

Beast II 101: Part 2



[XML](#)

[Add-ons](#)

[Applications](#)

Remco R. Bouckaert

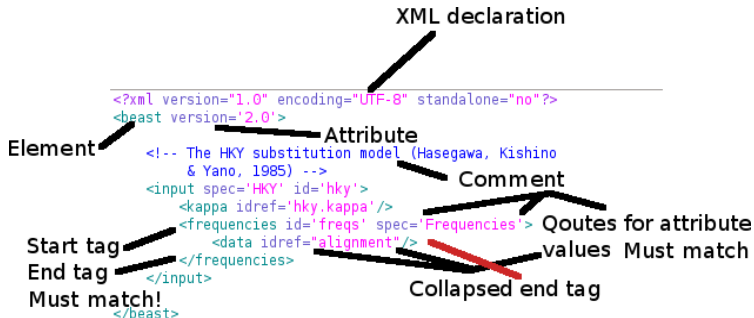
remco@cs.{auckland|waikato}.ac.nz

Department of Computer Science

University of Auckland & University of Waikato

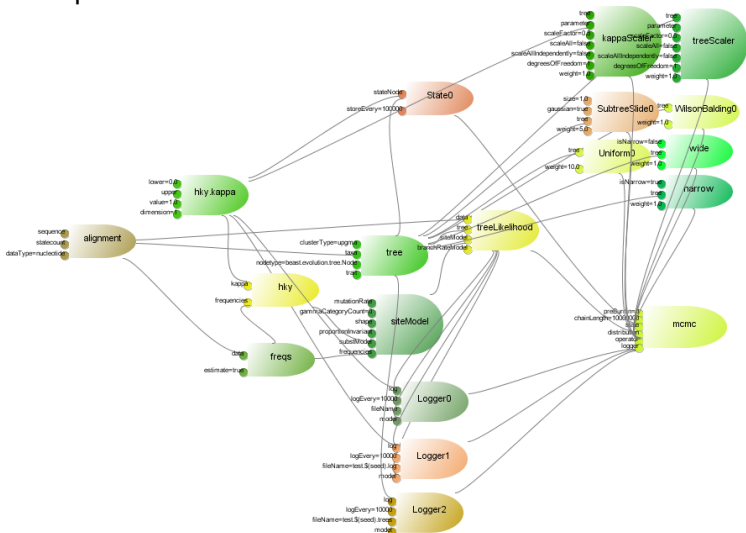
What is XML?

"The Extensible Markup Language (XML) is a simple text-based format for representing structured information"



Reserved characters in attribute values: " (") ' (') < (<) > (>) & (&) e.g. x="""

Let's put this model into XML



First the hky-scale operator

name, spec, id, idref

- XML element `input` can be used for every plugin.
- Specify `name` to match with input name.
- Specify `spec` to identity Plugin.
- XML id/idref mechanism to reuse Plugins.
- XML attributes for primitives (Integer, Double, Boolean, String).

```
<input name='operator' id='kappaScaler'  
  spec='beast.evolution.operators.ScaleOperator'  
  scaleFactor='0.5' weight='1'>  
  <input name='parameter' idref='hky.kappa' />  
</input>
```

XML

Add-ons

Applications

Top level `beast` element can be used to define namespaces in the usual Java fashion.

```
<beast namespace="beast.core:beast.evolution.operators">
```

This allows shortening of spec-values:

```
<input name='operator' id='kappaScaler'  
  spec='beast.evolution.operators.ScaleOperator'  
  scaleFactor='0.5' weight='1'>  
  <input name='parameter' idref='hky.kappa' />  
</input>
```

becomes

```
<input name='operator' id='kappaScaler' spec='ScaleOperator'  
  scaleFactor='0.5' weight='1'>  
  <input name='parameter' idref='hky.kappa' />  
</input>
```

Input elements with name attributes equal the name's value as element name

```
<input name='xyz'></input> == <xyz></xyz>
```

so

```
<input name='operator' id='kappaScaler' spec='ScaleOperator'  
  scaleFactor='0.5' weight='1'>  
  <input name='parameter' idref='hky.kappa' />  
</input>
```

equals

```
<operator id='kappaScaler' spec='ScaleOperator'  
  scaleFactor='0.5' weight='1'>  
  <parameter idref='hky.kappa' />  
</operator>
```

If idref is only attribute in element, an attribute with element name and before the idref.

```
<name idref="some-id"/> == name='@some-id'
```

