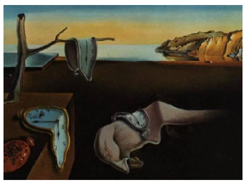

Journal of Statistical Software

October 2016, Volume 74, Issue 4. *doi: 10.18637/jss.v074.i04*

R Package clickstream: Analyzing Clickstream Data with Markov Chains

Michael Scholz
University of Passau

Abstract

Clickstream analysis is a useful tool for investigating consumer behavior, market research and software testing. I present the **clickstream** package which provides functionality for reading, clustering, analyzing and writing clickstreams in R. The package allows for a modeling of lists of clickstreams as zero-, first- and higher-order Markov chains. I illustrate the application of **clickstream** for a list of representative clickstreams from an online store.

Keywords: clickstream, Markov chain, R.

1. Introduction

Online retailers analyze their visitors to improve their stores, marketing strategies and product offers. A lot of information such as customer reviews, purchase histories, demographic characteristics or the sequence of clicks in an online store is available for such a type of analysis. R supports online retailers in preparing, analyzing and visualizing most of this information. For example, package **tm** (Feinerer and Hornik 2008) provides functionality for preparing textual customer reviews such that their sentiment can be detected in a next step using support vector machines with package **e1071** (Meyer, Dimitriadou, Hornik, Weingessel, and Leisch 2015). Sequences of clicks, so called clickstreams, can either be analyzed by mining sequential patterns with algorithms such as Apriori or PrefixSpan (Pitman and Zanker 2010; Pei, Han, Mortazavi-Asl, Wang, Pinto, Chen, Dayal, and Hsu 2004) or with probabilistic models such as Markov chains (Montgomery, Li, Srinivasan, and Liechty 2004). Whereas sequential pattern mining is supported by **arules** (Hahsler, Grün, and Hornik 2005) and **arulesSequences** (Buchta and Hahsler 2016), packages for modeling clickstreams with Markov chains are missing so far. The package **markovchain** (Spedicato, Kang, and Yalamanchi 2016) only allows modeling zero- or first-order Markov chains and is furthermore limited to one stream of clicks.

Online stores collect and analyze collections of clickstreams, though.

This paper introduces the R package **clickstream** (Scholz 2016), a package for importing, analyzing and exporting clickstreams which is available from the Comprehensive R Archive Network (CRAN) at `https://CRAN.R-project.org/package=clickstream`. In contrast to other sequential data sets, clickstreams are collections of data sequences with different sizes. Two subsequent clicks might furthermore represent the same state. Consider for example the following extract of user sessions from an online store:

Session 1: `P1 P2 P1 P3 P4 Defer`
Session 2: `P3 P4 P1 P3 Defer`
Session 3: `P5 P1 P6 P7 P6 P7 P8 P7 Buy`
Session 4: `P9 P2 P11 P12 P11 P13 P11 Buy`
Session 5: `P4 P6 P11 P6 P1 P3 Defer`
Session 6: `P3 P13 P12 P4 P12 P1 P4 P1 P3 Defer`
Session 7: `P10 P5 P10 P8 P8 P5 P1 P7 Buy`
Session 8: `P9 P2 P1 P9 P3 P1 Defer`
Session 9: `P5 P8 P5 P7 P4 P1 P6 P4 Defer`

Each click on one out of 13 possible product pages in the online store is represented by the letter `P` and the product page identifier (1–13) while the final click represents a users decision that is either to defer the purchase (`Defer`) or to buy a product (`Buy`). `Defer` and `Buy` are here absorbing states. **clickstream** is suitable to handle clickstreams with and without absorbing states. Analyzing collections of clickstreams with R is challenging, as (i) R does not directly support importing data sets with varying row length, (ii) packages such as **markovchain** (Spedicato *et al.* 2016) only support analyzing a single sequence of data (not collections of sequences), and (iii) there is no package available for R which provides functions for reading, writing, clustering and analyzing clickstreams, yet.

The **clickstream** package provides a flexible basic infrastructure for importing, exporting and analyzing sets of clickstreams as recorded by most online stores. More precisely, the package provides functionality for clustering clickstreams, visualizing clickstreams and predicting future clicks for a given session.

The remainder of this paper is structured as follows. Section 2 introduces the operations for reading, writing and generating collections of clickstreams. Section 3 presents functions for analyzing clickstreams and presents the estimation method implemented in **clickstream** for fitting higher-order Markov chains with a moderate number of parameters. Section 4 demonstrates the usage of **clickstream** on an example of large simulated clickstream data. Alternative approaches to modeling clickstreams with higher-order Markov chains are discussed in Section 5. The paper concludes in Section 6 with a discussion of possible extensions of **clickstream**.

2. Clickstreams

A clickstream is a sequence of click events for exactly one session with an online store user. The clickstreams of different sessions typically differ in type and number of click events. Each click event is of type `character`. The clickstreams for a particular session can then be modeled as a vector, whereas a collection of clickstreams can be modeled as a list in R. The package **clickstream** provides an S3 class for storing lists of vectors of click events.

