Returned values	Description
`$z`	Matrix of posterior probabilities.
`$id`	Classification vector.
`$alpha`	Vector of mixing proportions.
`$gamma`	Array of transition probability matrices.
`$states`	Vector of states representing equivalent blocks.
`$logl`	Maximized log-likelihood value.
`$BIC/$AIC`	Bayesian/Akaike information criterion.

Table 7: Summary of arguments and values returned by functions `click.backward()` and `click.forward()`.

The arguments of both functions as well as their returned values are identical and provided in Table 7. The function calls have the following syntax:

```
click.forward(X, K, eps = 1e-10, r = 100, iter = 5, bic = TRUE,
  min.gamma = 1e-3, scale.const = 1.0, silent = FALSE)
click.backward(X, K, eps = 1e-10, r = 100, iter = 5, bic = TRUE,
  min.gamma = 1e-3, scale.const = 1.0, silent = FALSE)
```

The array `X` contains data in the transition frequency form. The parameter `K` specifies the number of mixture components. Arguments `eps`, `r`, `iter`, `min.gamma`, and `scale.const` have the same meaning as in the description of the function `click.EM()`. The argument `silent` allows to disable the output produced by both functions. The focus of both model selection procedures is on the reduction of the number of estimated transition probabilities. Therefore, initial state probabilities are assumed fixed and equal and are not estimated in the course of the state selection procedures.

The values returned by `click.backward()` and `click.forward()` are mainly the same as those returned by `click.EM()`. The only new object `$states` contains the vector of length p that represents the obtained blocks of equivalent states. States associated with the same number are included in a common equivalence block. As a result, the array of transition probability matrices `gamma` has dimensionality $d \times d \times K$, where d is the number of equivalence blocks such that $d \leq p$.

In the first example considered within this section, we illustrate the function `click.forward()`. The data simulated in the example of Section 3.7 are used here, which is achieved by running the corresponding demo.

```
R> set.seed(1234)
R> F2 <- click.forward(X = C$X, K = 2)


STAGE A: Separating states...
        States: 2 1 1 1 1         logL = -12118.09
        States: 1 2 1 1 1         logL = -12094.76
        States: 1 1 2 1 1         logL = -12143.2
        States: 1 1 1 2 1         logL = -12158.2
        States: 1 1 1 1 2         logL = -12096.73
        Current set of states: 1 2 1 1 1 logL = -12094.76 BIC = 24217.12

STAGE B: Rearranging states...
        States: 2 2 1 1 1         logL = -12141.66
        States: 1 2 2 1 1         logL = -12118.29
        States: 1 2 1 2 1         logL = -12129.22
        States: 1 2 1 1 2         logL = -12129.9
        Current set of states: 1 2 1 1 1 logL = -12094.76 BIC = 24217.12

        Stopped rearrangement...

STAGE A: Separating states...
        States: 3 2 1 1 1         logL = -11956.31
        States: 1 2 3 1 1         logL = -12057.77
        States: 1 2 1 3 1         logL = -12063.25
        States: 1 2 1 1 3         logL = -11928.9
        Current set of states: 1 2 1 1 3 logL = -11928.9 BIC = 23929.59

...

STAGE B: Rearranging states...
        States: 2 4 2 1 3         logL = -11878.51
        States: 2 4 3 1 3         logL = -11877.05
        States: 2 4 4 1 3         logL = -11868.52
        States: 2 4 1 2 3         logL = -11858.96
        States: 2 4 1 3 3         logL = -11894.13
        States: 2 4 1 4 3         logL = -11856.18
        Current set of states: 2 4 1 1 3 logL = -11701.26 BIC = 23540.56

        Stopped rearrangement...

STAGE A: Separating states...
        States: 2 4 5 1 3         logL = -11688.35
        States: 2 4 1 5 3         logL = -11688.35
        No improvement reached...


R> F2
```

```
K = 2, p = 5, d = 4, logl = -11701.26, BIC = 23540.56

States:[1] 2 4 1 1 3

Cluster sizes:
  1   2
177  73

id:
  [1] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [37] 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2
 [73] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[109] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[145] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[181] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
[217] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1

alpha:
[1] 0.6977365 0.3022635

Use $ to access:
        -transition probability matrices (gamma)
        -posterior probabilities (z)
```

As we can see from the output provided above, the search for the best model is conducted by rearranging and separating states. The best model is chosen based on the smallest BIC value. From the transition probabilities given in Section 3.7, we can conclude that states 3 and 4 are indeed equivalent and thus they are identified correctly. Overall, there are four equivalence blocks detected: {1}, {2}, {3, 4}, and {5}.

