A Proposal for Beast 2.0

Remco R. Bouckaert

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

Department of Computer Science

University of Auckland & University of Waikato

Vision

To provide tools for computational science that are

- *easy to use*, that is, well documented, have intuitive user interfaces with small learning curve.
- *open access*, that is, open source, open xml format, facilitating reproducability of results, runs on many platforms.
- *easy to extend*, by having extensibility in design.

Scope

Efficient testing of probabilistic hypotheses for sequence data analysis involving tree models.

Vision

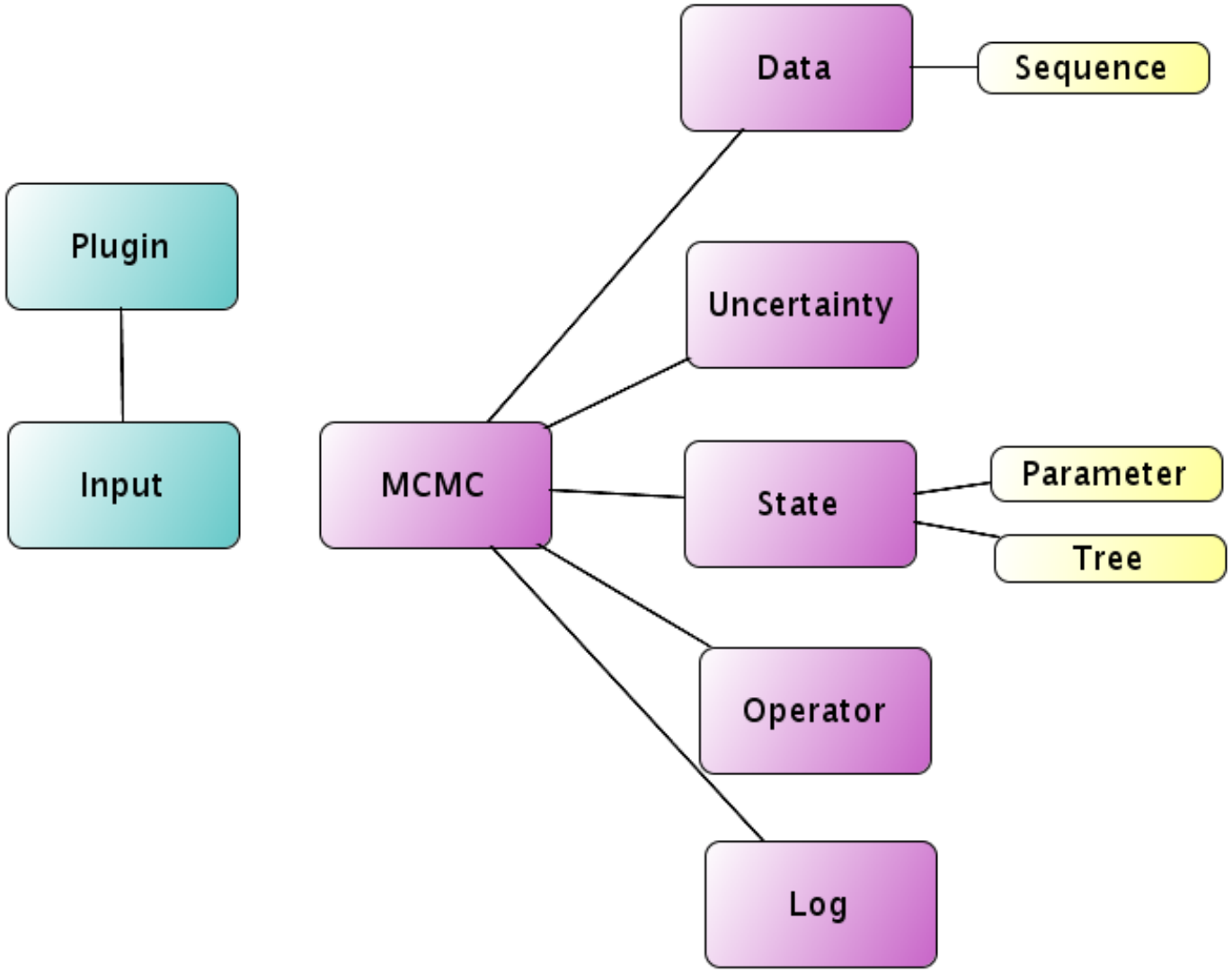
To provide tools for computational science that are

- *easy to use*, that is, well documented, have intuitive user interfaces with small learning curve.
- *open access*, that is, open source, open xml format, facilitating reproducability of results, runs on many platforms.
- *easy to extend*, by having extensibility in design.