So

```
<operator id='kappaScaler' spec='ScaleOperator'  
  scaleFactor='0.5' weight='1'>  
  <parameter idref='hky.kappa' />  
</operator>
```

equals

```
<operator id='kappaScaler' spec='ScaleOperator'  
  scaleFactor="0.5" weight="1" parameter="@hky.kappa"/>
```

```
<beast version='2.0' namespace='x.y.z:a.b.c' >  
<map name='xyz' >x.y.z.Class </map>
```

element `<xyz>` is expanded to

```
<input name='xyz' spec='x.y.z.Class'>
```

<input >

<run > spec must be `beast.core.Runnable`

<distribution > spec must be `beast.core.Distribution`

<operator > spec must be `beast.core.Operator`

<logger > spec=`beast.core.Logger`

<data > spec=`beast.evolution.alignment.Alignment`

<sequence > spec=`beast.evolution.alignment.Sequence`

<state > spec=`beast.core.State`

<parameter > spec=`beast.core.parameter.RealParameter`

<tree > spec=`beast.evolution.tree.Tree`

<plate > mainly for `Beauti` templates


```
<input name='substModel' id="hky" spec="HKY">  
  <input name='kappa' idref="hky.kappa" >  
    <input name='frequencies' id="freqs" spec="Frequencies">  
      <input name='data' idref="alignment"/>  
    </input>  
  </input>
```

```
<input spec="TreeLikelihood">  
  <input name='data' idref='alignment' />  
  <input name='tree' idref='tree' />  
  <input name='siteModel' spec="SiteModel">  
    <input name='substModel' idref='hky' />  
  </input>  
</input>
```

Assuming namespace='beast.evolution.sitemodel:
beast.evolution.substitutionmodel:
beast.evolution.likelihood'

Input elements with name attributes equal the name's value as element name

```
<input name='xyz' ></input> == <xyz></xyz>
```

Applying to the example

```
<substModel id="hky" spec="HKY">  
  <kappa idref="hky.kappa" >  
  <frequencies id="freqs" spec="Frequencies">  
    <data idref="alignment"/>  
  </frequencies>  
</substModel>
```

```
<distribution spec="TreeLikelihood">  
  <data idref='alignment' />  
  <tree idref='tree' />  
  <siteModel spec="SiteModel">  
    <substModel idref='hky' />  
  </siteModel>  
</distribution>
```

[XML](#)[Add-ons](#)[Applications](#)

if idref is only attribute in element, an attribute with element name and before the idref.

```
<name idref="some-id"/> == name='@some-id'
```

Applying to the example

```
<substModel id="hky" spec="HKY" kappa="@hky.kappa" >  
  <frequencies id="freqs" spec="Frequencies"  
    data="@alignment"/>  
</substModel>
```

```
<distribution data="@alignment" spec="TreeLikelihood"  
  tree="@tree">  
  <siteModel spec="SiteModel" substModel='@hky' />  
</distribution>
```

Note: you still can use any of the previous versions!
These are just short-cuts.

- specified in name attribute

```
<input name="xyz" >
```

- if not, use element name

```
<xyz value="3" >
```

- if input, use 'value' when there is text content, but no element content

```
<input>3</input>
```

- if idref is specified, use the referred object

```
<xyz idref="other" > or xyz='@other'
```

- specified in value attribute

```
<xyz value="3" >
```

- if not, use value of (non-reserved) attribute

```
<input xyz="3" >
```

- if not, use text content when there is text content, but no element content

```
<input>3</input>
```

