XML Hacking Tutorial

XML hacking

Nicola F. Müller

1 Background

Setting up BEAST2 xmls in BEAUti enables most of the more basic analyses in BEAST2. To get access to either more complex analyses, or even to simply change individual aspects of an analyses, such as the length of a chain, or a prior, it is often easier to directly change parts of the BEAST2 xml.

2 Programs used in this Exercise

2.0.1 BEAST2 - Bayesian Evolutionary Analysis Sampling Trees 2

BEAST2 (http://www.beast2.org) is a free software package for Bayesian evolutionary analysis of molecular sequences using MCMC and strictly oriented toward inference using rooted, time-measured phylogenetic trees. This tutorial is written for BEAST v2.6 (**BEAST2book2014**).

2.0.2 BEAUti2 - Bayesian Evolutionary Analysis Utility

BEAUti2 is a graphical user interface tool for generating BEAST2 XML configuration files.

Both BEAST2 and BEAUti2 are Java programs, which means that the exact same code runs on all platforms. For us it simply means that the interface will be the same on all platforms. The screenshots used in this tutorial are taken on a Mac OS X computer; however, both programs will have the same layout and functionality on both Windows and Linux. BEAUti2 is provided as a part of the BEAST2 package so you do not need to install it separately.

3 Practical: XML Hacking

In this tutorial, we will learn the basic structure of a BEAST2 xml and how to change it. The tutorial uses H1N1 data from the HA and NA segment. The xml provided considers the segments to have their own tree, tree prior, and evolutionary parameters. Further, there are some issues with the settings of the xml that we will have to fix.

The aim is to:

- Learn how to individual building blocks of a beast xml
- Change the MCMC settings directly in the xml
- get familiar with linking aspects of an analysis

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?><beast beautitemplate='Standard' beautistatus='' namespace

    <data
id="h1n1pdm_HA"
spec="Alignment"
name="alignment">
        <sequence id="seq_A/Alaska/07/2017|2017-02-13" spec="Sequence" taxon="A/Alaska/07/2017|2017-02-13" total
        <sequence id="seq_A/Arizona/32/2015|2015-12-29" spec="Sequence" taxon="A/Arizona/32/2015|2015-12-29" tot
        <sequence id="seq_A/Arizona/33/2017|2017-05-07" spec="Sequence" taxon="A/Arizona/33/2017|2017-05-07" tot
        <sequence id="seq_A/Bangkok/INS3_681/2012|2012-09-21" spec="Sequence" taxon="A/Bangkok/INS3_681/2012|201
        <sequence id="seq_A/California/19/2017|2017-01-29" spec="Sequence" taxon="A/California/19/2017|2017-01-2
        <sequence id="seq_A/California/45/2016|2016-02-20" spec="Sequence" taxon="A/California/45/2016|2016-02-2
        <sequence id="seq_A/Durham/INS3_648/2012|2012-03-08" spec="Sequence" taxon="A/Durham/INS3_648/2012|2012-
        <sequence id="seq_A/Finland/75/2014|2014-02-10" spec="Sequence" taxon="A/Finland/75/2014|2014-02-10" tot
        <sequence id="seq_A/Finland/87/2014|2014-02-14" spec="Sequence" taxon="A/Finland/87/2014|2014-02-14" tot
        <sequence id="seq_A/Florida/100/2015|2015-12-20" spec="Sequence" taxon="A/Florida/100/2015|2015-12-20" t
        <sequence id="seq_A/Hawaii/67/2014|2014-10-12" spec="Sequence" taxon="A/Hawaii/67/2014|2014-10-12" total
        <sequence id="seq_A/Illinois/26/2017|2017-03-16" spec="Sequence" taxon="A/Illinois/26/2017|2017-03-16" t
        <sequence id="seq_A/Kansas/14/2016|2016-03-24" spec="Sequence" taxon="A/Kansas/14/2016|2016-03-24" total
        <sequence id="seq_A/Nepal/VIROAF5/2012|2012-08-26" spec="Sequence" taxon="A/Nepal/VIROAF5/2012|2012-08-2
        <sequence id="seq_A/New_Mexico/19/2016|2016-02-28" spec="Sequence" taxon="A/New_Mexico/19/2016|2016-02-2
        <sequence id="seq_A/New_York/WC_LVD_14_021/2014|2014-01-30" spec="Sequence" taxon="A/New_York/WC_LVD_14_
        <sequence id="seq_A/New_York/WC_LVD_14_063/2014|2014-02-09" spec="Sequence" taxon="A/New_York/WC_LVD_14_
        <sequence id="seq_A/Nicaragua/6322_06/2015|2015-12-08" spec="Sequence" taxon="A/Nicaragua/6322_06/2015|2
        <sequence id="seq_A/Nizhnii_Novgorod/CRIE_BLM/2011|2011-01-26" spec="Sequence" taxon="A/Nizhnii_Novgoroc
        <sequence id="seq_A/North_Dakota/15/2017|2017-03-19" spec="Sequence" taxon="A/North_Dakota/15/2017|2017-
        <sequence id="seq_A/Seoul/224/2016|2016-02-03" spec="Sequence" taxon="A/Seoul/224/2016|2016-02-03" total
        <sequence id="seq_A/South_Dakota/13/2017|2017-02-18" spec="Sequence" taxon="A/South_Dakota/13/2017|2017-
        <sequence id="seq_A/Utah/35/2016|2016-04-22" spec="Sequence" taxon="A/Utah/35/2016|2016-04-22" totalcour
        <sequence id="seq_A/Washington/01/2017|2017-01-06" spec="Sequence" taxon="A/Washington/01/2017|2017-01-0
    </data>