Package **clickstream** provides functions to generate clickstreams in three ways. First, they can be manually generated by creating a new instance of the S3 class '`Clickstreams`'. We can create the list of clickstreams presented in Section 1 with the following code:

```
R> cls <- list(Session1 = c("P1", "P2", "P1", "P3", "P4", "Defer"),
+    Session2 = c("P3", "P4", "P1", "P3", "Defer"),
+    Session3 = c("P5", "P1", "P6", "P7", "P6", "P7", "P8", "P7", "Buy"),
+    Session4 = c("P9", "P2", "P11", "P12", "P11", "P13", "P11", "Buy"),
+    Session5 = c("P4", "P6", "P11", "P6", "P1", "P3", "Defer"),
+    Session6 = c("P3", "P13", "P12", "P4", "P12", "P1", "P4", "P1", "P3",
+      "Defer"),
+    Session7 = c("P10", "P5", "P10", "P8", "P8", "P5", "P1", "P7", "Buy"),
+    Session8 = c("P9", "P2", "P1", "P9", "P3", "P1", "Defer"),
+    Session9 = c("P5", "P8", "P5", "P7", "P4", "P1", "P6", "P4", "Defer"))
R> class(cls) <- "Clickstreams"
```

Second, the function `randomClickstreams` can be called to randomly generate an object of class '`Clickstreams`'. This function requires five arguments – a vector of possible states, a vector of start probabilities, a first-order Markov chain transition matrix, the mean length of clickstreams and the number of clickstreams. We can generate 100 clickstreams with average length of 10, two possible states ($P1$ and $P2$), start probability of 0.5 for both states and transition probabilities of $P_{ij} = \{P1 \rightarrow P1 = 0.2, P1 \rightarrow P2 = 0.8, P2 \rightarrow P1 = 0.4, P2 \rightarrow P2 = 0.6\}$ as follows:

```
R> cls <- randomClickstreams(states = c("P1", "P2"),
+    startProbabilities = c(0.5, 0.5),
+    transitionMatrix = matrix(c(0.2, 0.8, 0.4, 0.6), nrow = 2),
+    meanLength = 10, n = 100)
```

And third, we can generate a '`Clickstreams`' object by reading a list of clickstreams from a file. The function `readClickstreams()` expects a comma-separated file in which each line corresponds to exactly one clickstream. The first entry of each line can optionally be used as session name. If clickstreams were generated without session names a unique numeric identifier is used instead. The file `sample.csv` contains the clickstreams of the example in Section 1 as

```
Session1,P1,P2,P1,P3,P4,Defer
Session2,P3,P4,P1,P3,Defer
Session3,P5,P1,P6,P7,P6,P7,P8,P7,Buy
Session4,P9,P2,P11,P12,P11,P13,P11,Buy
Session5,P4,P6,P11,P6,P1,P3,Defer
Session6,P3,P13,P12,P4,P12,P1,P4,P1,P3,Defer
Session7,P10,P5,P10,P8,P8,P5,P1,P7,Buy
Session8,P9,P2,P1,P9,P3,P1,Defer
Session9,P5,P8,P5,P7,P4,P1,P6,P4,Defer
```

and is imported via

```
R> cls <- readClickstreams(file = "sample.csv", sep = ",", header = TRUE)
R> cls
```

The corresponding output is

```
Clickstreams

Session1: P1 P2 P1 P3 P4 Defer
Session2: P3 P4 P1 P3 Defer
Session3: P5 P1 P6 P7 P6 P7 P8 P7 Buy
Session4: P9 P2 P11 P12 P11 P13 P11 Buy
Session5: P4 P6 P11 P6 P1 P3 Defer
Session6: P3 P13 P12 P4 P12 P1 P4 P1 P3 Defer
Session7: P10 P5 P10 P8 P8 P5 P1 P7 Buy
Session8: P9 P2 P1 P9 P3 P1 Defer
Session9: P5 P8 P5 P7 P4 P1 P6 P4 Defer
```

The function `summary()` provides the basic information for a '`Clickstreams`' object.

```
R> summary(cls)

Observations: 9

Click Frequencies:
  Buy Defer  P1  P10  P11  P12  P13   P2   P3   P4   P5   P6   P7   P8   P9
    3     6  11    2    4    3    2    3    7    7    5    5    5    4    3
```

clickstream provides a function for exporting a '`Clickstreams`' object to file. As for reading clickstreams from a file, we need to specify the file name, the separator and a Boolean flag indicating if the clickstreams will have a session name. We additionally must define the name of the '`Clickstreams`' object we want to store and we can optionally specify whether the click events will be quoted. A '`Clickstreams`' object `cls` can be written to a file by calling

```
R> writeClickstreams(cls, "sample.csv", header = TRUE, sep = ",")
```

3. Analyzing clickstreams

3.1. Markov chains

A Markov chain is a stochastic process $X^{(n)}$ that takes state m_n from a finite set M at each time n. If the state in n only depends on the recent k states, we call $X^{(n)}$ a Markov chain of order k. The probability to be in any of the m states in the next step is hence independent of the present state in a zero-order Markov chain. Time homogeneous Markov chains, where the transition probability is independent of time n, can be described by transition matrices, where $P_{ij}^{(k)}$ describes the probability to obtain a transition from state i at time $n-k$ to state

j at time n. Each probability $P_{ij}^{(k)}$ corresponds to a parameter to be estimated. Higher-order Markov chains are thus characterized by $(m-1)m^k$ model parameters. The major challenge of using higher-order Markov chains for analyzing clickstream data is the number of parameters that exponentially increases with the order of the Markov chain. Users of an online store typically decide to visit one of several possible websites not only based on the website they are currently visiting. Usually, a user considering a product page might either add the product to the shopping cart, view product reviews, follow a product recommendation, or search for another product. Moe (2003) proposes that the probability for a transition to either of the possible next states depends on the mode (browsing, searching, or buying) the user is currently in. This mode can be identified when considering the recent k states (websites) of a user rather than only the last state (website). Higher-order Markov chains are hence more promising when analyzing clickstream data.