XML

Add-ons

Applications

```

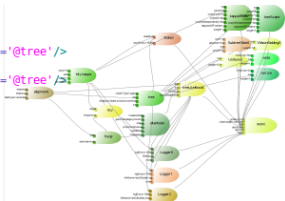
<run chainLength="1000000" id="mcmc" preBurnin="0" spec="MCMC">
  <state>
    <parameter id="hky.kappa" name="stateNode" value="1.0"/>
    <tree id="tree" name="stateNode" spec="beast.util.ClusterTree">
      <taxa idref="alignment"/>
    </tree>
  </state>

  <distribution id="likelihood" spec="TreeLikelihood" tree="@tree" data="@alignment">
    <siteModel spec="SiteModel">
      <substModel id="hky" kappa="@hky.kappa" spec="HKY">
        <frequencies spec="Frequencies" data="@alignment" estimate="true"/>
      </substModel>
    </siteModel>
  </distribution>

  <operator id='kappaScaler' spec='ScaleOperator' scaleFactor="0.5" weight="1" parameter="@hky.kappa"/>
  <operator id='treeScaler' spec='ScaleOperator' scaleFactor="0.5" weight="1" tree="@tree"/>
  <operator spec='Uniform' weight="10" tree="@tree"/>
  <operator spec='SubtreeSlide' weight="5" gaussian="true" size="1.0" tree="@tree"/>
  <operator id='narrow' spec='Exchange' isNarrow='true' weight="1" tree="@tree"/>
  <operator id='wide' spec='Exchange' isNarrow='false' weight="1" tree="@tree"/>
  <operator spec='WilsonBalding' weight="1" tree="@tree"/>

  <logger logEvery="10000" fileName="test.%(seed).log">
    <model idref='likelihood'/>
    <log idref='likelihood'/>
    <log idref='hky.kappa'/>
    <log spec='beast.evolution.tree.TreeHeightLogger' tree="@tree"/>
  </logger>
  <logger logEvery="10000" fileName="test.%(seed).trees" log="@tree"/>
  <logger logEvery="10000">
    <model idref='likelihood'/>
    <log idref='likelihood'/>
    <ESS spec='ESS' name='log' arg="@likelihood"/>
    <log idref='hky.kappa'/>
    <ESS spec='ESS' name='log' arg="@hky.kappa"/>
  </logger>
</run>

```



XMLParser produces semi sensible parser error messages:

Error 124 parsing the xml input file

This plugin (treeLikelihood) has no input with name xxx. Choose one of these inputs: data,tree,siteModel,branchRateModel,useAmbiguities

Error detected about here:

```
<beast>
  <run id='mcmc' spec='MCMC'>
    <distribution id='posterior' spec='CompoundDistribution'>
      <distribution id='treeLikelihood' spec='TreeLikelihood'>
```

and

Error 122 parsing the xml input file

Cannot create class: CompoundDistribution. Class could not be found. Did you mean beast.core.util.CompoundDistribution?

Error detected about here:

```
<beast>
  <run id='mcmc' spec='MCMC'>
    <distribution id='posterior' spec='CompoundDistribution'>
```

[XML](#)[Add-ons](#)[Applications](#)

A Beast 2 add-on is a library based on Beast 2

Why add-ons:

- Making work easier citable
- Making the core easier to learn – it's a lot smaller / cleaner
- Separating out stable / experimental code / dead code
- ...

- SnAP - multi-species coalescent for SNP and AFLP data <http://code.google.com/p/snap-mcmc/>
- beastii - utilities, Peter Will's AARS substitution model <http://code.google.com/p/beastii/>
- Subst-BMA - Bayes model averaging over subst. models
<http://code.google.com/p/subst-bma/>
- EBSP/*BEAST - Joseph's thesis work
- Experimental phylogeography
- David Welch's Prevalence/SI-likelihood
- Sibon's MCMC monitoring thing
- ...

What makes an Add-on

- A jar file that contains the code
- A jar file with the source
- Example XML files
- Documentation
- A Beauti 2 template

Recommended directory structure:

<code>myAddOn/./beast2</code>	Beast 2 files
<code>myAddOn/src</code>	source files
<code>myAddOn/examples</code>	XML examples
<code>myAddOn/build</code>	class files
<code>myAddOn/build/dist</code>	jar files
<code>myAddOn/lib</code>	libraries used (if any)
<code>myAddOn/doc</code>	documentation
<code>myAddOn/templates</code>	Beauti templates (optional)

Setting up an Add-on

Checkout Beast 2 code, available at

<http://code.google.com/p/beast2>

Setting up an add-on in IntelliJ, basic steps

- Make a project containing Beast 2
- Create new module, e.g. called MyAddOn
- Create module dependency of MyAddOn on Beast 2

see SDK documentation for more details.

Setting up an add-on in Eclipse, basic steps

- Make a project containing Beast 2
- Create new project, e.g. called MyAddOn
- Add beast 2 to the Java build path of MyAddOn