    <data
```

Figure 1: A first look at the xml.

3.1 open the xml

First, we have to open the xml in a plain text editor that can highlight the xml synthax.

After opening the xml, we can scroll down to have a look at the settings of the analyses. The length of the MCMC chain can be found on the following line.

```
<run id="mcmc" spec="MCMC" chainLength="10">
```

At the moment, the analyses will only run for 10 iteartions, which is nowhere near enough for this analysis, so let's go ahead and up it to 10000000. Let's scroll to the end of the file where it says:

```
<logger id="tracelog" spec="Logger" fileName="$(filebase).log" logEvery="1000" model="↩
    @posterior" sanitiseHeaders="true" sort="smart">
<log idref="posterior"/>
<log idref="likelihood"/>
<log idref="prior"/>
```

```
  <log idref="treeLikelihood.h1n1pdm_HA"/>
  <log id="TreeHeight.t:h1n1pdm_HA" spec="beast.base.evolution.tree.TreeStatLogger" tree="@Tree.↩
      t:h1n1pdm_HA"/>
  <log idref="treeLikelihood.h1n1pdm_NA"/>
  <log id="TreeHeight.t:h1n1pdm_NA" spec="beast.base.evolution.tree.TreeStatLogger" tree="@Tree.↩
      t:h1n1pdm_NA"/>
  <log idref="clockRate.c:h1n1pdm_NA"/>
  <log idref="clockRate.c:h1n1pdm_HA"/>
  <log idref="kappa.s:h1n1pdm_HA"/>
  <log idref="freqParameter.s:h1n1pdm_HA"/>
  <log idref="freqParameter.s:h1n1pdm_NA"/>
  <log idref="kappa.s:h1n1pdm_NA"/>
  <log idref="popSize.t:h1n1pdm_HA"/>
  <log idref="CoalescentConstant.t:h1n1pdm_HA"/>
  <log idref="popSize.t:h1n1pdm_NA"/>
  <log idref="CoalescentConstant.t:h1n1pdm_NA"/>
</logger>
<logger id="screenlog" spec="Logger" logEvery="1000">
  <log idref="posterior"/>
  <log idref="likelihood"/>
  <log idref="prior"/>
</logger>
<logger id="treelog.t:h1n1pdm_HA" spec="Logger" fileName="$(filebase)-$(tree).trees" logEvery="↩
    10000000" mode="tree">
  <log id="TreeWithMetaDataLogger.t:h1n1pdm_HA" spec="beast.base.evolution.TreeWithMetaDataLogger↩
      " tree="@Tree.t:h1n1pdm_HA"/>
</logger>
<logger id="treelog.t:h1n1pdm_NA" spec="Logger" fileName="$(filebase)-$(tree).trees" logEvery="↩
    1000" mode="tree">
  <log id="TreeWithMetaDataLogger.t:h1n1pdm_NA" spec="beast.base.evolution.TreeWithMetaDataLogger↩
      " tree="@Tree.t:h1n1pdm_NA"/>
</logger>
<operatorschedule id="OperatorSchedule" spec="OperatorSchedule"/>
```

This part of the code says that we are logging 3 files, one log files (with all the parameters) and 2 trees files, one for each alignment. It also specifies how often each of these files is logged. For each file, a filename is specified, $\$(filebase)$ means that beast logs to the file with the same name as the xml. $\$(tree)$ uses the id of the tree that is being logged in the filename. To reduce the length of the filename, we can change the filenames, for example to:

```
<logger id="treelog.t:h1n1pdm_HA" spec="Logger" fileName="$(filebase)-HA.trees" logEvery="↩
    10000000" mode="tree">
  <log id="TreeWithMetaDataLogger.t:h1n1pdm_HA" spec="beast.base.evolution.TreeWithMetaDataLogger↩
      " tree="@Tree.t:h1n1pdm_HA"/>
</logger>
<logger id="treelog.t:h1n1pdm_NA" spec="Logger" fileName="$(filebase)-NA.trees" logEvery="1000" ↩
    mode="tree">
  <log id="TreeWithMetaDataLogger.t:h1n1pdm_NA" spec="beast.base.evolution.TreeWithMetaDataLogger↩
      " tree="@Tree.t:h1n1pdm_NA"/>
</logger>
```

Does anything seem odd in how the files will be logged? If so, make the change.

If you want to make sure that none of the changes broke the xml, we can save the changes and start the run in BEAST. If the run starts and logs the correct file names. The run proceeds oddly fast, however. Let's open the log file in Tracer and see if we can spot what is going wrong.

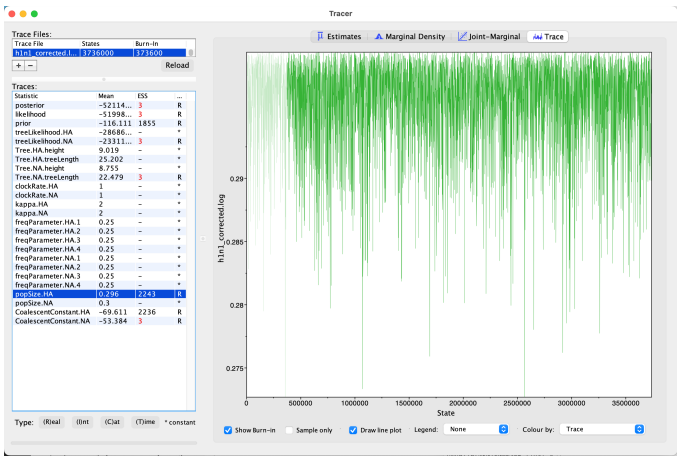

Figure 2: Only one of the $popSize.t : h1n1pdm\_HA$ is being explored.

Since only one of the pop sizes parameters is being explored, something is probably wrong in the settings of the xml. Additionally, the pop size parameter seems to hit some wall at 0.3 where it's blocked. Let's see if we can find out what is going on.

Let's go back to the xml and search for all the places where $popSize.t : h1n1pdm\_HA$ shows up. First, we see it in the state block. Can you spot why the parameter is hitting a wall at 0.3?

```
<parameter id="clockRate.c:h1n1pdm_NA" spec="parameter.RealParameter" lower="0.0" name="stateNode↩
    ">1.0</parameter>
<parameter id="clockRate.c:h1n1pdm_HA" spec="parameter.RealParameter" lower="0.0" name="stateNode↩
    ">1.0</parameter>
<parameter id="kappa.s:h1n1pdm_HA" spec="parameter.RealParameter" lower="0.0" name="stateNode">↩
    2.0</parameter>
<parameter id="freqParameter.s:h1n1pdm_HA" spec="parameter.RealParameter" dimension="4" lower="↩
    0.0" name="stateNode" upper="1.0">0.25</parameter>
<parameter id="freqParameter.s:h1n1pdm_NA" spec="parameter.RealParameter" dimension="4" lower="↩
    0.0" name="stateNode" upper="1.0">0.25</parameter>
<parameter id="kappa.s:h1n1pdm_NA" spec="parameter.RealParameter" lower="0.0" name="stateNode">↩
    2.0</parameter>
<parameter id="popSize.t:h1n1pdm_HA" spec="parameter.RealParameter" lower="0.0" name="stateNode" ↩
    upper="0.3">0.1</parameter>
<parameter id="popSize.t:h1n1pdm_NA" spec="parameter.RealParameter" lower="0.0" name="stateNode">↩
    0.3</parameter>
</state>
```

Let's see where else the parameter shows up. In the operator block of each xml, we specify which MCMC operations can occur. We have one operator acting on the parameter $popSize.t : h1n1pdm\_HA$.