In the second example, the function `click.backward()` is illustrated on the same dataset.

```
R> set.seed(1234)
R> B2 <- click.backward(X = C$X, K = 2)

STAGE A: Merging equivalence blocks...
        States: 1 1 2 3 4         logL = -11917
        States: 1 2 1 3 4         logL = -11878.51
        States: 1 2 3 1 4         logL = -11858.96
        States: 1 2 3 4 1         logL = -11960.55
        States: 1 2 2 3 4         logL = -11868.52
        States: 1 2 3 2 4         logL = -11856.18
        States: 1 2 3 4 2         logL = -12059.25
        States: 1 2 3 3 4         logL = -11701.26
        States: 1 2 3 4 3         logL = -11877.05
        States: 1 2 3 4 4         logL = -11894.13
        Current set of states: 1 2 3 3 4 logL = -11701.26 BIC = 23540.56
```

```
STAGE B: Rearranging states...
        States: 1 2 1 3 4        logL = -11878.51
        States: 1 2 2 3 4        logL = -11868.52
        States: 1 2 4 3 4        logL = -11877.05
        States: 1 2 3 1 4        logL = -11858.96
        States: 1 2 3 2 4        logL = -11856.18
        States: 1 2 3 4 4        logL = -11894.13
        Current set of states: 1 2 3 3 4 logL = -11701.26 BIC = 23540.56

        Stopped rearrangement...

STAGE A: Merging equivalence blocks...
        States: 1 1 2 2 3        logL = -11928.52
        States: 1 2 1 1 3        logL = -11928.9
        States: 1 2 3 3 1        logL = -11974.71
        States: 1 2 2 2 3        logL = -11917.48
        States: 1 2 3 3 2        logL = -12070.8
        States: 1 2 3 3 3        logL = -11956.31
        Current set of states: 1 2 3 3 4 logL = -11701.26 BIC = 23540.56
```

```
R> B2
```

```
K = 2, p = 5, d = 4, logl = -11701.26, BIC = 23540.56

States:[1] 1 2 3 3 4

Cluster sizes:
  1   2
177  73

id:
  [1] 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 [37] 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 2 2 2
 [73] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[109] 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[145] 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[181] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2
[217] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1

alpha:
[1] 0.6977571 0.3022429

Use $ to access:
        -transition probability matrices (gamma)
        -posterior probabilities (z)
```

According to the output from `click.backward()`, the results are the same as those obtained by `click.forward()`. The BIC value produced by both state selection procedures is

Arguments	Description
`M`	Number of transition steps.
`gamma`	Array of transition probability matrices.
`pr`	Probability vector associated with mixture components.

Table 8: Summary of arguments for the function `click.predict()`.

23,540.56, which is considerably lower than corresponding BIC values obtained in Section 3.2 by the naive clustering with no state search conducted (23,603.09 and 23,638.98). This proves that backward and forward state selection approaches are effective tools for simplifying models. The misclassification rate associated with the detected clustering solution is 0.036 (9 misclassifications). This result is similar to that of the naive clustering with and without initial state probabilities estimated (10 and 8 misclassifications, respectively).

3.6. Predicting future states with ClickClust

The functionality for predicting future states visited is another important feature that is incorporated in the package **ClickClust**. The function `click.predict()` calculates the transition probability matrix associated with the M-step transition. The description of the process is provided in Section 2.4. The arguments of the function are listed in Table 8. The function call has the following syntax:

```
click.predict(M = 1, gamma, pr = NULL)
```

The parameter `M` provides a positive integer indicating the number of transitions needed. Thus, the default value `M = 1` implies that the user is interested in a one-step transition. The parameter `gamma` has the same meaning as before and denotes the set of transition probabilities associated with K components. Finally, the parameter `pr` is responsible for providing a probability distribution corresponding to the components. Depending on a particular problem, `pr` can be chosen to be a vector of mixing proportions or the vector of posterior probabilities associated with a specific sequence of states. If the option `pr` is not specified, the function `click.predict()` assumes equally likely components.

In the example provided below, the results obtained by the EM algorithm in the first example from Section 3.2 are employed to illustrate the use of the function `click.predict()`. The corresponding demo can be run by the command `demo("StatePrediction", package = "ClickClust")`.

```
R> T <- click.predict(M = 3, gamma = N2$gamma, pr = N2$z[1, ])
R> colnames(T) <- ch.levs
R> T[S[[1]][length(S[[1]])], ]

        A         B         C         D         E
0.1985400 0.1595880 0.1601376 0.1636892 0.3180452
```

The function `click.predict()` is used to find the probability distribution associated with the three-step transition from the last state in the first sequence, i.e., `S[[1]]`. The posterior vector `N2$z[1, ]` associated with this sequence specifies the values of the parameter `pr`. From

Arguments	Description
`n`	Number of sequences to simulate.
`int`	Vector of the lower and upper bound for the length of sequences.
`alpha`	Vector of mixing proportions.
`beta`	Matrix of initial state probabilities.
`gamma`	Array of transition probability matrices.

Returned values	Description
`$S`	List of simulated sequences.
`$id`	Group membership of generated sequences.

Table 9: Summary of arguments and values returned by the function `click.sim()`.

the obtained results, we can see that the most likely state to visit in three steps is E; the probability associated with this prediction is slightly higher than 0.318.

3.7. Simulating categorical sequences with ClickClust

One more important feature of **ClickClust** is the ability to simulate data according to the model provided in Equation 3. The function `click.sim()` allows simulating categorical sequences according to pre-specified model parameters such as mixing proportions, initial state probabilities, and transition probabilities. The function has the following syntax:

```
click.sim(n, int = c(5, 100), alpha, beta = NULL, gamma)
```

All available arguments and values returned by the function `click.sim()` are provided in Table 9. The user specifies the number of sequences to be simulated through the parameter `n`. The interval `int` provides the lower and upper bound for the length of simulated sequences. The length of each sequence is a random realization from the discrete uniform distribution defined over all integers in the interval `int`. Parameters `alpha`, `beta`, and `gamma` have the same meaning as before. If `beta` is not provided, initial state probabilities are assumed to be all equal to $1/p$. The function `click.sim()` returns the list of simulated sequences `$S` and the corresponding vector of membership labels `$id`. The sequences `$S` are provided in the form required by the function `click.read()`.

