

Rtreemix: a package for estimating mutagenetic trees mixture models and genetic progression scores

Jasmina Bogojeska, Jörg Rahnenführer

April 24, 2008

<http://www.mpi-sb.mpg.de/~jasmina>

1 Introduction

The mixture of mutagenetic trees introduced in [1] is an evolutionary model that provides an interpretable probabilistic framework for modeling multiple paths of ordered accumulation of permanent genetic changes that can be used for describing many disease processes. Each path captures a possible route of disease development. These models are used to model HIV progression characterized by accumulation of resistance mutations in the viral genome under drug pressure [1] and cancer progression by accumulation of chromosomal aberrations in tumor cells [2]. From the mixture model, a genetic progression score (GPS) can be computed for each patient [2] that gives an estimate of the disease progression and can be used for specifying therapies or estimating survival times of the patients. Both the mixture model itself and the derived GPS values are shown to improve the interpretation of disease progression and to have predictive power for estimating the drug resistance in HIV [1] or the survival time in cancer [2]. Beerenwinkel *et al.* in [3] introduced the **Mtreemix** package implemented in C/C++ that provides an efficient code for estimating the mutagenetic trees mixture models from cross-sectional data and using them for various predictions. Building up on the **Mtreemix** software and using the C/C++ API that R provides, we created the **Rtreemix** package. By reusing already existing C functions our package provides the users with R functions as efficient as the programs available in the **Mtreemix**. Similar to **Mtreemix**, the **Rtreemix** package provides functions for learning the mixture model from given data, simulation, likelihood computation and estimation of the GPS values. Furthermore, it introduces new functionality for estimating genetic progression scores with corresponding bootstrap confidence intervals and for performing stability analysis of the mixture models [4].

2 Structure and Functionality

The class structure of the **Rtreemix** package is given in Figure 1. In this way, all data structures and information connected to some entity, like the mixture model or the data for the model estimation, are packed together, which makes the code compact and easy to understand and work with. In what follows we will present a working scenario in which we will use and briefly discuss most of the functions available in **Rtreemix**. More detailed information about the parameters of all the functions used in the text below and their default values can be found in the help files of the package. First, we load the package.

```
> library(Rtreemix)
```

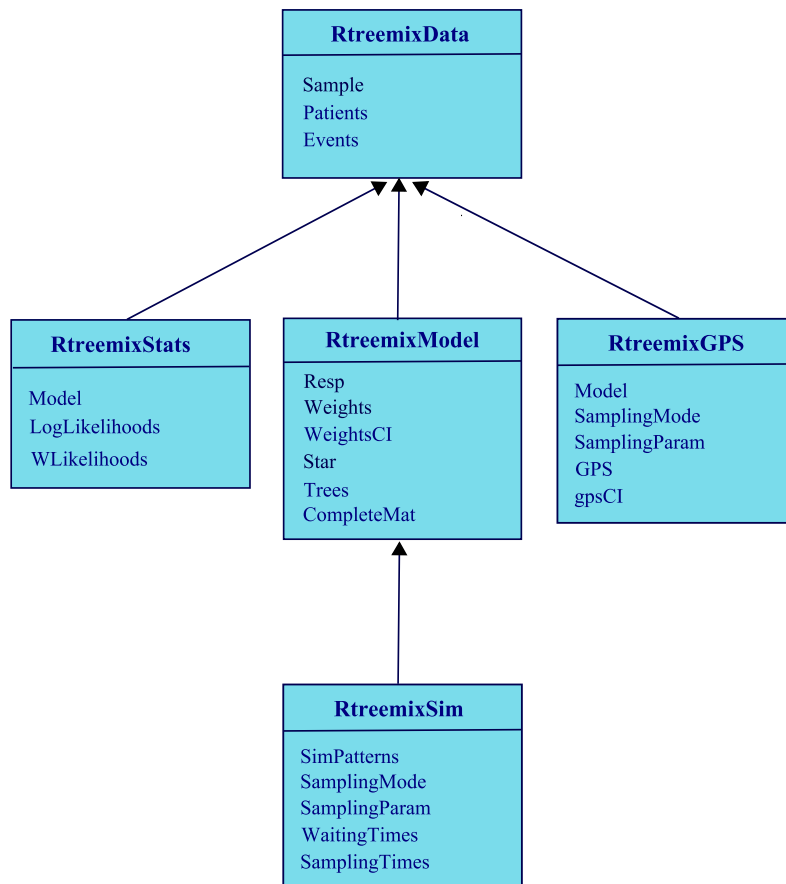


Figure 1: Class diagram of the package *Rtreemix*. The diagram illustrates the classes with their attributes and the relationships among them.