```
<operator id="CoalescentConstantNarrow.t:h1n1pdm_HA" spec="Exchange" tree="@Tree.t:h1n1pdm_HA" ↩
    weight="15.0"/>
<operator id="CoalescentConstantWide.t:h1n1pdm_HA" spec="Exchange" isNarrow="false" tree="@Tree.↩
    t:h1n1pdm_HA" weight="3.0"/>
<operator id="CoalescentConstantWilsonBalding.t:h1n1pdm_HA" spec="WilsonBalding" tree="@Tree.↩
    t:h1n1pdm_HA" weight="3.0"/>
<operator id="PopSizeScaler.t:h1n1pdm_HA" spec="kernel.BactrianScaleOperator" parameter="@popSize↩
    .t:h1n1pdm_HA" upper="10.0" weight="100000000.0"/>
<operator id="CoalescentConstantBICEPSEpochTop.t:h1n1pdm_NA" spec="EpochFlexOperator" scaleFactor↩
    ="0.1" tree="@Tree.t:h1n1pdm_NA" weight="2.0"/>
<operator id="CoalescentConstantBICEPSEpochAll.t:h1n1pdm_NA" spec="EpochFlexOperator" ↩
```

```
    fromOldestTipOnly="false" scaleFactor="0.1" tree="@Tree.t:h1n1pdm_NA" weight="2.0"/>
```

Oddly that operator has weigth 100000000. BEAST will use one operator per iteration in the MCMC. The chance of this operator being called is given by its relative weight. 100000000 means that BEAST will essentially only use this one operator and ignore all others. Let's change it to 3 (the default for this operator) and try running the file again. (Pro tip: it's a lot faster to start a run from the Terminal instead of using the GUI of BEAST).

After a few iterations, we can reload the log file in Tracer and confirm that all the values change over the course of the MCMC.

4 Changing priors

If I just want to change some of my priors in the xml, it's often easier to do that directly in the xml instead of loading things in BEAUti or setting up everything from scratch.

In the xml, there is an entire block that is called prior

```
<distribution id="prior" spec="CompoundDistribution">
<distribution id="CoalescentConstant.t:h1n1pdm_HA" spec="Coalescent">
    <populationModel id="ConstantPopulation.t:h1n1pdm_HA" spec="ConstantPopulation" popSize="↩
        @popSize.t:h1n1pdm_HA"/>
    <treeIntervals id="TreeIntervals.t:h1n1pdm_HA" spec="beast.base.evolution.tree.TreeIntervals"↩
         tree="@Tree.t:h1n1pdm_HA"/>
</distribution>
<distribution id="CoalescentConstant.t:h1n1pdm_NA" spec="Coalescent">
    <populationModel id="ConstantPopulation.t:h1n1pdm_NA" spec="ConstantPopulation" popSize="↩
        @popSize.t:h1n1pdm_NA"/>
    <treeIntervals id="TreeIntervals.t:h1n1pdm_NA" spec="beast.base.evolution.tree.TreeIntervals"↩
         tree="@Tree.t:h1n1pdm_NA"/>
</distribution>
<prior id="ClockPrior.c:h1n1pdm_HA" name="distribution" x="@clockRate.c:h1n1pdm_HA">
    <Uniform id="Uniform.0" name="distr" upper="Infinity"/>
</prior>
<prior id="ClockPrior.c:h1n1pdm_NA" name="distribution" x="@clockRate.c:h1n1pdm_NA">
    <Uniform id="Uniform.3" name="distr" upper="Infinity"/>
</prior>
<prior id="FrequenciesPrior.s:h1n1pdm_HA" name="distribution" x="@freqParameter.s:h1n1pdm_HA">
    <distr id="Dirichlet.0" spec="distribution.Dirichlet">
        <parameter id="RealParameter.4" spec="parameter.RealParameter" dimension="4" estimate="↩
            false" name="alpha">4.0 4.0 4.0 4.0</parameter>
    </distr>
</prior>
<prior id="FrequenciesPrior.s:h1n1pdm_NA" name="distribution" x="@freqParameter.s:h1n1pdm_NA">
    <distr id="Dirichlet.0.h1n1pdm_NA" spec="distribution.Dirichlet">
        <parameter id="RealParameter.4.h1n1pdm_NA" spec="parameter.RealParameter" dimension="4" ↩
            estimate="false" name="alpha">4.0 4.0 4.0 4.0</parameter>
    </distr>
</prior>
<prior id="KappaPrior.s:h1n1pdm_HA" name="distribution" x="@kappa.s:h1n1pdm_HA">
    <LogNormal id="LogNormalDistributionModel.0" name="distr">
        <parameter id="RealParameter.2" spec="parameter.RealParameter" estimate="false" name="M">↩
            1.0</parameter>
        <parameter id="RealParameter.3" spec="parameter.RealParameter" estimate="false" name="S">↩
            1.25</parameter>
    </LogNormal>
</prior>
<prior id="KappaPrior.s:h1n1pdm_NA" name="distribution" x="@kappa.s:h1n1pdm_NA">
    <LogNormal id="LogNormalDistributionModel.0.h1n1pdm_NA" name="distr">
        <parameter id="RealParameter.2.h1n1pdm_NA" spec="parameter.RealParameter" estimate="false↩
```