The following example illustrates the use of the function. 250 sequences with lengths between 10 and 50 are simulated from a mixture with two 5-state Markov model components. Mixing proportions and transition probability matrices are specified in the objects `mix.prop` and `TP`, respectively. Initial state probabilities are assumed equal. The code provided below simulates the sequences and saves them along with the classification vector into the object `A`. The simulated data can be also accessed by running the command `data("synth", package = "ClickClust")`. This is the dataset used in previous sections for illustrating the utility of functions. In the output produced by the code, some printout is intentionally omitted to save space as indicated where necessary. The considered example is also available through the demo called `"DataSimulation"`.

```
R> set.seed(123)
R> n.seq <- 250
```

```
R> p <- 5
R> K <- 2
R> mix.prop <- c(0.3, 0.7)
R> TP <- array(rep(NA, p * p * K), c(p, p, K))
R> TP[, , 1] <- matrix(c(0.20, 0.10, 0.15, 0.15, 0.40,
+    0.10, 0.10, 0.20, 0.20, 0.40, 0.15, 0.10, 0.20, 0.20, 0.35,
+    0.15, 0.10, 0.20, 0.20, 0.35, 0.30, 0.30, 0.10, 0.10, 0.20),
+    byrow = TRUE, ncol = p)
R> TP[, , 2] <- matrix(c(0.15, 0.35, 0.20, 0.20, 0.10,
+    0.40, 0.10, 0.20, 0.20, 0.10, 0.25, 0.20, 0.15, 0.15, 0.25,
+    0.25, 0.20, 0.15, 0.15, 0.25, 0.10, 0.20, 0.20, 0.20, 0.30),
+    byrow = TRUE, ncol = p)
R> A <- click.sim(n = n.seq, int = c(10, 50), alpha = mix.prop, gamma = TP)
R> A

$S
$S[[1]]
 [1] 5 1 3 3 5 1 4 1 3 5 3 1 5 1 5 5 5 1 3 5 1 1 5 2 3 3

...

$S[[250]]
 [1] 3 1 2 1 1 2 1 2 1 4 5 2 3 5 3 4 1 2 3 1 4 4 2 1 3 2 2 1 1
$id
  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [37] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [73] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[109] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[145] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[181] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
[217] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

R> C <- click.read(A$S)
R> C

$X
, , 1
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    0    3    1    3
[2,]    0    0    1    0    0
[3,]    1    0    2    0    3
[4,]    1    0    0    0    0
[5,]    5    1    1    0    2

...

, , 250
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    2    4    1    2    0
[2,]    5    1    2    0    0
[3,]    2    1    0    1    1
[4,]    1    1    0    1    1
[5,]    0    1    1    0    0

$y
  [1] 5 4 4 5 3 3 2 5 3 4 3 5 3 5 5 1 4 2 3 1 2 1 5 4 2 1 2 3 1 2 1 3 4 1 1 5
 [37] 1 2 2 1 4 4 5 1 5 4 2 5 2 1 3 4 5 5 1 4 4 4 3 1 3 2 5 3 2 2 1 1 2 1 2 5
 [73] 2 3 2 1 2 4 5 3 5 2 2 4 1 2 5 3 5 1 4 5 1 5 3 4 5 3 3 3 4 1 1 2 3 2 2 3
[109] 2 5 4 1 2 3 1 3 2 2 2 3 2 5 4 1 2 2 2 2 3 1 2 2 4 3 5 5 2 2 4 3 2 5 1 5
[145] 4 4 4 4 2 3 2 1 5 5 2 2 2 5 2 3 4 1 1 3 1 4 5 3 4 3 5 3 3 1 1 2 2 3 1 1
[181] 4 3 4 3 1 2 4 4 3 4 1 4 2 3 3 4 2 1 1 4 2 5 5 4 1 2 3 5 1 5 3 2 3 1 4 4
[217] 2 1 2 3 4 3 3 2 1 3 2 2 5 4 2 5 2 2 3 5 4 3 4 3 2 1 4 5 5 4 1 1 1 3
```

4. Illustrative use of the package

In this section, we consider an illustrative example that highlights the usefulness of the developed methodology on a rather complicated simulated dataset. Given parameters of the tri-component 15-state Markov mixture model as provided in Table 10, 100 categorical sequences were simulated with lengths varying between 10 and 100. Indeed, a sequence consisting of 10 elements is considered very short in this setting, while one with 100 elements is of considerable length. Mixing proportions are chosen to be 0.25, 0.35, and 0.40. All states are equally likely as starting points for all components. The code corresponding to this example is provided in the supplementary file `v74i09.R` and can also be run by the command `demo("utility", package = "ClickClust")`.

Table 11 provides the performance summary (using **MixSim**, Melnykov, Chen, and Maitra 2012) for four model-based clustering methods: the EM algorithm with and without initial state probabilities estimated as well as forward and backward state selection procedures. Each method is evaluated over mixture models with the number of components ranging from 1 to 5. The performance of methods is compared by means of BIC. As we can see, both EM algorithms severely underestimate the order of the mixture, choosing the model with a single component. On the contrary, both state selection procedures correctly estimate the order of the mixture model, producing the lowest BIC value of 29,397.38. The goodness of the obtained partition is evaluated by means of the adjusted Rand index (ARI; Hubert and Arabie 1985), which compares two partitions and yields the value 1 when they match completely. The weaker the agreement between the partitions, the lower are the values produced by ARI. Since the true membership of all observations is known in our illustrative example, ARI can be conveniently employed to measure the agreement between the original and estimated groupings. Table 11 provides ARI results obtained for all methods with different K values. The highest value of 0.898 is reached by the forward and backward selection algorithms when $K = 3$. The other two approaches yield lower ARI values for $K = 3$, implying that even when the correct number of components is known and does not have to be estimated, the obtained partitions are still not as good as the one obtained when the state selection procedures are applied. This

