An introduction to TipDatingBeast v1.0-6

Adrien Rieux[1] and Camilo E. Khatchikian[2]

[1] CIRAD, UMR PVBMT, La Réunion, France

[2] Department of Biological Sciences, The University of Texas at El Paso, USA

November 20, 2017

Abstract

This vignette provides an introductory tutorial to the TipDatingBeast package (Rieux & Khatchikian submitted) for the R software (R Development Core Team 2016). This package aims to assist BEAST users implementing various phylogenetic tip-dating tests to assess the genetic and temporal signal of a dataset. This tutorial provides practical guidance to use the R functions implemented in this package and interpret the results.

Please cite as

Rieux, A. and Khatchikian, C.E., 2017. TipDatingBeast: An R package to assist the implementation of phylogenetic tip-dating tests using BEAST. *Molecular ecology resources*, **17**(4), pp.608-613.

TipDatingBeast@gmail.com

Contents

1) Introduction

Tip dating of phylogenetic trees is a growing discipline aiming to make use of sequence data isolated at different points in time (i.e., heterochronous datasets) to co-estimate the timing of evolutionary events with rates of molecular evolution. We built the R package TipDatingBeast to assist BEAST users with the implementation of various phylogenetic tip-dating tests aiming to assess the genetic and temporal signal present in a dataset. TipDatingBeast currently contains two main functions. The first function allows preparing date-randomization analyses, which assess the temporal signal of a dataset. The second function allows performing leave-one-out analyses, which allows testing for the consistency between independent calibration sequences and pinpointing those leading to potential bias. In this tutorial we provide guidance on how to use those functions and interpret the results.

2) Getting started

2.1 Installing the package

Launch R and install TipDatingBEAST using:

```
>install.packages("TipDatingBeast")
```

Then launch TipDatingBEAST:

```
>library(TipDatingBeast)
```

2.2 Getting help in R

This guide is distributed with the TipDatingBeast package and can be found in the R repository. Helpful information for each function can also specifically be obtained from R. For instance, getting help for the RandomDates function is possible by typing:

```
>?RandomDates
```

2.3 Bug report and feature requests

Free software evolves through interactions between communities of developers and users. This is especially true in R, where these two communities are very much intermingled. You can use the following dedicated email address: TipDatingBeast@gmail.com to:

• ask a specific question on how to use this package (please carefully check if you cannot find your answer in this tutorial before contacting us!)

• ask for a new feature: something useful is missing, and you think it will be useful to others? Please let us know!

• report a possible bug.

3) Using TipDatingBeast functions

3.1 Beast 1.x vs Beast 2.x

The software BEAST (BEAST 1.x) (Drummond & Rambaut 2007) has recently undergone a rewrite, leading to the release of a new version (BEAST 2) (Bouckaert *et al.* 2014). Because the input files required by BEAST 1.x and 2.x are markedly different and we feel that both versions will be independently updated with new features and models, we made the functions implemented in TipDatingBeast compatible with both versions. The functions currently implemented in the TipDatingBeast package automatically screen the input files and detect whether they have been produced by BEAST 1.x or 2.x.

3.2 The influenza dataset as an example

In this tutorial, we will use the influenza dataset provided in the "Virus Practical.zip" folder of the BEAST tutorials (http://beast.bio.ed.ac.uk/tutorials) as an empirical example, The "Flu.nex" file is an alignment of 21 sequences from the Influenza A virus sampled between 1997 and 2004 that contains 1698 nucleotides in length.