```
            " name="M">1.0</parameter>
        <parameter id="RealParameter.3.h1n1pdm_NA" spec="parameter.RealParameter" estimate="false↩
            " name="S">1.25</parameter>
    </LogNormal>
</prior>
<prior id="PopSizePrior.t:h1n1pdm_HA" name="distribution" x="@popSize.t:h1n1pdm_HA">
    <OneOnX id="OneOnX.2" name="distr"/>
</prior>
<prior id="PopSizePrior.t:h1n1pdm_NA" name="distribution" x="@popSize.t:h1n1pdm_NA">
    <OneOnX id="OneOnX.3" name="distr"/>
</prior>
</distribution>
```

the x=... gives you which parameter the prior is on. The prior distribution for the parameter $popSize.t : h1n1pdm\_HA$ is currently OneOnX, which is an inverse uniform distribution. Let's assume we have some more information about that prior and we know a lognormal distribution with M="2" and S="1" is a better representation of our prior knowledge of that parameter. To change the prior distribution, we can replace the prior on $popSize.t : h1n1pdm\_HA$ with the below xml part.

```
<prior id="PopSizePrior.t:h1n1pdm_HA" name="distribution" x="@popSize.t:h1n1pdm_HA">
    <LogNormal id="OneOnX.2" name="distr" M="2" S="1"/>
</prior>
```

Note that the id of the distribution is still OneOnX.2. The id of an xml object however doesn't matter with regards to what the object does.

5 Linking parameters

Sometimes, we want to be able to link parameters in different parts of the xml. For example, the same population level process gave rise to the HA and NA segments, so we might want to link the population size parameters of the two segments. We can do that directly in the xml.

At the moment, we have two tree priors, one for each segment. The tree priors are constant population coalescent priors, and there is one for each tree. In the xml, the tree priors are specified as follows:

```
<distribution id="CoalescentConstant.t:h1n1pdm_HA" spec="Coalescent">
<populationModel id="ConstantPopulation.t:h1n1pdm_HA" spec="ConstantPopulation" popSize="@popSize↩
    .t:h1n1pdm_HA"/>
<treeIntervals id="TreeIntervals.t:h1n1pdm_HA" spec="beast.base.evolution.tree.TreeIntervals" ↩
    tree="@Tree.t:h1n1pdm_HA"/>
</distribution>
<distribution id="CoalescentConstant.t:h1n1pdm_NA" spec="Coalescent">
<populationModel id="ConstantPopulation.t:h1n1pdm_NA" spec="ConstantPopulation" popSize="@popSize↩
    .t:h1n1pdm_NA"/>
<treeIntervals id="TreeIntervals.t:h1n1pdm_NA" spec="beast.base.evolution.tree.TreeIntervals" ↩
    tree="@Tree.t:h1n1pdm_NA"/>
</distribution>
```

How do we link the two population size parameters? We can simply replace the population size parameter of one of the tree priors with the parameter of the other tree prior. Let's replace the population size parameter of the NA segment with the population size parameter of the HA segment.