2.1 The Dataset and the RtreemixData class

The datasets used for estimating the mixture models consist of binary patterns that describe the occurrence of a set of genetic events in a set of patients. Each pattern corresponds to a single patient. The set of genetic events comprises genetic changes relevant for the disease taken into consideration. In *Rtreemix* the data used for estimating a mutagenetic trees mixture model is represented as an object of the class **RtreemixData**. We consider the dataset from the Stanford HIV Drug Resistance Database [5] that comprises genetic measurements of 364 HIV patients treated only with the drug *zidovudine*. This dataset is given as an **RtreemixData** object *hiv.data.RData* in the *data* folder of the package and can be loaded and displayed as follows.

```

> data(hiv.data)
> show(hiv.data)

```

It should be also pointed out that a text file with a specific format can be used for creating an object of class **RtreemixData**. An example of such file is the file *treemix.pat* in the

examples directory of the package. The text files used to create an `RtreemixData` object should follow the format of this file. Using *treemix.pat* the `RtreemixData` object is created as follows.

```
> ex.data <- new("RtreemixData", File = paste(.path.package(package = "Rtreemix"),
+      "/examples/treemix.pat", sep = ""))
> show(ex.data)
```

One can also create an `RtreemixData` object by specifying the set of patient profiles as a binary matrix as shown in the code below.

```
> bin.mat <- cbind(c(1, 0, 0, 1, 1), c(0, 1, 0, 0, 1),
+   c(1, 1, 0, 1, 0))
> toy.data <- new("RtreemixData", Sample = bin.mat)
> show(toy.data)
```

Additionally, there are functions for listing the set of profiles, the genetic events, the patient IDs, the number of events and the number of patients in the dataset.

```
> Sample(hiv.data)
> Events(hiv.data)
> Patients(hiv.data)
> eventsNum(hiv.data)
> sampleSize(hiv.data)
```

2.2 Learning mutagenetic trees mixture models

Having a set of patterns that indicate the occurrence of genetic events for a group of patients we can learn a mutagenetic trees mixture model. The model is an object of the `Rtreemix-Model` class that extends the `RtreemixData` class. We fit a 2-trees mixture model for the HIV data [5].

```
> mod <- fit(data = hiv.data, K = 2, equal.edgeweights = TRUE,
+   noise = TRUE)
> show(mod)
```

The tree components of the fitted model can be visualized as follows. When the mixture model contains a large number of tree components it is convenient to be able to plot a specific tree component. The following code plots the second tree component of the mixture model learned from the HIV dataset.

```
> dev.off()
> plot(mod, k = 2, fontSize = 14)
```

It is assumed that mixture models with at least two components always have the noise (star) component as a first component. When only one tree component is fitted to the given data it can be either a star or a nontrivial tree component based on the choice of the parameter `noise`. The mixture components comprising the model are represented as a list of directed

```
$T1
```

```
[1] "A graph with 7 nodes."
```

```
$T2
```

```
[1] "A graph with 7 nodes."
```