The functions implemented in TipDatingBeast require a BEAST xml-formatted input file in which each sequence is associated to a date (of any unit e.g., days, months, years etc). Such BEAST xml input files are generally generated from a molecular sequence alignment file using the graphical user-interface (GUI) application BEAUti. BEAUti is distributed in the same package as BEAST (downloadable from the BEAST web page). For the purpose of this tutorial, we generated two different xml input files from the "Flu.nex" alignment file: one for BEAST 1.x and one for BEAST 2.x. Those two xml files are included in the TipDatingBeast package that can be downloaded from:

https://cran.r-project.org/web/packages/TipDatingBeast/index.html

3.3 Functions related to the "Date-randomized test"

The Date randomization test (hereafter "DRT") is a critical initial step in any tip-dating analyse as it aims to investigate the strength of the temporal signal within a dataset (see (Duchêne *et al.* 2015) for a recent evaluation of this test's performance). The DRT test involves *i)* generating multiple randomized BEAST xml-formatted input files by permutation of sampling times, *ii)* running them in BEAST and *iii)* comparing the parameter estimates obtained with the original *vs.* the randomized datasets.

The functions "RandomDates" and "RandomClusters" allow generating such date-randomized XML-formatted input files in which tip dates have been shuffled among the taxa.

3.3.1 The "RandomDates" function

The "RandomDates" function reads an XML-formatted input file, randomly shuffles the dates (sequence ages) between all samples and writes a new date-randomized-XML. In TipDatingBeast, this function should be called with the following command:

```
>RandomDates(name,reps,writeTrees)
```

With:

name – between quotes, the name of the original XML-formatted input file on which to apply the date-randomization procedure (the .xml extension should not be included).

reps - the number of repetitions required by the user. There will be as many date-randomized datasets produced as the value of reps (default = 20).

writeTrees – set to False (F) if you do not want the trees to be written when running the date-randomized datasets in BEAST. To make the DRT, only the log files are needed (default = T).

Here is an example with the Influenza dataset for BEAST 1.x:

```
>RandomDates(name="Flu_BEAST_18",reps=20,writeTrees=F)
```

The above command reads the Flu-BEAST-18.xml, shuffles the dates 20 times and output 20 new date-randomized xml's. Note that the name of the xml can be anything and does not have to include the version of BEAST (it is the case here but it is not mandatory). The date-randomized XMLs are automatically labeled: "name".Rep[*i*].xml where *i* is the number of date-randomized datasets.

3.3.2 The "RandomCluster" function

This function is similar to "RandomDates" excepts that in "RandomCluster", samples are grouped into clusters and the shuffling procedure randomizes dates among the clusters but not within (see Duchêne *et al.* 2015) for more details on this procedure). There are two distinct ways to group the samples into clusters. The first one is through the upload of a .csv file containing the names of the samples (as given in the XML) and a cluster number. Any positive integer (i.e., positive number) can be used to identified cluster; if a "0" is given to any sample, it would be excluded from the procedure (refer to Duchêne *et al.* 2015) for a explanation of when such approach could be required). The file containing the classification should be labeled: clusters."name".cvs. An example for such a file in the case of the Influenza dataset can be found distributed with the package. This function allows performing date-randomization amongst ancient samples only, if necessary. To do so, the user needs to set the group identifier for the ancient sample to “1” while assigning a “0” to the modern samples

In a second approach, a model-based clustering classification is automatically performed using the mclust library (Fraley & Raftery 2002). In this case, the option loadCluster should be set to FALSE (loadCluster = F). If this option is chosen, the new classification is written in a csv file that is labeled: clusters."name".cvs.

In TipDatingBeast:

```
>RandomCluster(name,reps,loadCluster,writeTrees)
```

With:

name - between quotes, the name of the original XML-formatted input file on which to apply the date-randomization procedure (the .xml extension should not be included).

reps - the number of repetitions required by the user. There will be as many date-randomized datasets produced as the value of reps (default = 20).

loadCluster - set to True (T) if the user provide the .csv file to define the clusters. In that case, this file should be labeled "clusters.name.csv", where name is the name of the original XML-formatted input file. If set to False (F) the clustering is automatically made using the mclust library (Fraley & Raftery 2002).

writeTrees – set to False (F) if you do not want the trees to be written when running the date-randomized datasets in BEAST. To make the DRT, only the log files are needed (default = T).