```
  <distribution id="CoalescentConstant.t:h1n1pdm_HA" spec="Coalescent">
```

```
  <populationModel id="ConstantPopulation.t:h1n1pdm_HA" spec="ConstantPopulation" popSize="↩
      @popSize.t:h1n1pdm_HA"/>
  <treeIntervals id="TreeIntervals.t:h1n1pdm_HA" spec="beast.base.evolution.tree.TreeIntervals" ↩
      tree="@Tree.t:h1n1pdm_HA"/>
  </distribution>
  <distribution id="CoalescentConstant.t:h1n1pdm_NA" spec="Coalescent">
  <populationModel id="ConstantPopulation.t:h1n1pdm_NA" spec="ConstantPopulation" popSize="↩
      @popSize.t:h1n1pdm_HA"/>
  <treeIntervals id="TreeIntervals.t:h1n1pdm_NA" spec="beast.base.evolution.tree.TreeIntervals" ↩
      tree="@Tree.t:h1n1pdm_NA"/>
  </distribution>
```

Now, $CoalescentConstant.t : h1n1pdm\_NA$, the tree prior on the NA segment, uses the population size parameter of the HA segment. Essentially, we are done now. However, the parameter $popSize.t : h1n1pdm\_NA$ is still in the xml, and it is still being explored. We can remove it from the xml, and BEAST will no longer explore it. Let's remove it from the state block.

```
  <parameter id="kappa.s:h1n1pdm_NA" spec="parameter.RealParameter" lower="0.0" name="stateNode">↩
      2.0</parameter>
  <parameter id="popSize.t:h1n1pdm_HA" spec="parameter.RealParameter" lower="0.0" name="stateNode↩
      " upper="3">0.1</parameter>
  <parameter id="popSize.t:h1n1pdm_NA" spec="parameter.RealParameter" lower="0.0" name="stateNode↩
      ">0.3</parameter>
</state>
```

Next, we have to remove it from the priors block

```
  <prior id="PopSizePrior.t:h1n1pdm_HA" name="distribution" x="@popSize.t:h1n1pdm_HA">
      <OneOnX id="OneOnX.2" name="distr"/>
  </prior>
  <prior id="PopSizePrior.t:h1n1pdm_NA" name="distribution" x="@popSize.t:h1n1pdm_NA">
      <OneOnX id="OneOnX.3" name="distr"/>
  </prior>
</distribution>
```

Then, we can remove it from the operators block.

```
<operator id="PopSizeScaler.t:h1n1pdm_HA" spec="kernel.BactrianScaleOperator" parameter="@popSize↩
    .t:h1n1pdm_HA" upper="10.0" weight="3.0"/>
<operator id="CoalescentConstantBICEPSEpochTop.t:h1n1pdm_NA" spec="EpochFlexOperator" scaleFactor↩
    ="0.1" tree="@Tree.t:h1n1pdm_NA" weight="2.0"/>
<operator id="CoalescentConstantBICEPSEpochAll.t:h1n1pdm_NA" spec="EpochFlexOperator" ↩
    fromOldestTipOnly="false" scaleFactor="0.1" tree="@Tree.t:h1n1pdm_NA" weight="2.0"/>
<operator id="CoalescentConstantBICEPSTreeFlex.t:h1n1pdm_NA" spec="TreeStretchOperator" ↩
    scaleFactor="0.01" tree="@Tree.t:h1n1pdm_NA" weight="2.0"/>
<operator id="CoalescentConstantTreeRootScaler.t:h1n1pdm_NA" spec="kernel.BactrianScaleOperator" ↩
    rootOnly="true" scaleFactor="0.1" tree="@Tree.t:h1n1pdm_NA" upper="10.0" weight="3.0"/>
<operator id="CoalescentConstantUniformOperator.t:h1n1pdm_NA" spec="kernel.BactrianNodeOperator" ↩
    tree="@Tree.t:h1n1pdm_NA" weight="30.0"/>
<operator id="CoalescentConstantSubtreeSlide.t:h1n1pdm_NA" spec="kernel.BactrianSubtreeSlide" ↩
    tree="@Tree.t:h1n1pdm_NA" weight="15.0"/>
<operator id="CoalescentConstantNarrow.t:h1n1pdm_NA" spec="Exchange" tree="@Tree.t:h1n1pdm_NA" ↩
    weight="15.0"/>
<operator id="CoalescentConstantWide.t:h1n1pdm_NA" spec="Exchange" isNarrow="false" tree="@Tree.↩
    t:h1n1pdm_NA" weight="3.0"/>
<operator id="CoalescentConstantWilsonBalding.t:h1n1pdm_NA" spec="WilsonBalding" tree="@Tree.↩
    t:h1n1pdm_NA" weight="3.0"/>
<operator id="PopSizeScaler.t:h1n1pdm_NA" spec="kernel.BactrianScaleOperator" parameter="@popSize↩
    .t:h1n1pdm_NA" upper="10.0" weight="3.0"/>
```

Parameters can have multiple operators, for example, the tree parameters have multiple operators acting on the tree. Removing just one of them will not fix this parameter.

And, lastly, we have to keep the parameter from being logged to a file.

