

BADTRIP Manual

Nicola De Maio

January 11, 2018

1 Installing BADTRIP

BADTRIP is a BEAST2 package. To install it, first download and install BEAST2 if you have not already done so (<http://beast2.org/>), the earliest version tested is BEAST 2.4.0 . Then, from the BEAST2 distribution folder, run BEAUti (double-click on the BEAUti icon). In BEAUti, click on "File" from the top bar menu, select "Manage packages" from the scroll-down menu, then select BADTRIP from the table of packages, and click on "Install/Upgrade". Now BEAST2 will run BADTRIP xml files. If BADTRIP is not yet available in the "Manage packages" list, you have to click on the "Package repositories" button. A window pops up where you can click "Add URL" and add

`"https://raw.githubusercontent.com/CompEvol/CBAN/master/packages-extra.xml"`. After you have done this, BADTRIP will be available in the "Manage packages" list.

The source code of BADTRIP, examples, and this documentation, can be also found in <https://bitbucket.org/nicofmay/BADTRIP/>. It is alternatively possible to install BADTRIP from the source code

2 Citing BADTRIP

The BADTRIP manuscript is being peer-reviewed while I write, but is available on bioArxiv: <https://www.biorxiv.org/content/early/2017/11/08/213819> However, you can cite the PoMo model used in BADTRIP:

De Maio, Nicola, Christian Schloetterer, and Carolin Kosiol. Molecular biology and evolution 30.10 (2013): 2249-2262

De Maio, Nicola, Dominik Schrempf, and Carolin Kosiol. Systematic biology 64.6 (2015): 1018-1031

Schrempf, Dominik, et al. Journal of theoretical biology 407 (2016): 362-370

3 Creating a working xml file

A python script is included in this distribution (`/scripts/create_BADTRIP_xml.py`) to create an input xml file for BADTRIP. This script requires three files in input, one with the epidemiological data, one with the sampling time data, and one with the sequencing data. Example data files are given in the `/scripts/` folder, and to run the script you can simply cd into the `/scripts/` folder and use a terminal command like the following:

```
python create_BADTRIP_xml.py -a inputAlignment_example.txt -e
inputEpiData_example.txt -s inputSamples_example.txt -o BADTRIP_xml_example -E
True
```

The output will be named after the -o option, in this case BADTRIP_xml_example.xml .

Instead, to see the script options, simply run
python create_BADTRIP_xml.py -h

If you want to use the create_BADTRIP_xml.py script as explained above, skip the rest of this manual until the section “Running BADTRIP” below. Otherwise, it is also possible to create a working xml input file for BADTRIP by using the given xml template file (/examples/BADTRIP.xml) and manually adding/modifying parts as explained below. In the following, I will assume that the “BADTRIP.xml” example file is used as a template.

3.1 Including the genetic alignment

An example genetic alignment section:

```
<alignmentPoMo spec="beast.evolution.alignment.AlignmentPoMo" id="alignment"
  dataTypePoMo="PoModata">
  <sequence spec="SequencePoMo" id="sS1" taxon="tS1" value="10-0-0-0,
    15-0-0-0, 0-0-8-3"/>
  <sequence spec="SequencePoMo" id="sS2" taxon="tS2" value="20-0-0-0,
    0-23-0-0, 0-0-5-0"/>
  <sequence spec="SequencePoMo" id="sS3" taxon="tS3" value="0-0-0-0, 0-2-0-0,
    0-0-0-1"/>
  <sequence spec="SequencePoMo" id="sS4" taxon="tS4" value="3-0-0-0, 0-5-0-0,
    0-0-5-2"/>
</alignmentPoMo>
```

Data for each site and population is in the form “0-1-0-9”, that is, 4 dash-separated integers. These four integers represent respectively the allele counts of As, Cs, Gs, and Ts at the given position and population. It is important that all taxa have the same number of positions. An element “0-0-0-0” is equivalent to providing no data for the position, ideal in cases when a deletion removes part of the genome, for example.

3.2 Including time and host information

Information regarding hosts (start, “HostDatesStartValue”, and end, “HostDatesEndValue”, times for their exposure intervals) and samples (host from which samples are taken, “samplesHostsValue”, and times of sampling, “samplesDatesValue”) are specified like this:

```
<TraitSetPoMo spec='beast.evolution.tree.TraitSetPoMo' id='traitSet'
  direction="forward" units="day"
  samplesHostsValue="tS1=HH1, tS2=HH1, tS3=HH2, tS4=HH2"
  samplesDatesValue="tS1=5, tS2=10, tS3=7, tS4=8"
  HostDatesStartValue="HH1=-5.0, HH2=2.0, HH3=10.0"
  HostDatesEndValue="HH1=15.0, HH2=9.0, HH3=18.0">
  <taxa spec='beast.evolution.alignment.TaxonSet' alignment='@alignment'/>
</TraitSetPoMo>
```

The numbers here can represent any time unit (days, years, hours, etc) and all other parameters (e.g., mutation rates) have to be interpreted in function of the time scale used.