Here is an example with the Influenza dataset for BEAST 2.x:

```
>RandomCluster(name="Flu_BEAST_232",reps=20,loadCluster=T,
writeTrees=F)
```

The above command reads the Flu-BEAST-232.xml, shuffles the dates 20 times among but not within the clusters (as defined in Flu-BEAST-232.clusters.csv) and output 20 new date-randomized xmls. Note that the name of the xml can be anything and does not have to include the version of BEAST (it is the case here but it is not mandatory). The date-randomized XMLs are automatically labeled: "name".Rep[i].xml where i is the number of date-randomized datasets.

3.3.3 The "PlotDRT" function

Once analyzed with BEAST, the PlotDRT function allows comparing the parameter estimates obtained with the original *vs.* the randomized datasets. This function requires uploading, reading and analyzing the "Log" files generated by BEAST. A Log file contains a row for each MCMC sampling and a column for each estimated parameter. When considered as frequency distributions, this file provides an estimate of the marginal

posterior probability distribution for each parameter. The function "PlotDRT" enable a graphical comparison of the parameter estimates obtained with the original *vs.* the date-randomized datasets (whatever the shuffling procedure considered).

In TipDatingBeast:

```
>PlotDRT(name,reps,burnin)
```

With:

name - between quotes, the name of the original Log file (excluding the .log extension) to upload and compute the real parameter estimates on. The name of the Log files computed from the date-randomized datasets should look like "name.Rep[i].log".

reps - the number of date-randomized log files to be used.

burnin - the fraction of the first MCMC sampling to exclude from the Log files when computing the parameter estimates distribution (default = 0.1, which means 10%).

An example set of log files for the influence dataset can be found online (doi:10.5061/dryad.43q71), the plot can be made by using:

```
>PlotDRT(name="Flu_BEAST_18",reps=20,burnin=0.1)
```

The following message should appear on the console:

```
enter parameter, type VAR to list variable names, or hit
enter to cancel:
```

This means that you should indicate which parameter of the log files you want the DRT to be performed on. You generally have the choice between using the rate of evolution or the age of the root. To select a variable, type VAR and hit enter:

```
enter parameter, type VAR to list variable names, or hit
enter to cancel: VAR
```

The list of parameter extracted from the log file should appear. Let say you want the DRT to be performed on the rate of evolution (noted clock.rate), then type clock.rate

```
enter parameter, type VAR to list variable name, or hit
enter to cancel: clock.rate
```

R will now extract the clock.rate column from both the real and the date-randomized log files and output *i)* summary statistics (mean, median, 95% HPD) in a stats.csv file as well as *ii)* the 95%HPD (both in natural and log scale) in a figure (see Fig. 1).