3.2. Fitting a Markov chain model

Raftery (1985) has proposed a model for higher-order Markov chains that can be estimated with one additional parameter for each order k. His model is based on the idea that the distribution of state probabilities X can be approximated as weighted sum of the last k transition probabilities:

$$X^{(n+k+1)} = \sum_{i=1}^{k} \lambda_i \mathbf{Q} X^{(n+k+1-i)} \tag{1}$$

s.t.

$$\sum_{i=1}^{k} \lambda_i = 1, \qquad \lambda_i \geq 0 \quad \forall i. \tag{2}$$

$\mathbf{Q}$ is a $m \times m$ transition probability matrix and λ_i denotes the weight for each lag i in the model. Ching, Huang, Ng, and Siu (2013) introduced a more general form of Raftery's model by defining lag-specific transition probability matrices $\mathbf{Q}_i$:

$$X^{(n+k+1)} = \sum_{i=1}^{k} \lambda_i \mathbf{Q}_i X^{(n+k+1-i)} \tag{3}$$

with the same constraints as defined in Equation 2.

$\mathbf{Q}_i$ is a non-negative $m \times m$ matrix with column sums equal to one. This generalized model has $k + km^2$ parameters. We can estimate $\hat{\mathbf{Q}}_i$ by observing the transition probability from $n - i$ to n. State probabilities are estimated from the sequence $X^{(n)}$. We are now able to derive the following optimization problem from Equation 3 to estimate the lag parameters λ:

$$\min_{\lambda} \left\{ \left\| \sum_{i=1}^{k} \lambda_i \hat{\mathbf{Q}}_i \hat{X} - \hat{X} \right\| \right\} \tag{4}$$

subject to the constraints defined in Equation 2.

The function `fitMarkovChain()` estimates the parameters of a Markov chain model of order k for a given '`Clickstreams`' object by solving Equation 4 either as a linear problem or as a quadratic problem. Optimization parameters such as the used optimizer are specified as a list of control parameters in function `fitMarkovChain()`. A Markov chain is fitted for an object `cls` via

```
R> mc <- fitMarkovChain(clickstreamList = cls, order = 2,
+    control = list(optimizer = "quadratic"))
R> mc
```

The corresponding output shows the transition probability matrices $\mathbf{Q}_i$ and the lag parameters λ_i for each of the two specified lags (i.e., $k = 2$)[1].

```
Higher-Order Markov Chain (order=2)

Transition Probabilities:

Lag:  1
lambda:  0.22
      Buy Defer    P1 P10  P11  P12 P13   P2   P3   P4  P5  P6  P7   P8   P9
Buy     0     0 0.000 0.0 0.25 0.00 0.0 0.00 0.00 0.00 0.0 0.0 0.4 0.00 0.00
Defer   0     0 0.091 0.0 0.00 0.00 0.0 0.00 0.43 0.29 0.0 0.0 0.0 0.00 0.00
P1      0     0 0.000 0.0 0.00 0.33 0.0 0.67 0.14 0.43 0.4 0.2 0.0 0.00 0.00
P10     0     0 0.000 0.0 0.00 0.00 0.0 0.00 0.00 0.00 0.2 0.0 0.0 0.00 0.00
P11     0     0 0.000 0.0 0.00 0.33 0.5 0.33 0.00 0.00 0.0 0.2 0.0 0.00 0.00
P12     0     0 0.000 0.0 0.25 0.00 0.5 0.00 0.00 0.14 0.0 0.0 0.0 0.00 0.00
P13     0     0 0.000 0.0 0.25 0.00 0.0 0.00 0.14 0.00 0.0 0.0 0.0 0.00 0.00
P2      0     0 0.091 0.0 0.00 0.00 0.0 0.00 0.00 0.00 0.0 0.0 0.0 0.00 0.67
P3      0     0 0.364 0.0 0.00 0.00 0.0 0.00 0.00 0.00 0.0 0.0 0.0 0.00 0.33
P4      0     0 0.091 0.0 0.00 0.33 0.0 0.00 0.29 0.00 0.0 0.2 0.2 0.00 0.00
P5      0     0 0.000 0.5 0.00 0.00 0.0 0.00 0.00 0.00 0.0 0.0 0.0 0.50 0.00
P6      0     0 0.182 0.0 0.25 0.00 0.0 0.00 0.00 0.14 0.0 0.0 0.2 0.00 0.00
P7      0     0 0.091 0.0 0.00 0.00 0.0 0.00 0.00 0.00 0.2 0.4 0.0 0.25 0.00
P8      0     0 0.000 0.5 0.00 0.00 0.0 0.00 0.00 0.00 0.2 0.0 0.2 0.25 0.00
P9      0     0 0.091 0.0 0.00 0.00 0.0 0.00 0.00 0.00 0.0 0.0 0.0 0.00 0.00

Lag:  2
lambda:  0.78
      Buy Defer  P1 P10  P11  P12 P13   P2   P3  P4  P5  P6   P7   P8   P9
Buy     0     0 0.1 0.0 0.00 0.00 0.5 0.00 0.00 0.0 0.0 0.0 0.00 0.25 0.00
Defer   0     0 0.3 0.0 0.00 0.00 0.0 0.00 0.50 0.0 0.0 0.2 0.00 0.00 0.00
P1      0     0 0.2 0.0 0.33 0.00 0.0 0.00 0.25 0.2 0.0 0.0 0.33 0.25 0.67
P10     0     0 0.0 0.5 0.00 0.00 0.0 0.00 0.00 0.0 0.0 0.0 0.00 0.00 0.00
P11     0     0 0.0 0.0 0.67 0.00 0.0 0.00 0.00 0.2 0.0 0.0 0.00 0.00 0.33
P12     0     0 0.0 0.0 0.00 0.33 0.0 0.33 0.25 0.0 0.0 0.0 0.00 0.00 0.00
P13     0     0 0.0 0.0 0.00 0.33 0.0 0.00 0.00 0.0 0.0 0.0 0.00 0.00 0.00
P2      0     0 0.0 0.0 0.00 0.00 0.0 0.00 0.00 0.0 0.0 0.0 0.00 0.00 0.00
P3      0     0 0.1 0.0 0.00 0.00 0.0 0.33 0.00 0.4 0.0 0.2 0.00 0.00 0.00
P4      0     0 0.2 0.0 0.00 0.33 0.5 0.00 0.00 0.0 0.2 0.0 0.00 0.00 0.00
P5      0     0 0.0 0.0 0.00 0.00 0.0 0.00 0.00 0.0 0.2 0.0 0.00 0.25 0.00
P6      0     0 0.0 0.0 0.00 0.00 0.0 0.00 0.00 0.2 0.2 0.4 0.00 0.00 0.00
P7      0     0 0.1 0.0 0.00 0.00 0.0 0.00 0.00 0.0 0.2 0.0 0.67 0.25 0.00
```