				$\alpha_1 = 0.25$		$\alpha_2 = 0.35$			$\alpha_3 = 0.40$						
	1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
$\boldsymbol{\beta}_1 \times 15$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\boldsymbol{\beta}_2 \times 15$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\boldsymbol{\beta}_3 \times 15$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$\boldsymbol{\Gamma}_1 \times 100$	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
1	30	1	3	3	10	15	10	5	5	5	2	2	5	2	2
2	1	30	5	5	5	10	10	7	7	7	3	3	1	3	3
3	5	5	20	20	5	5	5	5	5	5	2	2	12	2	2
4	5	5	20	20	5	5	5	5	5	5	2	2	12	2	2
5	2	5	10	10	40	5	5	2	2	2	4	4	1	4	4
6	2	2	1	1	2	35	1	5	5	5	10	10	1	10	10
7	3	5	10	10	5	5	15	2	2	2	10	10	1	10	10
8	5	10	5	5	10	2	5	15	15	15	3	3	1	3	3
9	5	10	5	5	10	2	5	15	15	15	3	3	1	3	3
10	5	10	5	5	10	2	5	15	15	15	3	3	1	3	3
11	10	5	5	5	10	3	5	5	5	5	10	10	2	10	10
12	10	5	5	5	10	3	5	5	5	5	10	10	2	10	10
13	3	1	15	15	1	10	5	10	10	10	3	3	8	3	3
14	10	5	5	5	10	3	5	5	5	5	10	10	2	10	10
15	10	5	5	5	10	3	5	5	5	5	10	10	2	10	10
$\boldsymbol{\Gamma}_2 \times 100$	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
1	1	1	3	3	10	5	10	5	5	5	12	12	4	12	12
2	1	5	5	5	5	10	10	16	16	16	2	2	3	2	2
3	5	5	5	5	5	5	5	15	15	15	3	3	8	3	3
4	5	5	5	5	5	5	5	15	15	15	3	3	8	3	3
5	10	5	10	10	10	5	5	4	4	4	5	5	13	5	5
6	2	2	1	1	2	5	2	5	5	5	10	10	30	10	10
7	13	2	10	10	5	5	10	3	3	3	5	5	16	5	5
8	11	1	5	5	3	2	3	5	5	5	10	10	15	10	10
9	11	1	5	5	3	2	3	5	5	5	10	10	15	10	10
10	11	1	5	5	3	2	3	5	5	5	10	10	15	10	10
11	10	5	5	5	10	15	5	5	5	5	5	5	10	5	5
12	10	5	5	5	10	15	5	5	5	5	5	5	10	5	5
13	6	1	15	15	1	10	5	10	10	10	3	3	5	3	3
14	10	5	5	5	10	15	5	5	5	5	5	5	10	5	5
15	10	5	5	5	10	15	5	5	5	5	5	5	10	5	5
$\boldsymbol{\Gamma}_3 \times 100$	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
1	10	10	5	5	10	10	10	5	5	5	5	5	5	5	5
2	5	5	5	5	5	5	10	10	10	10	5	5	10	5	5
3	5	5	5	5	5	5	10	10	10	10	5	5	10	5	5
4	5	5	5	5	5	5	10	10	10	10	5	5	10	5	5
5	5	5	5	5	4	5	5	4	4	4	1	1	50	1	1
6	2	4	5	5	12	5	2	5	5	5	10	10	10	10	10
7	15	10	10	10	5	5	5	5	5	5	5	5	5	5	5
8	5	1	7	7	2	3	10	5	5	5	10	10	10	10	10
9	5	1	7	7	2	3	10	5	5	5	10	10	10	10	10
10	5	1	7	7	2	3	10	5	5	5	10	10	10	10	10
11	4	15	3	3	5	10	5	5	5	5	7	7	12	7	7
12	4	15	3	3	5	10	5	5	5	5	7	7	12	7	7
13	2	3	5	5	60	1	4	3	3	3	2	2	3	2	2
14	4	15	3	3	5	10	5	5	5	5	7	7	12	7	7
15	4	15	3	3	5	10	5	5	5	5	7	7	12	7	7

Table 10: Parameters of the mixture with three 15-state Markov model components used in the illustrative example from Section 4. $\boldsymbol{\beta}_k$ and $\boldsymbol{\Gamma}_k$ for $k = 1, 2, 3$ represent the vector of initial state probabilities and transition probability matrices, respectively.

Method		$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$
EM algorithm	BIC	**30504.68**	30622.71	30997.65	31791.41	32648.21
(with $\boldsymbol{\beta}$)	ARI	0.000	0.575	0.866	0.769	0.518
EM algorithm	BIC	**30464.14**	30531.22	30855.81	31551.64	32288.55
(without $\boldsymbol{\beta}$)	ARI	0.000	0.551	0.752	0.678	0.570
Forward selection	BIC	29983.95	29566.18	**29397.38**	29997.52	30057.80
	ARI	0.000	0.572	0.898	0.788	0.658
Backward selection	BIC	29983.95	29566.18	**29397.38**	29571.22	29708.93
	ARI	0.000	0.572	0.898	0.810	0.701

Table 11: Results of model-based clustering with (forward and backward selection) and without (EM algorithm) state selection procedures. Corresponding BIC and ARI values are provided for each number of components $K = 1, \ldots, 5$. BIC values given in bold font represent the best model detected by a specific method.

example highlights the importance of the biclustering approach in which states are organized into equivalence blocks along with the estimation of the mixture order.

Now, we focus on the optimal solution obtained by the state selection approaches. There are 9 equivalence blocks identified by both procedures: {1}, {2}, {3, 4}, {5}, {6}, {7}, {8, 9, 10}, {11, 12, 14, 15}, and {13}. This result is fully correct as we can see from Table 10, where the row and column probabilities match for all states within each non-singleton block. It is also important to mention that the procedures do not discover spurious blocks. The performance of the algorithm is quite impressive as there are just 100 observations and many of them have rather short lengths. In particular, there are just 3 misclassifications and the lengths of two of these observations are 11 and 12. Clearly, the correct classification of these short data sequences is challenging. On the other hand, there are multiple observations with similar lengths that are classified correctly.