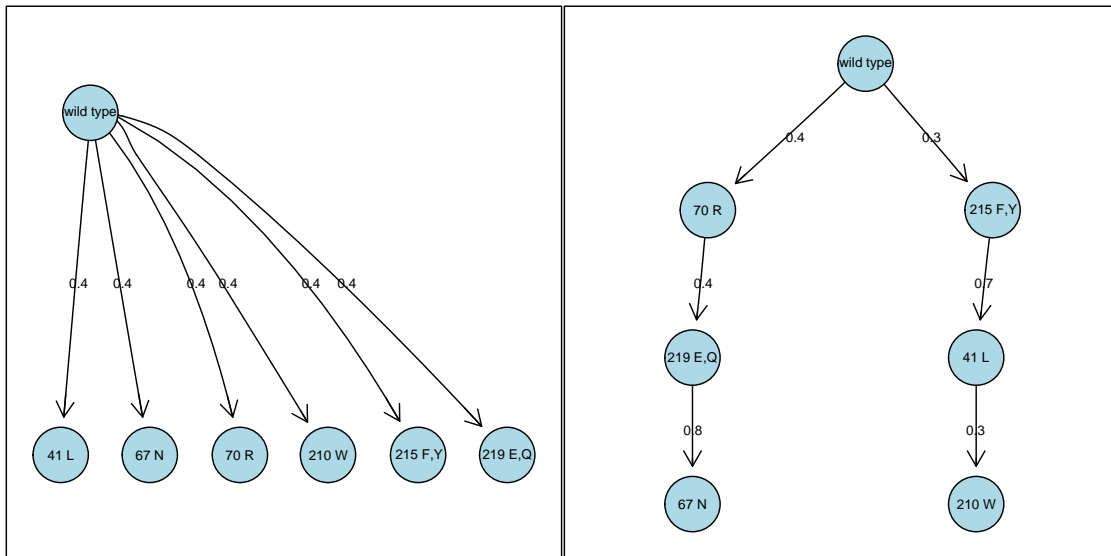


Figure 2: *The mutagenetic trees mixture model for the HIV dataset.*

graphNEL objects, and their weights (the mixture parameters) are given as a numeric vector. There are functions for getting the mixture parameters of the model, the number of tree components, the dataset used for estimating the model, etc.

```
> Weights(mod)
> Trees(mod)
> getTree(object = mod, k = 2)
> edgeData(getTree(object = mod, k = 2), attr = "weight")
> numTrees(mod)
> getData(mod)
```

This class can also contain other useful information connected with the mixture model: an indicator for the presence of the star component, a matrix of the responsibilities of each tree component for each pattern of the data used for learning the model, a matrix of the complete dataset in case of missing data, etc.

```
> Star(mod)
> Resp(mod)
> CompleteMat(mod)
```

The mutagenetic trees mixture model encodes a probability distribution on the set of all possible patterns [1].

```
> distr <- distribution(model = mod)
> distr$probability
```

One can also generate a random mutagenetic mixture model. In this case each tree component from the model is drawn uniformly at random from the tree topology space by using the Prüfer encoding of trees. The number of tree components and the number of genetic events have to be specified. Additionally, one can specify the range from which the edge weights of the tree components are randomly drawn ([0.2, 0.8] in the example bellow).

```
> rand.mod <- generate(K = 3, no.events = 9, noise.tree = TRUE,  
+   prob = c(0.2, 0.8))  
> show(rand.mod)
```

It is also possible to fit a mixture model and analyze its variance by deriving confidence intervals for the mixture parameters and the edge weights (resulting from a bootstrap analysis). An example for this is given in the PDF file *ExtendedVignette* in the *doc* folder of the package.

2.3 The likelihood method

The package `Rtreemix` implements the function `likelihoods` which calculates the (log, weighted) likelihoods for the patterns in a given dataset (`RtreemixData` object) derived with respect to a given `RtreemixModel`. The likelihoods are contained in an object of class `RtreemixStats` which extends the `RtreemixData` class. The number of the genetic events in the patterns from the given dataset has to be equal to the number of genetic events in the branchings from the given mixture model. In the code that follows we calculate the likelihoods of the HIV dataset with respect to the fitted mixture model.

```
> mod.stat <- likelihoods(model = mod, data = hiv.data)  
> Model(mod.stat)  
> getData(mod.stat)  
> LogLikelihoods(mod.stat)  
> WLikelihoods(mod.stat)
```

When having the weighted likelihoods, one can easily derive the responsibilities of the model components for generating the patterns in the specified dataset.

```
> getResp(mod.stat)
```

2.4 The sim method

The mutagenetic trees mixture model encodes a probability distribution on the set of all possible patterns for a specified set of genetic events. The `sim` method provides the possibility of simulating (drawing) patterns from a given `RtreemixModel`. The simulated patterns are then returned as an `RtreemixData` object. Let's draw a specified number of patterns from our randomly generated model.

```
> data <- sim(model = rand.mod, no.draws = 300)  
> show(data)
```

When besides the mixture model also the sampling mode and its respective sampling parameter are specified, this function simulates patterns together with their waiting and sampling times from the respective model. The waiting and sampling times result from a waiting time simulation along the branchings of the mixture model. The `sim` method presents the results in an `RtreemixSim` object. The example given in the text below uses the randomly generated model and demonstrates the methods available in the class `RtreemixSim`.

```
> sim.data <- sim(model = rand.mod, sampling.mode = "exponential",
+   sampling.param = 1, no.sim = 300)
> SimPatterns(sim.data)
> SamplingMode(sim.data)
> SamplingParam(sim.data)
> WaitingTimes(sim.data)
> SamplingTimes(sim.data)
> getModel(sim.data)
> noDraws(sim.data)
```