[1]Note that the number of digits was set to 2 for this output.

```
P8      0      0 0.0 0.5 0.00 0.00 0.0 0.00 0.00 0.0 0.2 0.2 0.00 0.00 0.00
P9      0      0 0.0 0.0 0.00 0.00 0.0 0.33 0.00 0.0 0.0 0.0 0.00 0.00 0.00

Start Probabilities:

  P1  P10   P3   P4   P5   P9
0.11 0.11 0.22 0.11 0.22 0.22

End Probabilities:

  Buy Defer
 0.33  0.67
```

Start (end) probabilities are shown for the states the corresponding clickstreams started (ended) with. We can, for example, see that 22% of our clickstreams started with a click on product 3 (`P3`) and 33% of the sessions ended with a purchase (`Buy`).

The result of function `fitMarkovChain()` is an instance of the S4 class '`MarkovChain`'. Objects of class '`MarkovChain`' consist of the following slots:

- `states`: A vector of all states.
- `order`: The order k of the Markov chain.
- `transitions`: A list of k transition matrices.
- `lambda`: A vector of k lag parameters λ.
- `logLikelihood`: Log-likelihood of the fitted Markov chain model.
- `observations`: Number of observations used to fit the Markov chain model.
- `start`: Probability of each state to be the first state of a clickstream.
- `end`: Probability of each state to be the last state of a clickstream.
- `transientStates`: A vector of transient states.
- `absorbingStates`: A vector of absorbing states.
- `absorbingProbabilities`: Probability of each absorbing state that a clickstream ends with that state.

`fitMarkovChain()` computes the log-likelihood of a '`MarkovChain`' object based on the $m \times m$ transition frequency matrices $\mathbf{F}_i$:

$$LL = \sum_{i=1}^{k} \lambda_i \mathbf{F}_i \log \left(\frac{\mathbf{F}_i}{1_s \mathbf{F}_i} \right). \tag{5}$$

Based on this log-likelihood the method `summary()` returns Akaike's information criterion (AIC) and Bayes' information criterion (BIC) to compare two fitted '`MarkovChain`' objects.