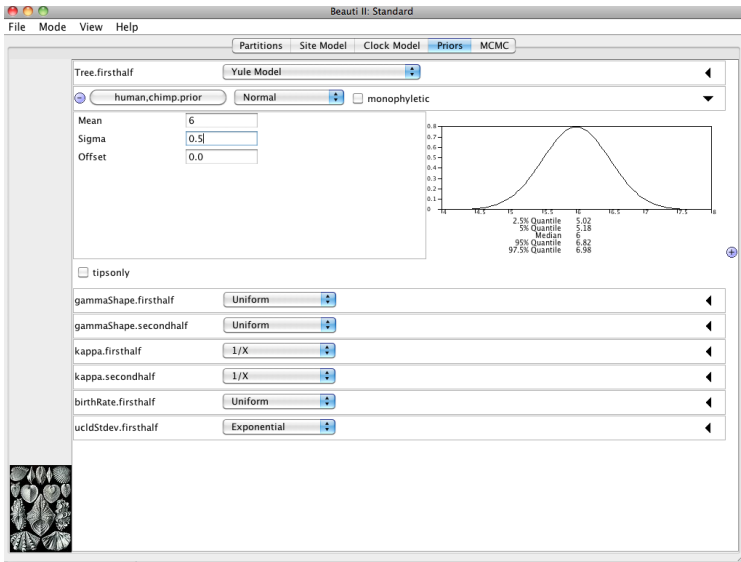
see SDK documentation for more details.

Setting up an add-on in Hudson (for automatic regression testing): ask Walter

- Create Add-on files containing add-on classes only
- Put on a web-site
- Download to \$BEAST_HOME/beastlib
- It will be picked up from there when running BeastMCMC or Beauti

Future work: automate this process, provide catalogue, GUI, etc.

Beauti 2: replacement of Beauti 1



Tree: firsthalf Yule Model

human_chimp_prior Normal monophyletic

Mean 6

Sigma 0.5

Offset 0.0

tipsonly

gammaShape.firsthalf Uniform

gammaShape.secondhalf Uniform

kappa.firsthalf 1/X

kappa.secondhalf 1/X

birthRate.firsthalf Uniform

ucldStdev.firsthalf Exponential

Graph showing a normal distribution curve with the following statistics:

2.5% Quantile	5.02
5% Quantile	5.18
Median	6
95% Quantile	6.82
97.5% Quantile	6.98

[XML](#)[Add-ons](#)[Applications](#)

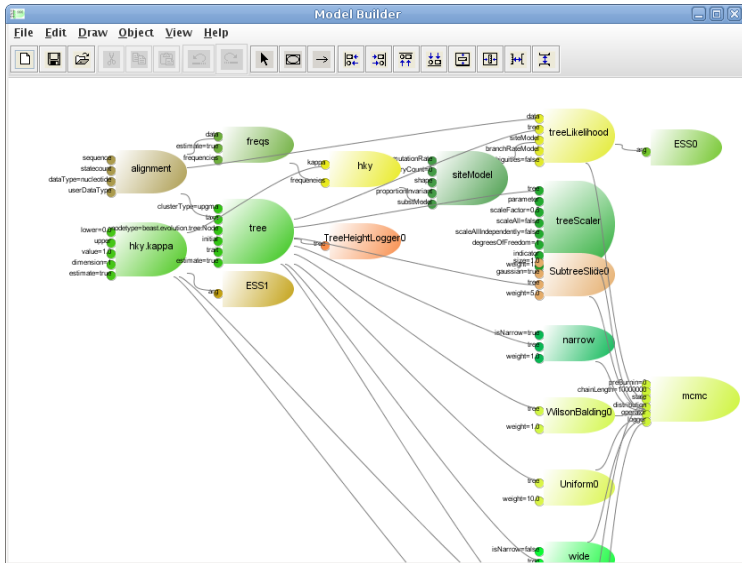
More about Beauti in Part III

Model builder: GUI for graphical manipulation of Plugins

XML

Add-ons

Applications



In development...