2.5 The genetic progression score (GPS)

When assuming independent Poisson processes for the occurrence of events on the edges of each mixture component and for the sampling time of the disease, waiting times can be mapped on the tree edges of the branchings. Consequently, a genetic progression score (GPS) that incorporates correlations among events and time intervals among occurrences of events can be associated to the mixture model as proposed in [2]. The GPS estimates the stage of the disease and is important for estimating survival times and giving patients proper therapies. In the `Rtreemix` package the method `gps` calculates the GPS of a given set of patterns with respect to a specified mixture model (an `RtreemixModel` object). The GPS values are derived in a waiting time simulation for a given large number of simulation iterations, and for a specified sampling mode ("constant" or "exponential") and its corresponding sampling parameter. The result of the GPS calculation is given as an object of the class `RtreemixGPS` which extends the class `RtreemixData`. We will demonstrate this process by using the HIV dataset as an example.

```
> modGPS <- gps(model = mod, data = hiv.data, no.sim = 10000,
+   sampling.mode = "exponential", sampling.param = 1)
> Model(modGPS)
> SamplingMode(modGPS)
> SamplingParam(modGPS)
> GPS(modGPS)
> getData(modGPS)
```

When the set of profiles is not specified the GPS is calculated for the set of all possible patterns as shown in the example below.

```
> modGPS.all <- gps(model = mod, no.sim = 10000, sampling.mode = "exponential",
+   sampling.param = 1)
> show(modGPS.all)
> getData(modGPS.all)
```

With the function `confIntGPS` the package gives the possibility to analyze the variance of the GPS values. This function first calculates the GPS for the patterns in a given dataset based on a fitted K-mutagenetic trees mixture model. Then, it derives a 95% confidence intervals for the GPS values with bootstrap analysis. The data and the number of tree components K have to be specified. The confidence intervals reflect the variability of the estimated GPS values. The results are again given as an `RtreemixGPS` object. In the code bellow we derive confidence intervals for the GPS values of the patterns in the HIV dataset.

```
> conf.GPS <- confIntGPS(data = hiv.data, K = 2, B = 1000)
> show(conf.GPS)
> GPS(conf.GPS)
> gpsCI(conf.GPS)
```

3 Stability analysis with the Rtreemix package

The `Rtreemix` package implements functions that can be used for assessing the quality and analyzing the stability of different attributes of the mutagenetic trees mixture model (the probability distributions induced by the model, the number of tree components, the topologies of the tree components, the GPS values and so on). This is usually done by first choosing a model attribute of interest and its appropriate similarity measure, and then inspecting its stability in a simulation setting. In the simulation study typically attributes from simulated true mixture models are compared with the corresponding attributes from models fitted to observations drawn from these true models. The similarity between the true and the fitted model is then compared to the similarities between the true and a sufficient number of random models sampled uniformly from the mixture model space. The quality of a fitted attribute is then assessed by estimating a p-value as the percentage of cases in which the true model is closer to a random model than to the fitted model with respect to the chosen model attribute. Different similarity measures are used for different model attributes. For this purpose the package realizes the Euclidian distance, the Cosine distance, the L_1 distance, the rank-correlation distance and the Kullback-Lebler divergence. The following example code demonstrates the usage of the similarity measures for two numeric vectors `x` and `y` with equal lengths.

```
> x <- c(1, 2, 3, 4, 5)
> y <- c(5, 6, 7, 8, 9)
> L1.dist(x, y)
> euclidian.dist(x, y)
> cosin.dist(x, y)
> rank.cor.dist(x, y)
```

In order to be able to compare the tree topologies of two different mixture models we developed and implemented two similarity measures. The method `comp.models` implements the similarity of the tree topologies of two models based on the sum of the number of different edges of matched tree components (similarity pairs). Since the topology of the components of the mixture models should capture the order in which the genetic events appear during disease progression, we developed another similarity measure implemented with the method `comp.models.levels`. Herein besides adding each edge difference of a similarity pair we also add the L_1 distance of the level vectors of the trees creating the similarity pair. A

level vector assigns to each node the level at which that node is in a specific tree and therefore reflects the order of the genetic events in the tree. The two mixture models that are compared should have the same number of tree components. The code bellow gives an example of using the two previously described similarity measures.

```
> rand.mod1 <- generate(K = 3, no.events = 9, noise.tree = TRUE,  
+   prob = c(0.2, 0.8))  
> rand.mod2 <- generate(K = 3, no.events = 9, noise.tree = TRUE,  
+   prob = c(0.2, 0.8))  
> comp.models(rand.mod1, rand.mod2)  
> comp.models.levels(rand.mod1, rand.mod2)
```