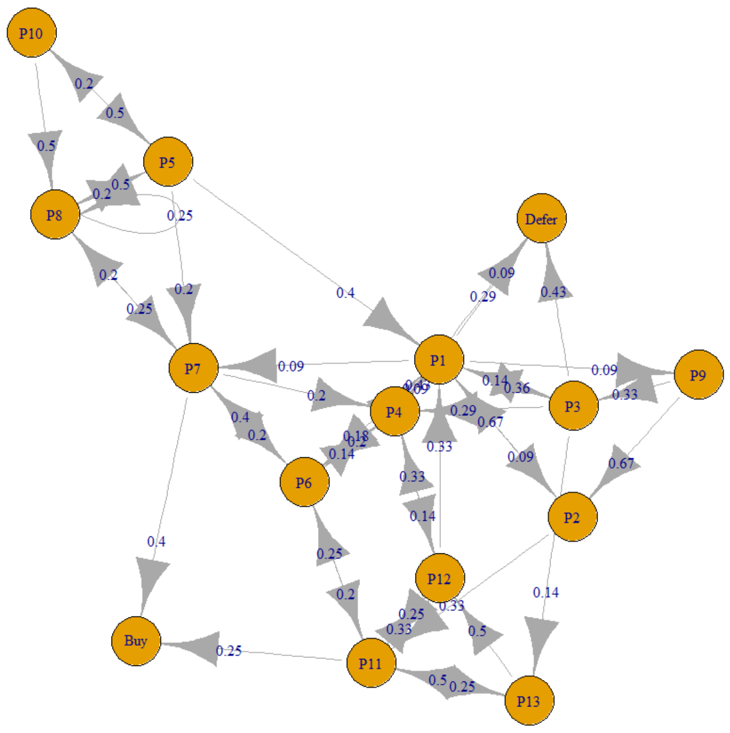

Figure 1: The plot illustrates the transitions between clicks with lag $k = 2$.

```
R> summary(mc)

Higher-Order Markov Chain (order=2) with 15 states.
 The Markov Chain has absorbing states.

Observations: 70
LogLikelihood: -66.1378
AIC: 196.2756
BIC: 268.2274
```

clickstream uses **igraph** (Csardi and Nepusz 2006) to plot a transition matrix of lag k as a directed graph (see Figure 1). A simple bar chart is plotted for zero-order Markov chains showing the frequencies of the possible states.

```
R> plot(mc, order = 2)
```

3.3. Clustering clickstreams

When exploring clickstream data, we will observe that a large number of similar clickstreams rarely exists. This is due to the complexity of most websites and the amount of paths that a user can follow to come to a particular website. Huang, Ng, Ching, Ng, and Cheung (2001) propose to use clustering algorithms to cluster a list of clickstreams. **clickstream** offers the function `clusterClickstreams()` for clustering clickstreams based on a transition matrix of order k with the k-means algorithm. This function has 3 parameters – a list of clickstreams, the order k for which a transition matrix is computed and used as basis for clustering, and the number of cluster centers. Since `clusterClickstreams()` directly calls `kmeans()` from package **stats**, it allows specifying further parameters such as the maximal number of iterations or the algorithm used to compute the clusters. We can cluster a '`Clickstreams`' object `cls` based on its first-order transition matrix into 3 groups via

```
R> clusters <- clusterClickstreams(clickstreamList = cls, order = 1,
+    centers = 3)
R> clusters
```

```
[[1]]
Clickstreams

Session1: P1 P2 P1 P3 P4 Defer
Session4: P9 P2 P11 P12 P11 P13 P11 Buy
Session8: P9 P2 P1 P9 P3 P1 Defer

[[2]]
Clickstreams

Session2: P3 P4 P1 P3 Defer
Session5: P4 P6 P11 P6 P1 P3 Defer
Session6: P3 P13 P12 P4 P12 P1 P4 P1 P3 Defer

[[3]]
Clickstreams

Session3: P5 P1 P6 P7 P6 P7 P8 P7 Buy
Session7: P10 P5 P10 P8 P8 P5 P1 P7 Buy
Session9: P5 P8 P5 P7 P4 P1 P6 P4 Defer
```

The result is a list consisting of 3 '`Clickstreams`' objects. We can now fit a '`MarkovChain`' object for each '`Clickstreams`' object or write these objects to file with `writeClickstreams()`.

3.4. Predicting clicks

Clickstream analysis is often used to predict either the next click or the final click (state) of a consumer. A consumer's next click depends on the k previous clicks in a k-order Markov chain. The probability distribution $X^{(n)}$ of the click at n is:

$$X^{(n)} = \sum_{i=1}^{k} \lambda_i Q_i X^{(n-i)} \tag{6}$$

Data analysts often are interested in those clicks just before a final decision (buy or defer). Each clickstream hence has an absorbing state which is either "Buy" or "Defer". If we know the probability B that our clickstreams will be absorbed in any of the possible absorbing states, we can use this information to more accurately predict the next click.

$$X^{(n)} = B \sum_{i=1}^{k} \lambda_i Q_i X^{(n-i)} \tag{7}$$

`clickstream` predicts the next click for a given '`Pattern`' object. A '`Pattern`' object is a (part) of a clickstream described by a sequence of clicks and optionally a probability of occurrence and a vector of absorbing probabilities. The next click is predicted with Equation 6 if no absorbing states exist or the '`Pattern`' object is specified without absorbing probabilities and with Equation 7 otherwise. To predict the next click of a given '`Pattern`' object based on a given `MarkovChain`-object, we can use the `predict()` function as follows:

```
R> pattern <- new("Pattern", sequence = c("P9", "P2"))
R> resultPattern <- predict(mc, startPattern = pattern, dist = 1)
R> resultPattern

Sequence: P1
Probability: 0.6666667
Absorbing Probabilities:
  None
1    0
```

If a user starts with the clickstream `P9 P2`, the user will most likely click on `P1` next. We can also predict the next n clicks by varying the parameter `dist`. Equation 6 or 7 is then iteratively applied.

```
R> pattern <- new("Pattern", sequence = c("P9", "P2"),
+    absorbingProbabilities = data.frame(Buy = 0.333, Defer = 0.667))
R> resultPattern <- predict(mc, startPattern = pattern, dist = 2)
R> resultPattern
```

The defined '`Pattern`' object corresponds to a user who has recently viewed products `P9` and `P2` and now a probability of 33.3% to buy a product. The prediction for the next two clicks is shown in the following output:

```
Sequence: P1 P3
Probability: 0.2618064
Absorbing Probabilities:
         Buy     Defer
1 0.05818405 0.9418159
```

Our user has a purchasing probability of 5.83% after 2 further clicks and we expect that she will visit product `P3` in two clicks and finally defers the purchase. However, the probability that she really continues visiting products `P1 P3` is only 26.17%.