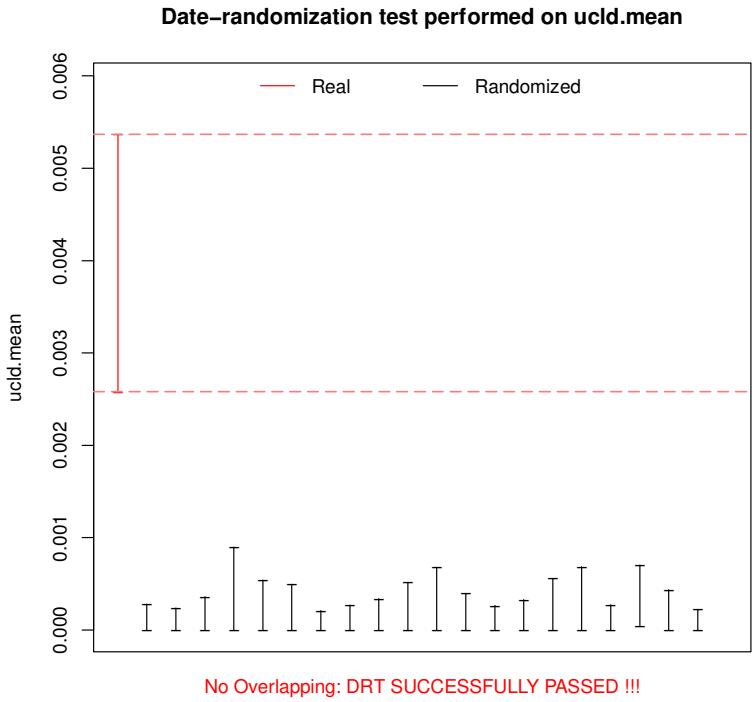

Figure 1: Plot obtained with the PlotDRT function displaying the 95% HPD for the rate of evolution calculated on the real (red) and the date-randomized (black) datasets.

This plot, often called the DRT is fairly straightforward to interpret. If the 95%HPD of the rate of evolution (or of the age of the root) calculated on the real dataset (in red) do not overlap with any of the ones on the date-randomized datasets (in black) as above, then the DRT is passed and tip-dating inferences can thoroughly be performed on the real dataset. If this is not the case, then the temporal signal in the data set is probably not wide enough to

perform tip-dating inferences (see Duchêne *et al.* 2015; Rieux & Balloux 2016; Rieux & Khatchikian 2017) for more information).

3.4 Functions related to the Leave-one-out-cross-validation

The Leave-one-out-cross-validation (hereafter LOOCV) is a statistical procedure aiming to detect whether some particular sequences, when used as calibration in a tip dating analyses, could lead to systematic bias.

For a dataset of *n* taxa, the LOOCV involves *i)* generating *n* XML-formatted input files, each of them excluding one taxa, *ii)* running BEAST with each new input file and estimate the age of the missing taxa using all the other ones and *iii)* comparing the estimated *vs.* true sequence ages.

3.4.1 The "TaxaOut" function

The function "TaxaOut" allow generating as many XML-formatted input files as a dataset contains taxa and estimating. Those new XMLs are modified to, when ran in BEAST, provide an estimation of the age of the missing taxa using the calibrations of the other ones. In R:

```
>TaxaOut(name,lBound,hBound,writeTrees=F)
```

With:

name - between quotes, the name of the original XML-formatted input file on which to apply the LOOCV procedure (the .xml extension should be excluded). This xml should be set up so that earlier dates have lower numerical values (i.e., set direction = "forwards" when setting up date in BEAUti).

lBound - the uniform prior lower bound for the age of the missing taxa (default = 0)
hBound - the uniform prior higher bound for the age of the missing taxa (default = 1.0E100)

writeTrees – set to False (F) if you do not want the trees to be written when running the LOOCV datasets in BEAST. To proceed the LOOCV, only the log files are needed (default = T)

Here is an example with the Influenza dataset for BEAST 1.x:

```
>TaxaOut(name="Flu_BEAST_18",lBound=0.0,
hBound=1.E100,writeTrees=F)
```

The above command reads the Flu-BEAST-18.xml and generates 21 new XMLs, each of them excluding one different taxa from the tip-calibration panel. Note that the name of the xml can be anything and does not have to include the version of BEAST (it is the case here but it is not mandatory). The new XMLs for the LOOCV procedure are automatically labeled: "name".Taxon[i].xml where *i* is the number of the taxon being excluded.

Note: in case bounds are manually set up through the lBound and hBound arguments, ages of the samples should be within (and not on the boundaries of) the range of the age prior used. If not the case, it is possible that BEAST stalls.

3.4.2 The "listTaxa" and "TaxonOut" functions

The function "TaxonOut" does the same thing as "TaxaOut" but one one specific Taxon (instead for doing it sequentially for each one).

```
>TaxonOut(name,lBound,hBound,takeOut,writeTrees)
```

With:

See arguments description for "TaxaOut" above.

takeOut – The name of the taxon (between quotes) or the number indicating the position of the taxon (no quotes) to be excluded from the tip-calibration panel and for which the age

should be estimated using the other taxa. The names and positions of the taxa in an XML can be obtained with the "ListTaxa funtion" (see below).