When a mixture model has more than two nontrivial tree components one can quantify their diversity by adapting the two previously described similarity measures. All possible pairs of nontrivial components are considered when computing the similarity. In the `Rtreemix` package the methods `comp.trees` and `comp.trees.levels` implement these modified versions of the similarity measures in `comp.models` and `comp.models.levels`, respectively. An example of their usage for the randomly generated model `rand.mod` is given as follows.

```
> comp.trees(rand.mod)  
> comp.trees.levels(rand.mod)
```

With the functionality described so far one can analyze the stability of the mutagenetic trees mixture model in a simulation setting. As an example, the package includes the function `stability.sim` that can estimate the stability of the mixture model on three different levels: the GPS values, the encoded probability distribution, and the tree topologies. This function outputs a list of different values that summarize the results of the stability analysis - the values of different similarity measures for the different attributes with their corresponding p-values. As an example, Figure 3 depicts a boxplot that represents the results for the stability analysis of the fitted tree topologies for various sample sizes used for learning the mixture models. More details about performing stability analysis of the tree mixture models and interpreting the results can be found in [4].

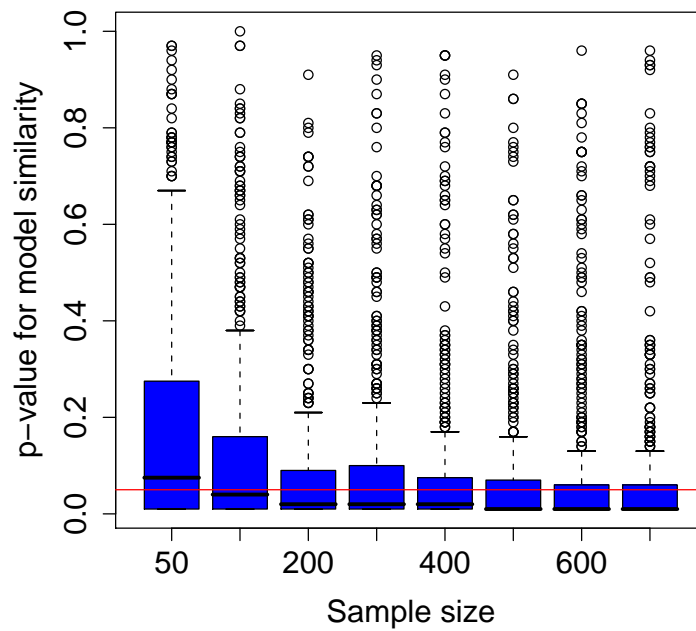


Figure 3: Significance of the similarity between the tree topologies of the true and the corresponding fitted mixture models for various sample sizes. The mixture models have three components ($K = 3$). The red line depicts the threshold p -value = 0.05.

References

- [1] N Beerenwinkel, J Rahnenführer, M Däumer, D Hoffmann, R Kaiser, J Selbig, and T Lengauer. Learning multiple evolutionary pathways from cross-sectional data. *Journal of Computational Biology*, 12(6):584–598, 2005.
- [2] J Rahnenführer, N Beerenwinkel, W A Schulz, C Hartmann, A V Deimling, B Wullich, and T Lengauer. Estimating cancer survival and clinical outcome based on genetic tumor progression scores. *Bioinformatics*, 21(10):2438–2446, 2005.
- [3] N Beerenwinkel, J Rahnenführer, R Kaiser, D Hoffmann, J Selbig, and T Lengauer. Mtreemix: a software package for learning and using mixture models of mutagenetic trees. *Bioinformatics*, 21(9):2106–2107, May 2005.
- [4] J Bogojeska, J Rahnenführer, and T Lengauer. Stability analysis of mixtures of mutagenetic trees. *BMC Bioinformatics*, 9(1), 2008.
- [5] S Rhee, M J Gonzales, R Kantor, B J Betts, J Ravela, and R W Shafer. Human immunodeficiency virus reverse transcriptase and protease sequence database. *Nucleic Acids Res*, 31(1):298–303, 2003.