Online stores often have evidence on how many of the visitors convert to a buyer or how many times a particular user has been only visiting the online store and how often she has bought a product. This information can be used to formulate initial absorbing probabilities for a user. If for example a user has been logged in and finally bought a product in 50% of her log ins, we can compute absorbing probabilities for a stream of clicks as follows:

```
R> absorbingProbabilities <- c(0.5, 0.5)
R> sequence <- c("P9", "P2")
R> for (s in sequence) {
+    absorbingProbabilities <- absorbingProbabilities *
+      data.matrix(subset(mc@absorbingProbabilities, state == s,
+      select = c("Buy", "Defer")))
+  }
R> absorbingProbabilities <- absorbingProbabilities /
+    sum(absorbingProbabilities)
R> absorbingProbabilities

         Buy     Defer
15 0.2262178 0.7737822
```

The output shows that our user has a probability of 22.62% to finally buy a product after she has visited products P9 and P2.

4. Example with simulated data

In this section, I will demonstrate the usage of **clickstream** in a simulated data example. The example models clickstreams for 100,000 user sessions that represent clicks on either one of 7 products or on one of the two final states "Buy" and "Defer". The clickstreams are generated with the following R code:

```
R> set.seed(123)
R> cls <- randomClickstreams(
+    states = c("P1", "P2", "P3", "P4", "P5", "P6", "P7", "Defer", "Buy"),
+    startProbabilities = c(0.2, 0.25, 0.1, 0.15, 0.1, 0.1, 0.1, 0, 0),
+    transitionMatrix = matrix(
+       c(0.01, 0.09, 0.05, 0.21, 0.12, 0.17, 0.11, 0.2, 0.04,
+         0.1, 0, 0.29, 0.06, 0.11, 0.13, 0.21, 0.1, 0,
+         0.07, 0.16, 0.03, 0.25, 0.23, 0.08, 0.03, 0.12, 0.03,
+         0.16, 0.14, 0.07, 0, 0.05, 0.22, 0.19, 0.1, 0.07,
+         0.24, 0.27, 0.17, 0.13, 0, 0.03, 0.09, 0.06, 0.01,
+         0.11, 0.18, 0.04, 0.15, 0.26, 0, 0.1, 0.11, 0.05,
+         0.21, 0.07, 0.08, 0.2, 0.14, 0.18, 0.02, 0.08, 0.02,
+         0, 0, 0, 0, 0, 0, 0, 0, 0,
+         0, 0, 0, 0, 0, 0, 0, 0, 0), nrow = 9),
+    meanLength = 50, n = 100000)
```

We will get a first impression on the simulated clickstreams by calling the `summary()` function as follows:

```
R> summary(cls)
```

The output shows that 100,000 observations are available and most of them do not end with a purchase.

```
Observations: 100000

Click Frequencies:
   Buy  Defer     P1     P2     P3     P4     P5     P6     P7
 22087  77813 108767 113760  86334 111054  97129  93397  89521
```

The next step in a clickstream analysis might be modeling the clickstreams with Markov chains of different orders and select that order that produces the highest fit to the data. We can implement a simple procedure for determining the "best" order for a Markov chain model as follows:

```
R> maxOrder <- 5
R> result <- data.frame()
R> for (k in 1:maxOrder) {
+     mc <- fitMarkovChain(clickstreamList = cls, order = k)
+     result <- rbind(result, c(k, summary(mc)$aic, summary(mc)$bic))
+ }
R> names(result) <- c("Order", "AIC", "BIC")
R> result
```

The maximal order `maxOrder` is the minimal length more than 50% of the clickstreams have, which is 5 in this example. The output indicates that a Markov chain with order $k = 2$ fits the data better than any Markov chain with a lower order.

```
  Order     AIC     BIC
1     1 2685427 2685543
2     2 2684008 2684240
3     3 2684028 2684376
4     4 2684048 2684512
5     5 2684068 2684648
```

We now can fit a Markov chain with order $k = 2$ or cluster the clickstreams and model the clickstreams of each cluster with a separate Markov chain. The following code clusters the ‘`Clickstreams`’ object `cls` into 5 clusters based on the transition matrix for lag $k = 1$:

```
R> clusters <- clusterClickstreams(clickstreamList = cls, order = 1,
+    centers = 5)
```

Calling the `summary()` function on the first cluster reveals that this cluster consists of 9664 clickstreams.

```
R> summary(clusters$clusters[[1]])

Observations: 9664

Click Frequencies:
  Buy Defer    P1    P2    P3    P4    P5    P6    P7
 2084  7576 19569 11750  7185 11980  9282  9081 15060
```