```
<log idref="kappa.s:h1n1pdm_HA"/>
<log idref="freqParameter.s:h1n1pdm_HA"/>
<log idref="freqParameter.s:h1n1pdm_NA"/>
<log idref="kappa.s:h1n1pdm_NA"/>
<log idref="popSize.t:h1n1pdm_HA"/>
<log idref="CoalescentConstant.t:h1n1pdm_HA"/>
<log idref="popSize.t:h1n1pdm_NA"/>
<log idref="CoalescentConstant.t:h1n1pdm_NA"/>
</logger>
```

Now, the parameter $popSize.t : h1n1pdm_NA$ is no longer in the xml, and BEAST will no longer explore it. We could have removed it from the MCMC by only changing the xml at two points, how?

6 Fixing parameters

Fixing parameter is rather simple. The intial value of each parameter is given by its value in the state block. Let's say we know the clock rate of the HA segment is 0.005. We can fix the clock rate of the HA segment by changing the xml as follows:

```
<parameter id="clockRate.c:h1n1pdm_NA" spec="parameter.RealParameter" lower="0.0" name="stateNode↩
    ">1.0</parameter>
```

to

```
  <parameter id="clockRate.c:h1n1pdm_NA" spec="parameter.RealParameter" lower="0.0" name="↩
      stateNode">0.005</parameter>
```

The parameter $clockRate.c : h1n1pdm\_NA$ is now initalized to 0.005 but will still be explored by the MCMC. To fix the parameter, we have to remove all operators acting on the parameter. Let's move to the operators block and search for $clockRate.c : h1n1pdm\_NA$

The operators acting on the parameter are:

```
  <operator id="StrictClockRateScaler.c:h1n1pdm_HA" spec="AdaptableOperatorSampler" weight="1.5">
  <parameter idref="clockRate.c:h1n1pdm_HA"/>
  <operator id="AVMNOperator.h1n1pdm_HA" spec="kernel.AdaptableVarianceMultivariateNormalOperator↩
      " allowNonsense="true" beta="0.05" burnin="400" initial="800" weight="0.1">
      <transformations id="AVMNSumTransform.h1n1pdm_HA" spec="operator.kernel.Transform$↩
          LogConstrainedSumTransform">
          <f idref="freqParameter.s:h1n1pdm_HA"/>
      </transformations>
      <transformations id="AVMNLogTransform.h1n1pdm_HA" spec="operator.kernel.Transform$↩
          LogTransform">
          <f idref="clockRate.c:h1n1pdm_HA"/>
          <f idref="kappa.s:h1n1pdm_HA"/>
      </transformations>
      <transformations id="AVMNNoTransform.h1n1pdm_HA" spec="operator.kernel.Transform$↩
          NoTransform">
          <f idref="Tree.t:h1n1pdm_HA"/>
```

```
      </transformations>
  </operator>
  <operator id="StrictClockRateScalerX.c:h1n1pdm_HA" spec="kernel.BactrianScaleOperator" ↩
      parameter="@clockRate.c:h1n1pdm_HA" upper="10.0" weight="3.0"/>
</operator>
<operator id="strictClockUpDownOperator.c:h1n1pdm_HA" spec="AdaptableOperatorSampler" weight="1.5↩
    ">
  <parameter idref="clockRate.c:h1n1pdm_HA"/>
  <tree idref="Tree.t:h1n1pdm_HA"/>
  <operator idref="AVMNOperator.h1n1pdm_HA"/>
  <operator id="strictClockUpDownOperatorX.c:h1n1pdm_HA" spec="operator.kernel.↩
      BactrianUpDownOperator" scaleFactor="0.75" weight="3.0">
      <up idref="clockRate.c:h1n1pdm_HA"/>
      <down idref="Tree.t:h1n1pdm_HA"/>
  </operator>
</operator>
```

We could just remove these lines, but then all other operators that reference the following would fail.