Figure 3 provides an illustration to the 3-cluster partition obtained by the state selection algorithms. It is worth mentioning that both selection procedures take a while to obtain the results we discuss in this section. Therefore, as suggested by one of the reviewers, the reader can get the immediate access to the corresponding click-plot by running the following commands:

```
R> data("utilityB3", package = "ClickClust")
R> dev.new(width = 11, height = 11)
R> click.plot(X = C$X, id = B3$id, colors = c("lightyellow", "red",
+    "darkred"), col.levels = 10)
```

From the click-plot, we can draw several conclusions regarding the typical navigation patterns of observations in each cluster. The first group, represented by the top segment of each cell, is mainly responsible for transitions within the same state. This statement is supported by red color hues associated with cells such as 1–1, 2–2, 3–3, 4–4, and 5–5. Another trend common for this cluster is transitions between states 3 and 4 reflected by cells 3–4 and 4–3. These results agree well with the original parameter values provided in matrix $\boldsymbol{\Gamma}_1$ from Table 10: Corresponding transition probabilities are all relatively high. Two other transitions worth mentioning with regard to the first group detected are 4–13 and 7–14. The other two clusters do not support the described navigation behavior. The second group, illustrated by the

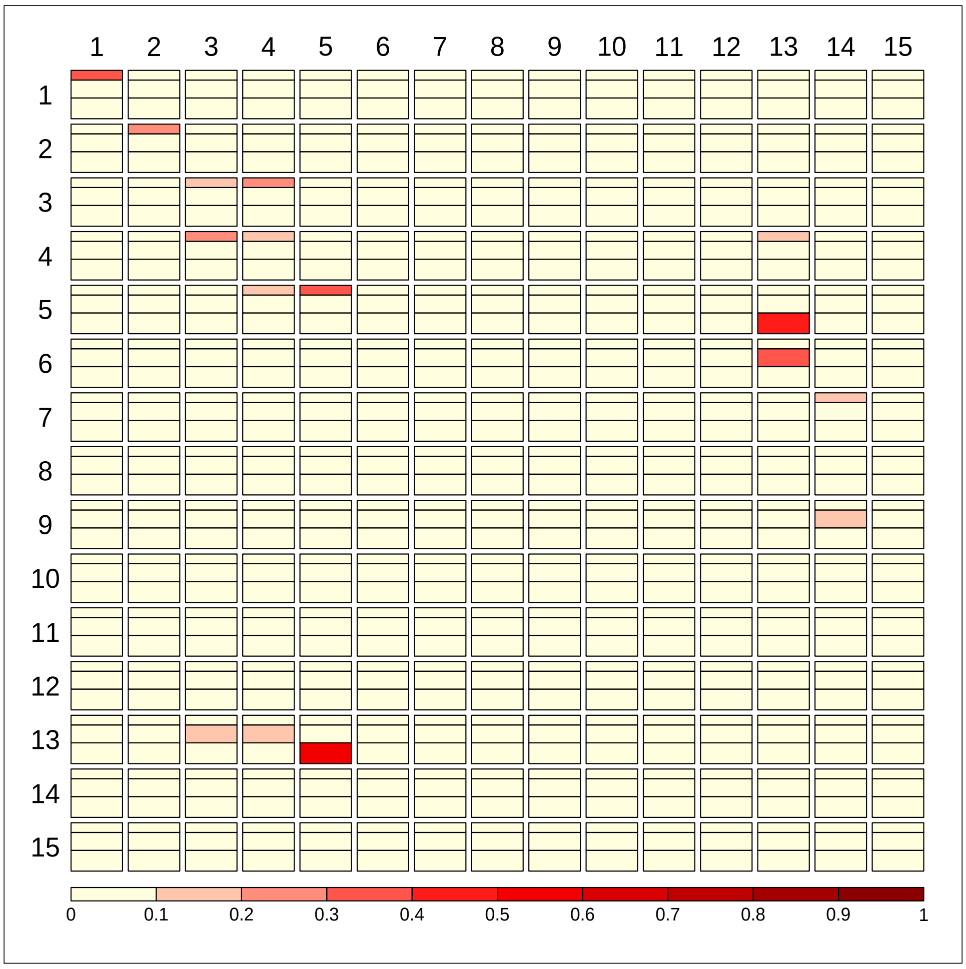

Figure 3: Click-plot representing the clustering solution obtained by the backward and forward selection procedures with $K = 3$ for the illustrative example of Section 4.

middle segment of each cell, is mostly characterized by the transitions 6–13, 13–3, 13–4, and 9–14. The final group, represented by the bottom cell segments, is almost entirely driven by the transitions between states 5 and 13. The corresponding cells include bright and dark red color hues. All the above-mentioned findings agree well with the model parameters specified in Table 10.

Based on the click-plot, we can also notice that states within the detected equivalence blocks behave similarly. To see this, we need to compare columns associated with states within blocks as well as rows cell-wise. Thus, for the block {3, 4}, the relation between states 3 and 4 is reflected through similar color hues (and thus transition probabilities) in cells associated with columns 3 and 4 as well as rows 3 and 4. Some minor distinction is observed between cells 5–3 and 5–4 as well as 3–13 and 4–13. However, all other pairs of cells (1–3 and 1–4, 2–3 and 2–4, 3–3 and 3–4, etc., as well as 3–1 and 4–1, 3–2 and 4–2, etc.) support combining states 3 and 4 into the common equivalence block. Similar analysis can be provided for the blocks {8, 9, 10} and {11, 12, 14, 15}. The majority of cells associated with these two blocks are pale yellow.