A comparison of the information criteria (i.e., AIC and BIC) of Markov chain models with several orders indicates that the first cluster should be modeled with a Markov chain of order $k = 2$, which is done by the following code:

```
R> mc <- fitMarkovChain(clickstreamList = clusters$clusters[[1]], order = 2)
R> summary(mc)
```

```
Higher-Order Markov Chain (order=2) with 9 states.
 The Markov Chain has absorbing states.

Observations: 93567
LogLikelihood: -145850.6
AIC: 291741.2
BIC: 291930.1
```

Approximately 22% of the sessions in the first cluster end with a purchase and 78% end with a choice deferral. This information can be used as prior distribution for the absorbing probabilities B when predicting the next clicks for a new user session. The next clicks of a user who has recently viewed products `P1 P4 P6` are predicted as follows:

```
R> pattern <- new("Pattern", sequence = c("P1", "P4", "P6"),
+    absorbingProbabilities = data.frame(Buy = 0.22, Defer = 0.78))
R> resultPattern <- predict(mc, startPattern = pattern, dist = 2)
R> resultPattern
```

The corresponding output shows that our user most likely will next click on `P2` and thereafter on `P7`. Her purchase probability reduces to 2.5% after these two additional clicks.

```
Sequence: P2 P7
Probability: 0.06835761
Absorbing Probabilities:
         Buy     Defer
1 0.02508478 0.9749152
```

Based on all clickstreams, we expect that our user will most likely continue her journey by visiting products `P5` and `P2` next. The more heterogeneous the clickstreams in a fitted Markov chain model, the lower is the probability that a current session indeed will be continued with the predicted clicks. Clustering clickstreams is thus of utmost importance especially in case of high clickstream heterogeneity.

5. Alternative approaches

There are several alternative approaches to model clickstreams. One possible approach is representing clickstreams by state frequencies instead of transition probabilities. The function `frequencies` extracts an incidence data frame from a '`Clickstreams`' object that contains the number of occurrence of each state in each clickstream.

```
R> frequencyDF <- frequencies(cls)
```

The output is a data frame with columns representing the states and rows representing the clickstreams (i.e., user sessions).

```
         Buy Defer P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13
Session1   0     1  2  1  1  1  0  0  0  0  0   0   0   0   0
Session2   0     1  1  0  2  1  0  0  0  0  0   0   0   0   0
Session3   1     0  1  0  0  0  1  2  3  1  0   0   0   0   0
Session4   1     0  0  1  0  0  0  0  0  0  1   0   3   1   1
Session5   0     1  1  0  1  1  0  2  0  0  0   0   1   0   0
Session6   0     1  2  0  2  2  0  0  0  0  0   0   0   2   1
Session7   1     0  1  0  0  0  2  0  1  2  0   2   0   0   0
Session8   0     1  2  1  1  0  0  0  0  0  2   0   0   0   0
Session9   0     1  1  0  0  2  2  1  1  1  0   0   0   0   0
```

Based on this incidence data frame, we can, for example, fit regression models to predict the final state (i.e., "Buy" or "Defer")[2]. This approach has the major drawback that the sequential structure of clickstreams is disregarded. This comes with the advantage of a higher performance when predicting new sessions' next or final clicks.

A second alternative to modeling clickstreams with higher-order Markov chains is representing them as sequential patterns. Packages **arules** (Hahsler *et al.* 2005) and **arulesSequences** (Buchta and Hahsler 2016) provide several functions for mining sequential patterns and for finding those click patterns having a particular minimum support (i.e., occur in minimum number of user sessions). Most of these functions require an object of type '`transactions`' as input. A '`Clickstreams`' object can be transformed to a '`transactions`' object by calling:

```
R> trans <- as.transactions(cls)
```

Extracting all click patterns with a particular minimum support is then possible with the Apriori (Agrawal, Imielinski, and Swami 1993), the eclat (Zaki, Parthasarathy, Ogihara, and Li 1997) or the cSPADE algorithm (Zaki 2001). The following code returns all pattern sequences having a minimum support of 0.4.

```
R> library("arulesSequences")
R> sequences <- as(cspade(trans, parameter = list(support = 0.4)),
+    "data.frame")
R> sequences
```

The corresponding output shows that 11 pattern sequences are supported by at least 40% of the clickstreams in `cls`.

```
              sequence   support
1            <{Defer}> 0.6666667
2               <{P1}> 0.8888889
3               <{P3}> 0.5555556
```

[2]Note that the number of sessions is typically much larger than the number of states in real data sets.

```
4              <{P4}> 0.5555556
5          <{P1},{P3}> 0.5555556
6          <{P4},{P1}> 0.4444444
7       <{P1},{Defer}> 0.6666667
8       <{P3},{Defer}> 0.5555556
9       <{P4},{Defer}> 0.5555556
10 <{P4},{P1},{Defer}> 0.4444444
11 <{P1},{P3},{Defer}> 0.5555556
```

Predicting the next click for a given pattern sequence S is possible by searching for the pattern sequence with the highest support that starts with S. **arules** provides the function `support` with which the support for a given set of pattern sequences is calculated.

6. Conclusion and outlook

I introduced a new R package for analyzing clickstreams with Markov chains. The package provides methods and functions for reading and writing lists of clickstreams, fitting clickstreams to Markov chains, clustering clickstreams and predicting the next click(s) of a given user. **clickstream** supports researchers as well as online store providers in getting insights into online consumer behavior as well as possible usability flaws (e.g., identify click patterns that always end with choice deferral) in their online store. **clickstream** furthermore provides functionality to convert lists of clickstreams into other formats such as an incidence data frame or a '`transactions`' object and thus makes the application of other analysis techniques such as regressions or sequential pattern mining as easy as possible.