3.3 Defining the substitution model

The following block defines the substitution model PoMo:

```
<siteModel spec="SiteModelPoMo" id="siteModel">
  <mutationRate spec='RealParameter' id="mutationRate" value="0.05"
    lower="0.0"/>
  <substModel spec="PoMoGeneral" virtualPop="15" estimateRootFreqs="false"
    useTheta="False" theta="0.01" id="PoMo.substModel">
    <fitness spec='RealParameter' id="PoMo.fitness" value="1.0 1.0 1.0 1.0"
      lower="0.8" upper="1.2"/>
    <rootNucFreqs spec='RealParameter' id="PoMo.rootFreqs" value="0.25 0.25
      0.25 0.25"/>
    <mutModel spec='MutationModel' id="PoMo.mutModel">
      <rateVector spec='RealParameter' id="PoMo.mutRates" value="0.01
        0.01" upper="0.3"/>
      <nucFreqs spec='RealParameter' id="PoMo.nucFreqs" value="0.25 0.25
        0.25 0.25"/>
    </mutModel>
  </substModel>
</siteModel>
```

In particular, this line defines many aspects of the model:

```
<substModel spec="PoMoGeneral" virtualPop="15" estimateRootFreqs="false"
  useTheta="False" theta="0.01" id="PoMo.substModel">
```

For example, "virtualPop" determines how many individuals are in the virtual population (here 15). "estimateRootFreqs" determines if root nucleotide frequencies should be estimated or if equilibrium should be assumed. Finally, "useTheta" specifies if a user-specified value of $\theta = 4N_e\mu$ for the root frequency of variant sites should be used, or if it should be estimated. if "useTheta" is set to "true", then θ can be set with:

```
<substModel spec="PoMoGeneral" virtualPop="5" estimateRootFreqs="false"
  useTheta="true" theta="0.016" id="PoMo.substModel">
```

The mutation model has to be one of the following: HKY85, GTR, or general non-reversible. The mutation model is specified in the following line:

```
<rateVector spec='RealParameter' id="PoMo.mutRates" value="0.01
  0.01" upper="0.3"/>
```

Depending on how many values are given after "value", the mutation model is picked. the number of values mean the number of mutation rates in the model, excluding nucleotide frequencies: 1=F81, 2=HKY, 6=GTR, 12=general non-reversible

3.4 Sequencing error

Starting value for sequencing error is specified as:

```
<seqError spec='RealParameter' id="PoMo.seqError" value="0.0001"/>
```

A starting value of "0.0" as shown above, means that no sequencing error is assumed. If the value is different from 0.0, sequencing error is taken into account, and if an operator (see next section) for the sequencing error is specified, the error rate is estimated.

3.5 Operators

Operators determine which parameters of the model are modified, and which are held constant. For example, the following code

```
<!--operator spec="ScaleOperator" id="errScaler"
      parameter="@PoMo.seqError"
      scaleFactor="0.9" weight="1"/-->
```

is commented out (same as if it was not there). By including the same operator,

```
<operator spec="ScaleOperator" id="errScaler"
      parameter="@PoMo.seqError"
      scaleFactor="0.9" weight="1"/>
```

we tell BEAST2 to alter the value, and therefore estimate, the considered parameter, in this case the sequencing error.

Similarly, other operators can be included or removed/commented out, for example:

```
<operator spec="ScaleOperator" id="fitScaler"
      parameter="@PoMo.fitness"
      scaleFactor="0.8" weight="3"/>
```

for nucleotide fitnesses (selection or fixation bias of nucleotides);

```
<operator spec="DeltaExchangeOperator" id="freqExchangerRoot"
      parameter="@PoMo.rootFreqs"
      delta="0.01" weight="0.5"/>
```

for root nucleotide frequencies (only to be used if we are estimating root nucleotide frequencies).

In the case of a nonreversible model, the following operator has to be removed:

```
<operator spec="DeltaExchangeOperator" id="freqExchanger"
      parameter="@PoMo.nucFreqs"
      delta="0.01" weight="0.5"/>
```

3.6 Specifying location and name of output files

To specify the file names and location of the output, simply modify the two following lines at the end of the xml file

```
<logger logEvery="1000" fileName="pathToFile/fileName.log">
```

and

```
<logger logEvery="10000" fileName="pathToFile/fileName.trees" mode="tree">
```

for the log file and the output trees file respectively.

4 Running BADTRIP

Just select the BADTRIP xml file you created, after you double-clicked on the BEAST2 icon in the BEAST2 folder.

Alternatively, you can download the most recent version of the BADTRIP package as a zip file from <https://bitbucket.org/nicofmay/BADTRIP/src/master/dist/>. After installing BEAST2, and after unzipping BADTRIP, and locating the jar file in the lib folder, you can run the following command from terminal:

```
java -cp /path_to_BADTRIP/BADTRIP.v0.1.0.jar:/path_to_BEAST/BEAST
2.4.0/lib/beast.jar beast.app.beastapp.BeastMain /path_to_xml_file/file.xml
```

If you get error, trying uninstalling PoMo from the BEAUti package manager.

4.1 Analysing the output

The output trees from BADTRIP can be used to get a summary of the inferred transmission events, graph-like plots of hosts and transmissions. The trees themselves cannot be visualized in FigTree because they are not bifurcating. To run these analysis, simply use the python script "Make_transmission_tree.py" which can be found within the BADTRIP package or at <https://bitbucket.org/nicofmay/BADTRIP/src/> → BADTRIP. For a description of the formats and options, simply run from terminal "python Make_transmission_tree.py -h" from the BADTRIP/scripts/ folder. To run it, use the command format "python Make_transmission_tree.py -i fileName.trees -o fileNameOutput". Sometimes the graphical output needs to be tuned, in which case you can run *python Make_transmission_tree.py -h* to see the possible options. Also, this script requires installation of *graph_tool*, which is a pain to install sometimes. If you are having problems with installing graph-tool (a dependency of the above script), you can run instead "Make_transmission_tree_alternative.py", a less fancy, but quicker and easier to use version of the visualization script. In case of any additional problem you can contact me: nicola.demaio@ndm.ox.ac.uk