Method	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$
k-means	0.000	0.054	0.313	0.372	0.284
k-medoids	0.000	0.148	0.152	0.143	0.135
Ward	0.000	-0.010	0.111	0.098	0.101
single	0.000	0.010	0.006	0.005	0.001
complete	0.000	0.010	0.019	0.012	0.022
average	0.000	0.010	0.021	0.017	0.015

Table 12: ARI results of various clustering procedures from the package **WeightedCluster** and k-means algorithm from the package **clickstream**.

Finally, it is worth remarking again that the considered illustrative example is rather challenging as the sample size is relatively small and the vast majority of transition probabilities do not differ considerably over the three components. Despite this, the proposed biclustering procedure gives impressive results. The complexity of the considered dataset can also be seen from the results obtained by the k-means algorithm implemented in the package **clickstream** (Table 12). This k-means algorithm is applied to relative transition frequency matrices. The highest ARI value of 0.372 is reached for $K = 4$, while for the correct $K = 3$ the corresponding ARI is equal to 0.313. Both ARI values are considerably lower than those obtained by model-based clustering approaches with or without state selection procedures.

Table 12 also provides ARI values for clustering methods available through the package **WeightedCluster**. There are k-medoids and four hierarchical algorithms with various links. For all the five methods, the dissimilarity measure was calculated based on the longest common subsequence approach. As we can see, none of the considered methods is capable of capturing the underlying clustering well. Of course, this result does not imply that our proposed approach should be blindly preferred over competitors as the choice of the partitioning method should be dictated by the goal of the study. When studying transition patterns is the priority, our developed procedure proves to be a strong candidate.

5. Application to clickstream analysis

In this section, we consider a real-life dataset and analyze it by means of **ClickClust**. The dataset `msnbc323` (Melnykov 2016a) is available in the package and can be accessed by the command `data("msnbc323", package = "ClickClust")`. This is a reduced version of the dataset `msnbc` analyzed by Cadez *et al.* (2003). There are 323 clickstream sequences that involve 17 different states: (1) *frontpage*, (2) *news*, (3) *tech*, (4) *local*, (5) *opinion*, (6) *on-air*, (7) *miscellaneous*, (8) *weather*, (9) *msn-news*, (10) *health*, (11) *living*, (12) *business*, (13) *msn-sports*, (14) *sports*, (15) *summary*, (16) *bbs*, and (17) *travel*. The length of sequences varies from 35 to 362. There are 289 possible transitions among 17 states. Indeed, few transitions are represented in the observed individual sequences, which makes the analyzed dataset rather challenging for clustering.

Table 13 contains the results of the analysis by means of **ClickClust**. They can be reproduced by running the command `demo("msnbc323", package = "ClickClust")`. The corresponding code is provided in the supplementary file `v74i09.R`. As state selection procedures take a rather long time in this case (and also do not provide an improvement in terms of BIC), we focus on the direct application of the EM algorithm without the analysis of states. As we can

Method	$K = 1$	$K = 2$	$K = 3$	$K = 4$	$K = 5$
Model with β	115,342	113,418	**113,262**	113,628	114,473
Model without β	115,868	113,906	**113,672**	114,084	114,787

Table 13: Results of the EM algorithm with and without initial state probabilities included in the model. Corresponding BIC values are provided for each number of components $K = 1, \ldots, 5$. BIC values given in bold font represent the best models detected.