Here is an example with the Influenza dataset for BEAST 1.x, specifying the name of the taxon (HUMAN_VIETNAM_CL105_2005) to be excluded from the tip-calibration panel and for which the age should be estimated using the other taxa.

```
>TaxonOut(name="Flu_BEAST_18",lBound=0.0,hBound=1.E100,
takeOut= "HUMAN_VIETNAM_CL105_2005",writeTrees=F)
```

The same result is obtained if instead of specifying the name of the taxon, the number indicating the position of the taxon is provided (no quotes).

```
>TaxonOut(name="Flu_BEAST_18",lBound=0.0,hBound=1.E100,
takeOut= 2,writeTrees=F)
```

The two above commands are equal, they read the Flu-BEAST-18.xml and generate one new XML excluding the taxon specified (the second taxon in the file, called “HUMAN_VIETNAM_CL105_2005”) from the tip-calibration panel. Note that the name of the xml can be anything and does not have to include the version of BEAST (it is the case here but it is not mandatory). The new XMLs for the LOOCV procedure are automatically labeled: "name".Taxon[i].xml where *i* is the number of the taxon being excluded.

Note: in case bounds are manually set up through the lBound and hBound arguments, age of the sample to be taken out should be within (and not on the boundaries of) the range of the age prior used. If not the case, it is possible that BEAST stalls.

The "ListTaxa" function lists the Taxa names in a BEAST XML-formatted file.

```
>ListTaxa(name)
```

With:

name - between quotes, the name of the BEAST XML-formatted input file

Here is an example with the Influenza dataset for BEAST 1.x:

```
>ListTaxa("Flu_BEAST_18")
```

The above command reads the Flu-BEAST-18.xml and displays in the console the 21 taxa names with the corresponding position number.

3.4.3 The "PlotLOOCV" function

Once the XMLs produced with the "TaxaOut" function analyzed with BEAST, the PlotLOOCV function allows a graphical comparison of the estimated *vs* "true" sequence ages. Similarly to the PlotDRT function, PlotLOOCV requires uploading, reading and analysing the "Log" files generated by BEAST.

```
>PlotLOOCV(name,burnin)
```

With:

name - between quotes, the name of the original XML file (excluding the .xml extension). In the same folder, should be present the *i* LogFiles generated from the *i* XMLs produced with the "TaxaOut" function (*i* being the number of taxa in the dataset). The name of the Log files should look like "name.Taxon[i].log".

burnin - the fraction of the first MCMC sampling to exclude from the Log files when computing the age estimates distribution (default = 0.1, which equals to 10% burnin).

An example set of log files and original xml necessary for the analysis can be found online (doi:10.5061/dryad.43q71), the plot can be made by using:

```
>PlotLOOCV("Flu_BEAST_18",burnin=0.1)
```

The above command will extract the sample "true" ages from the original XML and the BEAST estimated ones from the Log files. It will output the summary statistics in a "leave.one.out.txt" file and produced the following figure:

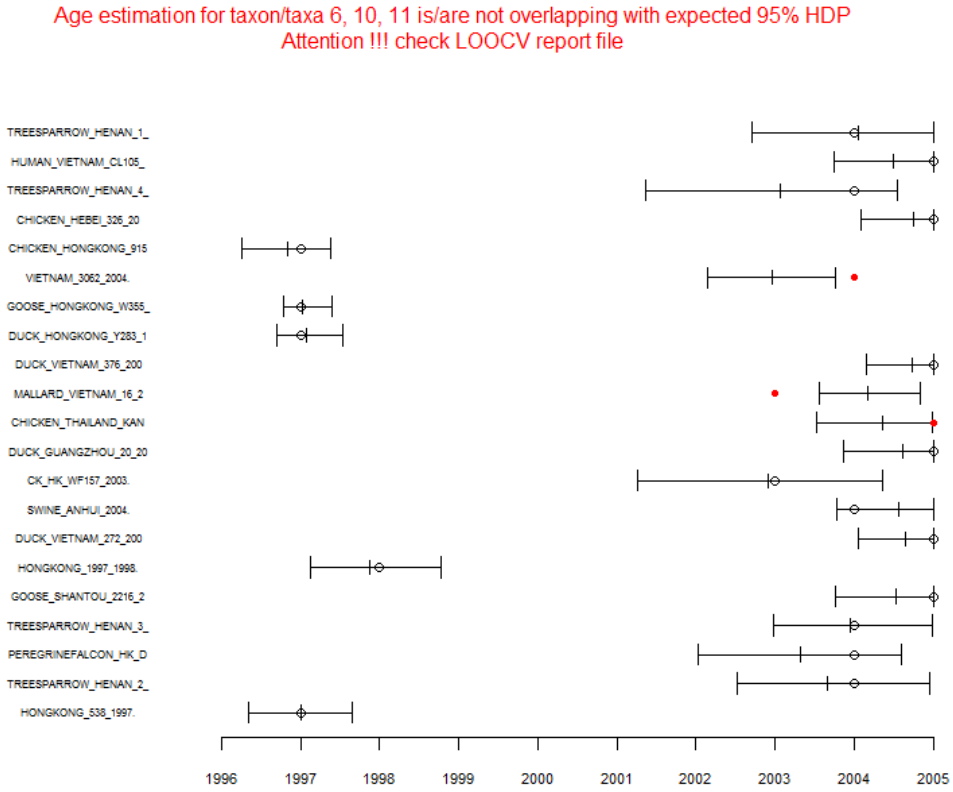

Figure 2: Plot obtained with the PlotLOOCV function. For each sample, horizontal bars represent their estimated ages (95% HPD) and circles their "real" ages (in black when included in the estimated interval, in red otherwise).

This plot is also fairly straightforward to interpret. Particular attention should be paid to the sequences for which "real" ages are not included into the estimated interval (see Rieux & Khatchikian submitted) for a discussion on the reasons and the consequences of such a deviation).

4) Common Issues

4.1 PlotDRT function report "cannot open the connection"

Check that all log files are present in the proper folder (the R working directory) with a .log **and** not a .log.txt extension.

4.2 PlotLOOCV function report "cannot open the connection"

Check that all log files are present in the proper folder (the R working directory) and that the original XML is present in the same folder. Check that the log file has .log **and not a** .log.txt extension. Also, check that the name of this file is the same that the base name of all taxa runs.

5) References

Bouckaert R, Heled J, Kuehnert D*, et al.* (2014) BEAST 2: a software platform for Bayesian evolutionary analysis. *PLoS Computational Biology* **10**.

Drummond AJ, Rambaut A (2007) BEAST: Bayesian evolutionary analysis by sampling trees. *BMC Evolutionary Biology* **7**.

Duchêne S, Duchêne D, Holmes EC, Ho SYW (2015) The performance of the date-randomization test in phylogenetic analyses of time-structured virus data. *Molecular Biology and Evolution,* msv056.

Fraley C, Raftery AE (2002) Model-based clustering, discriminant analysis and density estimation. *Journal of the American Statistical Association* **97**, 611-631.

R Development Core Team (2016) R: A language and environment for stadistical computing. R Foundation for Statistical Computing.

Rieux A, Balloux F (2016) Inferences from tip-calibrated phylogenies: a review and a practical guide. *Molecular Ecology* **25**, 1911-1924.

Rieux A, Khatchikian CE (2017) TipDatingBeast: an R package to assist the implementation of phylogenetic tip-dating tests using BEAST. *Molecular Ecology Resources* **17**, 608-613.