```
<operator id="AVMNOperator.h1n1pdm_HA" spec="kernel.AdaptableVarianceMultivariateNormalOperator" ↩
    allowNonsense="true" beta="0.05" burnin="400" initial="800" weight="0.1">
<transformations id="AVMNSumTransform.h1n1pdm_HA" spec="operator.kernel.Transform$↩
    LogConstrainedSumTransform">
    <f idref="freqParameter.s:h1n1pdm_HA"/>
</transformations>
<transformations id="AVMNLogTransform.h1n1pdm_HA" spec="operator.kernel.Transform$LogTransform">
    <f idref="clockRate.c:h1n1pdm_HA"/>
    <f idref="kappa.s:h1n1pdm_HA"/>
</transformations>
<transformations id="AVMNNoTransform.h1n1pdm_HA" spec="operator.kernel.Transform$NoTransform">
    <f idref="Tree.t:h1n1pdm_HA"/>
</transformations>
</operator>
```

To prevent that, we can remove the clock rate from the block.

```
  <operator id="AVMNOperator.h1n1pdm_HA" spec="kernel.AdaptableVarianceMultivariateNormalOperator↩
      " allowNonsense="true" beta="0.05" burnin="400" initial="800" weight="0.1">
  <transformations id="AVMNSumTransform.h1n1pdm_HA" spec="operator.kernel.Transform$↩
      LogConstrainedSumTransform">
      <f idref="freqParameter.s:h1n1pdm_HA"/>
  </transformations>
  <transformations id="AVMNLogTransform.h1n1pdm_HA" spec="operator.kernel.Transform$LogTransform"↩
      >
      <f idref="kappa.s:h1n1pdm_HA"/>
  </transformations>
  <transformations id="AVMNNoTransform.h1n1pdm_HA" spec="operator.kernel.Transform$NoTransform">
      <f idref="Tree.t:h1n1pdm_HA"/>
  </transformations>
  </operator>
```

And put this block instead in the next operator where it was previously referenced with idref. This is the case for the KappaScaler:

```
<operator id="KappaScaler.s:h1n1pdm_HA" spec="AdaptableOperatorSampler" weight="0.05">
<parameter idref="kappa.s:h1n1pdm_HA"/>
<operator idref="AVMNOperator.h1n1pdm_HA"/>
<operator id="KappaScalerX.s:h1n1pdm_HA" spec="kernel.BactrianScaleOperator" parameter="@kappa.↩
    s:h1n1pdm_HA" scaleFactor="0.1" upper="10.0" weight="0.1"/>
</operator>
```

We can now make sure that the *AVMNOperator.h1n1pdm_HA* is properly initialized here instead.

```
<operator id="KappaScaler.s:h1n1pdm_HA" spec="AdaptableOperatorSampler" weight="0.05">
<parameter idref="kappa.s:h1n1pdm_HA"/>
<operator id="AVMNOperator.h1n1pdm_HA" spec="kernel.AdaptableVarianceMultivariateNormalOperator" ↩
    allowNonsense="true" beta="0.05" burnin="400" initial="800" weight="0.1">
    <transformations id="AVMNSumTransform.h1n1pdm_HA" spec="operator.kernel.Transform$↩
        LogConstrainedSumTransform">
        <f idref="freqParameter.s:h1n1pdm_HA"/>
    </transformations>
    <transformations id="AVMNLogTransform.h1n1pdm_HA" spec="operator.kernel.Transform$↩
        LogTransform">
        <f idref="kappa.s:h1n1pdm_HA"/>
    </transformations>
    <transformations id="AVMNNoTransform.h1n1pdm_HA" spec="operator.kernel.Transform$NoTransform"↩
        >
        <f idref="Tree.t:h1n1pdm_HA"/>
    </transformations>
</operator>
<operator id="KappaScalerX.s:h1n1pdm_HA" spec="kernel.BactrianScaleOperator" parameter="@kappa.↩
    s:h1n1pdm_HA" scaleFactor="0.1" upper="10.0" weight="0.1"/>
</operator>
```

When we now run the xml in beast, the clock rate of the HA segment is fixed to 0.005.

- Bayesian Evolutionary Analysis with BEAST 2 (**BEAST2book2014**)
- BEAST 2 website and documentation: http://www.beast2.org/
- Join the BEAST user discussion: http://groups.google.com/group/beast-users