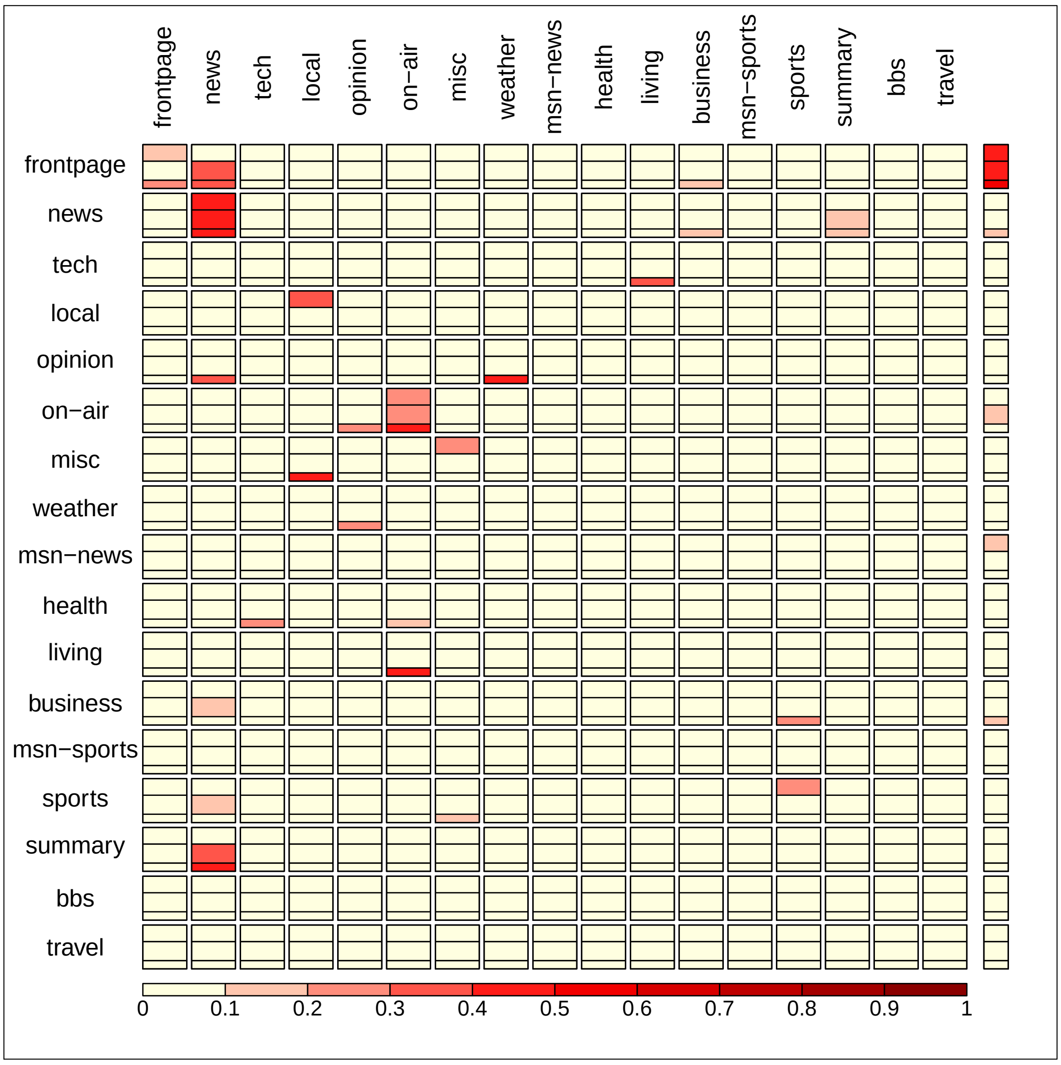

Figure 4: Click-plot representing the best clustering solution obtained for the clickstream application from Section 5.

see from Table 13, the best model has three components and involves initial state probabilities β. The corresponding BIC value is 113,262. Of course, ARI values cannot be calculated as the true partition is unavailable. Therefore, we assess and interpret the obtained solution by means of the click-plot provided in Figure 4.

As we can see, the cluster reflected by top cell segments is entirely driven by transitions within the same categories. In this cluster the red color hues correspond to diagonal cells only: *frontpage–frontpage*, *news–news*, *local–local*, *on-air–on-air*, *miscellaneous–miscellaneous*, and *sports–sports*. No other cell in Figure 4 has the color of its top segment other than pale yellow. Thus, this group represents people who make the majority of transitions within their preferred category and do not change categories frequently. It is worth remarking that the detection of such a cluster is rather typical in the analysis of clickstream sequences. The second cluster is characterized by higher probabilities of transitions *frontpage–news*, *news–news*, *news–summary*, *on-air–on-air*, *business–news*, *sports–news*, and *summary–news*. Six of these seven transitions involve the state *news*. The transition *frontpage–news* reflects the common pattern for the users starting with the category *frontpage* to proceed directly to the category *news*. Once the reader gets to the category *news*, he typically stays within it or proceeds to *summary*. Thus, the second cluster consists of people mostly concerned with news. The final data group represented by bottom cell segments is characterized by multiple transitions, thus highlighting a variety of trends. Transitions *opinion–news*, *opinion–weather*, *on-air–opinion*, *on-air–on-air*, *weather–opinion*, *health–on-air*, and *living–on-air* describe the behavior of users mostly interested in the rubrics *opinion* and *on-air*. From Figure 4, we can also conclude that users in the third cluster move from the category *on-air* to the category *opinion* and do not return even through other rubrics. Some other interesting trends highlighted by the click-plot suggest that people visiting the category *health* mostly move to the state *on-air* either directly or through the states *tech* and *living*. This conclusion is supported by the transition *health–on-air* and the sequence of transitions *health–tech*, *tech–living*, *living–on-air*. Another interesting sequence involves states *business*, *sports*, *miscellaneous*, and *local*. More specifically, transitions *frontpage–business*, *news–business*, *business–sports*, *sports–miscellaneous*, and *miscellaneous–local* suggest the common sequence of rubric transitions: *frontpage/news–business–sports–miscellaneous–local*. This sequence has great meaning as it is rather common for many people to follow this trend and mass media often present their material in a similar order.

With regard to the initial states, we can see that the majority of users in the three clusters start their clickstreams in the category *frontpage*. At the same time, some sequences from the first cluster also begin in the rubric *msn-news*, sequences from the second cluster sometimes start in the state *on-air*, while clickstreams from the last group can also originate in the categories *news* and *business*.

The analysis of this interesting but challenging dataset illustrates how click-plots can be used and interpreted to discover interesting navigation patterns common for observations within detected clusters.

6. Summary

In this paper, we introduced and discussed the R package **ClickClust**. The proposed software provides an efficient tool for model-based clustering and biclustering of categorical sequences. Necessary methodological details are carefully summarized and all package capabilities are thoroughly discussed and illustrated on demo examples. One challenging example is considered to highlight the utility of the package and to emphasize its usefulness. A real-life application to the analysis of clickstream sequences is studied. **ClickClust** can be of interest for researchers in the areas of clustering and classification as well as for a broad group of data mining practitioners.

References

Banfield J, Raftery A (1993). "Model-Based Gaussian and Non-Gaussian Clustering." *Biometrics*, **49**(3), 803–821. doi:10.2307/2532201.

Biernacki C, Celeux G, Govaert G (2003). "Choosing Starting Values for the EM Algorithm for Getting the Highest Likelihood in Multivariate Gaussian Mixture Models." *Computational Statistics & Data Analysis*, **41**(3–4), 561–575. doi:10.1016/s0167-9473(02)00163-9.

Brzinsky-Fay C, Kohler U, Luniak M (2006). "Sequence Analysis with Stata." *Stata Journal*, **6**(4), 435–460.

Cadez I, Heckerman D, Meek C, Smyth P, White S (2003). "Model-Based Clustering and Visualization of Navigation Patterns on a Web Site." *Data Mining and Knowledge Discovery*, **7**(4), 399–424. doi:10.1023/A:1024992613384.

Chen WC, Maitra R (2011). "Model-Based Clustering of Regression Time Series Data via APECM – An AECM Algorithm Sung to an Even Faster Beat." *Statistical Analysis and Data Mining*, **4**(6), 567–578. doi:10.1002/sam.10143.