Spreadsheet: calculates partial results on the fly

Beast II 101

Bouckaert



XML

Add-ons

Applications

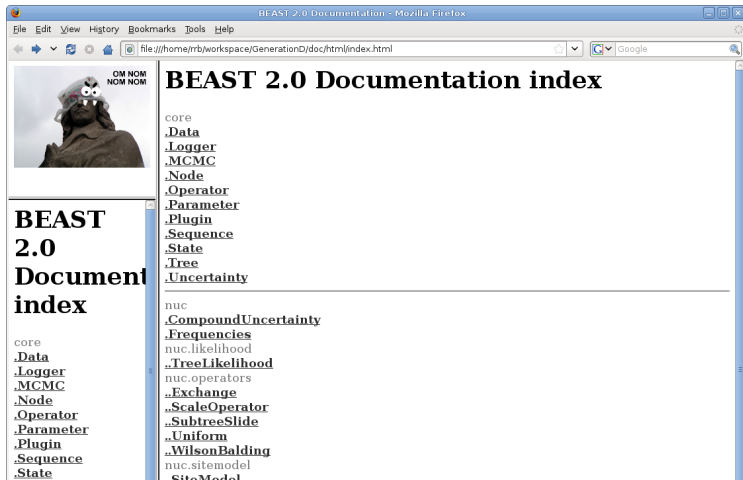
	A	B	C
1	State	State([\$B2,\$B3],100000)	
2	hky.kappa	RealParameter(0,0,null,2,0,1,true)	4.0
3	tree	ClusterTree(upgma,\$B5,beast.evolution.tree.Node,null,null,true)	0.107031...
4			
5	alignment	Alignment([\$B6,\$B7,\$B8,\$B9,\$B10,\$B11],null,nucleotide)	
6	Sequence	Sequence(4,human,AGAAATATGTCTGATAAAAGAGTTACT...	
7	Sequence	Sequence(4,chimp,AGAAATATGTCTGATAAAAGAAATTA...	
8	Sequence	Sequence(4,bonobo,AGAAATATGTCTGATAAAAGAAATTA...	
9	Sequence	Sequence(4,gorilla,AGAAATATGTCTGATAAAAGAGTTACTT...	
10	Sequence	Sequence(4,orangutan,AGAAATATGTCTGACAAAAGAGTTA...	
11	Sequence	Sequence(4,siamang,AGAAATACGTCTGACGAAAGAGTTAC...	
12			
13	treeLikelihood	TreeLikelihood(\$B5,\$B3,\$B14,null,false)	-1986.84...
14	siteModel	SiteModel(null,0,null,null,\$B15)	
15	hky	HKY(\$B2,\$B16)	
16	frequencies	Frequencies(\$B5,true,null)	
17			
18	kappaScaler	ScaleOperator(null,\$B2,0.5,false,false,1,1.0)	
19	treeScaler	ScaleOperator(\$B3,null,0.5,false,false,1,1.0)	
20	Uniform	Uniform(\$B3,10.0)	
21	SubtreeSlide	SubtreeSlide(1.0,true,\$B3,5.0)	
22	narrow	Exchange(true,\$B3,1.0)	33
23	wide	Exchange(false,\$B3,1.0)	66.0
24	WilsonBalding	WilsonBalding(\$B3,1.0)	
25			
26	Logger	Logger([\$B13,\$B27,\$B2,\$B28],10000,null,\$B13,autodetect)	
27	ESS	ESS(\$B13)	
28	ESS	ESS(\$B2)	
29			
30	Logger	Logger([\$B13,\$B2,\$B31],10000,test \$(seed).log,\$B13,autodetect)	
31	TreeHeightLog...	TreeHeightLogger(\$B3)	
32			
33	Logger	Logger([\$B3],10000,test \$(seed).trees,null,autodetect)	
34	mcmc	MCMC(0,10000000,\$B1,\$B13,[\$B18,\$B19,\$B20,\$B21,\$B22,\$B2...	
35			
36			

In development...

Documentation

XML documentation provided through

- @Description annotation on plug in
- Tooltip text on inputs
- getCitation method
- Input validation rules



BEAST 2.0 Documentation - Mozilla Firefox

file:///home/r/b/workspace/GenerationD/doc/html/index.html

OM NOM NOM NOM

BEAST 2.0 Documentation index

core
[.Data](#)
[.Logger](#)
[.MCMC](#)
[.Node](#)
[.Operator](#)
[.Parameter](#)
[.Plugin](#)
[.Sequence](#)
[.State](#)
[.Tree](#)
[.Uncertainty](#)

nuc
[.CompoundUncertainty](#)
[.Frequencies](#)
 nuc.likelihood
[.TreeLikelihood](#)
 nuc.operators
[.Exchange](#)
[.ScaleOperator](#)
[.SubtreeSlide](#)
[.Uniform](#)
[.WilsonBalding](#)
 nuc.sitemodel
[.SiteModel](#)

- o Sequence generator, for simulation studies
- o Sequence with XML merging through Beauti, handy for scripting
- o XMLParser to beautify XML
- o Alignment viewer: navigate an alignment
- o Log analyser: prints statistics of a trace log from command line

```
~> java beast.app.BeastMCMC
```

Usage: BeastMCMC [options] <Beast.xml>

where <Beast.xml> the name of a file specifying a Beast run
and the following options are allowed:

- resume : read state that was stored at the end of the last run from file
- overwrite : overwrite existing log files (**if** any). By **default**, existing
- seed [<int>|random] : sets random number seed (**default** 127), or picks a
- threads <int> : sets number of threads (**default** 1)
- beastlib <path> : Colon separated list of directories. All jar files in