Although tailored to improve clickstream analysis, data analysts might also benefit from **clickstream** when modeling consumers visiting behavior in offline stores, patient routing in hospital emergency rooms, demand based on a finite set of sales categories for products or any other categorical data sequences. A click can be represented as any state of a categorical variable. Clickstreams in this general sense are interpretable as a sequence of states of a categorical variable over time.

The **clickstream** package is subject to some limitations that provide an avenue for future extensions. Web servers typically log clicks with a time stamp. The duration between two subsequent clicks is capable of being integrated as additional information to more accurately predict the next click of a user (Montgomery *et al.* 2004). In future it is intended to use time stamps as additional data in clustering clickstreams and predicting the next clicks of a user.

I described the clicks of online users by the product they are considering in a particular online store. A click can, however, also be described by other criteria such as the average customer rating, the product category or the price of the current product. I will thus extend **clickstream** to use multiple criteria to describe the clicks.

The current version of **clickstream** allows to define a prior distribution for the absorbing states when predicting the next click for a user session. In the next version, I will implement the possibility to specify Dirichlet distribution priors on the transitions in a Markov chain.

Acknowledgments

I would like to thank Joachim Schnurbus for some meaningful comments to improve the readability of this article.

References

Agrawal R, Imielinski T, Swami A (1993). "Mining Association Rules between Sets of Items in Large Databases." In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'93)*, pp. 207–216.

Buchta C, Hahsler M (2016). ***arulesSequences**: Mining Frequent Sequences.* R package version 0.2-16, URL https://CRAN.R-project.org/package=arulesSequences.

Ching WK, Huang X, Ng M, Siu TK (2013). *Markov Chains: Models, Algorithms and Applications.* 2nd edition. Springer-Verlag.

Csardi G, Nepusz T (2006). "The **igraph** Software Package for Complex Network Research." *InterJournal*, **Complex Systems**, 1695.

Feinerer I, Hornik K (2008). "Text Mining Infrastructure in R." *Journal of Statistical Software*, **25**(5), 1–54. doi:10.18637/jss.v025.i05.

Hahsler M, Grün B, Hornik K (2005). "**arules** – A Computational Environment for Mining Association Rules and Frequent Item Sets." *Journal of Statistical Software*, **14**(15), 1–25. doi:10.18637/jss.v014.i15.

Huang J, Ng M, Ching WK, Ng J, Cheung D (2001). "A Cube Model and Cluster Analysis for Web Access Sessions." In R Kohavi, B Masand, M Spiliopoulou, J Srivastava (eds.), *WEBKDD 2001, Workshop on Mining Web Log Data across All Customer Touch Points*, pp. 47–58.

Meyer D, Dimitriadou E, Hornik K, Weingessel A, Leisch F (2015). ***e1071**: Misc Functions of the Department of Statistics (E1071), TU Wien.* R package version 1.6-7, URL https://CRAN.R-project.org/package=e1071.

Moe W (2003). "Buying, Searching, or Browsing: Differentiating between Online Shoppers Using In-Store Navigational Clickstream." *Journal of Consumer Psychology*, **13**(1–2), 29–39. doi:10.1207/s15327663jcp13-1&2_03.

Montgomery AL, Li S, Srinivasan K, Liechty JC (2004). "Modeling Online Browsing and Path Analysis Using Clickstream Data." *Marketing Science*, **23**(4), 579–595. doi:10.1287/mksc.1040.0073.

Pei J, Han J, Mortazavi-Asl B, Wang J, Pinto H, Chen Q, Dayal U, Hsu MC (2004). "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach." *IEEE Transactions on Knowledge and Data Engineering*, **16**(11), 1424–1440. doi:10.1109/tkde.2004.77.

Pitman A, Zanker M (2010). "Insights from Applying Sequential Pattern Mining to e-Commerce Clickstream Data." In *IEEE International Conference on Data Mining Workshops (ICDMW)*. doi:10.1109/ICDMW.2010.31.

Raftery A (1985). "A Model for High-Order Markov Chains." *Journal of the Royal Statistical Society B*, **47**(3), 528–539.

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

Spedicato GA, Kang TS, Yalamanchi SB (2016). ***markovchain**: An R Package to Easily Handle Discrete Markov Chains.* R package version 0.6.5.1, URL https://CRAN.R-project.org/package=markovchain.

Zaki M (2001). "SPADE: An Efficient Algorithm for Mining Frequent Sequences." *Machine Learning Journal*, **42**, 31–60. doi:10.1023/a:1007652502315.

Zaki M, Parthasarathy S, Ogihara M, Li W (1997). "New Algorithms for Fast Discovery of Association Rules." *Technical report*, Computer Science Department, University of Rochester, Rochester, NY 14627.

Affiliation:

Michael Scholz
Faculty of Business Administration and Economics
University of Passau
94032 Passau, Germany
E-mail: michael.scholz@uni-passau.de
URL: http://ecommerce.uni-passau.de/

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 4 *Submitted:* 2014-05-30
doi:10.18637/jss.v074.i04 *Accepted:* 2015-08-24