Dempster A, Laird N, Rubin D (1977). "Maximum Likelihood for Incomplete Data via the EM Algorithm." *Journal of the Royal Statistical Society B*, **39**(1), 1–38.

Forgy E (1965). "Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classifications." *Biometrics*, **21**, 768–780.

Fraley C, Raftery A (2002). "Model-Based Clustering, Discriminant Analysis, and Density Estimation." *Journal of the American Statistical Association*, **97**(458), 611–631. doi:10.1198/016214502760047131.

Gabadinho A, Ritschard G, Müller N, Studer M (2011). "Analyzing and Visualizing State Sequences in R with **TraMineR**." *Journal of Statistical Software*, **40**(4), 1–37. doi:10.18637/jss.v040.i04.

Halpin B (2014). "**SADI**: Sequence Analysis Tools for Stata." *University of Limerick Department of Sociology Working Paper Series*. URL http://www.ul.ie/sociology/pubs/wp2014-03.pdf.

Handcock M, Raftery A, Tantrum J (2007). "Model-Based Clustering for Social Networks." *Journal of the Royal Statistical Society A*, **170**(2), 301–354. doi:10.1111/j.1467-985x.2007.00471.x.

Harte D (2016). ***HiddenMarkov**: Hidden Markov Models.* R package version 1.8-7, URL https://CRAN.R-project.org/package=HiddenMarkov.

Hubert L, Arabie P (1985). "Comparing Partitions." *Journal of Classification*, **2**(1), 193–218. doi:10.1007/bf01908075.

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

Liao T (2005). "Clustering of Time Series Data – A Survey." *Pattern Recognition*, **38**(11), 1857–1874. doi:10.1016/j.patcog.2005.01.025.

MacQueen J (1967). "Some Methods for Classification and Analysis of Multivariate Observations." In *Proceedings of the Fifth Berkeley Symposium*, volume 1, pp. 281–297. University of California Press.

McLachlan G, Peel D (2000). *Finite Mixture Models*. John Wiley & Sons, New York. doi:10.1002/0471721182.

Melnykov V (2012). "Efficient Estimation in Model-Based Clustering of Gaussian Regression Time Series." *Statistical Analysis and Data Mining*, **5**(2), 95–99. doi:10.1002/sam.11138.

Melnykov V (2013a). "Challenges in Model-Based Clustering." *WIREs: Computational Statistics*, **5**(2), 135–148. doi:10.1002/wics.1248.

Melnykov V (2013b). "Finite Mixture Modelling in Mass Spectrometry Analysis." *Journal of the Royal Statistical Society C*, **62**(4), 573–592. doi:10.1111/rssc.12010.

Melnykov V (2016a). "Model-Based Biclustering of Clickstream Data." *Computational Statistics & Data Analysis*, **93**, 31–45. doi:10.1016/j.csda.2014.09.016.

Melnykov V (2016b). ***ClickClust**: An R Package for Model-Based Clustering of Categorical Sequences.* R package version 1.1.5, URL https://CRAN.R-project.org/package=ClickClust.

Melnykov V, Chen WC, Maitra R (2012). "**MixSim**: An R Package for Simulating Data to Study Performance of Clustering Algorithms." *Journal of Statistical Software*, **51**(12), 1–25. doi:10.18637/jss.v051.i12.

Nigam K, McCallum A, Thrun S, Mitchell T (2000). "Text Classification from Labeled and Unlabeled Documents Using EM." *Machine Learning*, **39**(2), 103–134. doi:10.1023/a:1007692713085.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Scholz M (2016a). ***clickstream**: An R Package for Analyzing Clickstreams.* R package version 1.1.9, URL https://CRAN.R-project.org/package=clickstream.

Scholz M (2016b). "R Package **clickstream**: Analyzing Clickstream Data with Markov Chains." *Journal of Statistical Software*, **74**(4), 1–17. doi:10.18637/jss.v074.i04.

Schwarz G (1978). "Estimating the Dimensions of a Model." *The Annals of Statistics*, **6**(2), 461–464. doi:10.1214/aos/1176344136.

Sneath P (1957). "The Application of Computers to Taxonomy." *Journal of General Microbiology*, **17**, 201–226. doi:10.1099/00221287-17-1-201.

Sorensen T (1948). "A Method of Establishing Groups of Equal Amplitude in Plant Sociology Based on Similarity of Species Content and Its Application to Analyses of the Vegetation on Danish Commons." *Biologiske Skrifter*, **5**, 1–34.

StataCorp (2015). *Stata Data Analysis Statistical Software: Release 14*. StataCorp LP, College Station. URL http://www.stata.com/.

Studer M (2013). "**WeightedCluster** Library Manual: A Practical Guide to Creating Typologies of Trajectories in the Social Sciences with R." *LIVES Working Papers*, NCCR LIVES. doi:10.12682/lives.2296-1658.2013.24.

Visser I, Speekenbrink M (2010). "**depmixS4**: An R Package for Hidden Markov Models." *Journal of Statistical Software*, **36**(7), 1–21. doi:10.18637/jss.v036.i07.

Ward J (1963). "Hierarchical Grouping to Optimize an Objective Function." *Journal of the American Statistical Association*, **58**(301), 236–244. doi:10.2307/2282967.

Affiliation:

Volodymyr Melnykov
Department of Information Systems, Statistics, and Management Science
The University of Alabama
Tuscaloosa, AL 35487, United States of America
E-mail: vmelnykov@cba.ua.edu
URL: http://bama.ua.edu/~vmelnykov

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
October 2016, Volume 74, Issue 9 *Submitted:* 2014-10-01
doi:10.18637/jss.v074.i09 *Accepted:* 2015-09-04