Scope

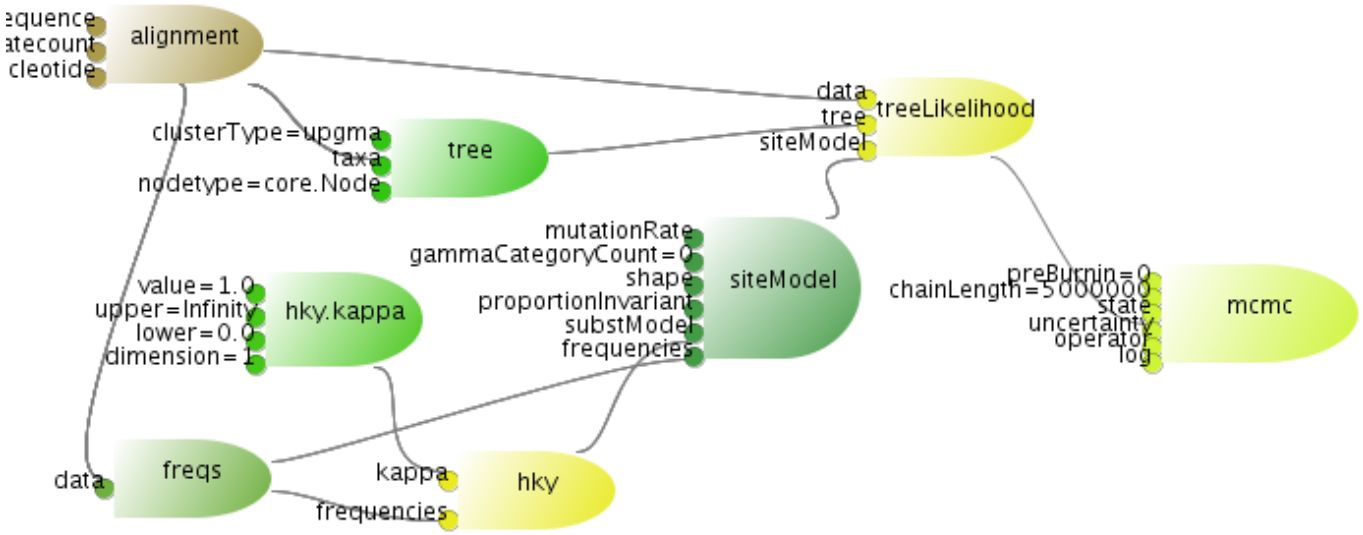
Efficient testing of probabilistic hypotheses for sequence data analysis involving tree models.

Core


Data
Sequence
Plugin
Uncertainty
Input
MCMC
State
Parameter
Tree
Operator
Log

Phylosophy

Everything is a plug-in



Plug-ins provide...

- connection with with other plug-ins/values through 'inputs'
- validation
- documentation
- 'XML parsing'

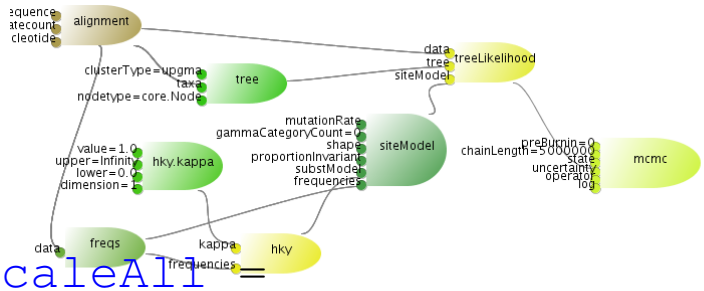
Plugin class

```
@Description("Description␣goes␣here")
public class Plugin {
    public void initAndValidate(State state)

    public String getCitations()

    public String getID()
    public void setID(String sID)

    public void store(int nSample)
    public void restore(int nSample)
} // class Plugin
```

HKY Plugin


```
@Description("HKY85 (Hasegawa, Kishino & Yano, 1985) substitution
public class HKY extends 'Plugin' {
    public Input<Frequencies> m_freqs = new Input<Frequencies>("f
    public Input<Parameter> m_kappa = new Input<Parameter>("kappa

    @Override public void initAndValidate(State state) throws Exc
        initialiseEigen();
    }

    public void getTransitionProbabilities(double distance, doubl
        ...
    }
    @Override public void store(int nSample) {}
    @Override public void restore(int nSample) {
        updateMatrix = true;
        updateIntermediates = true;
    }
    @Override public String getCitation() {
        return "Hasegawa, M., Kishino, H and Yano, T. 1985. Datin
    }
} // class HKY
```

Inputs



Simple primitives

```
public Input<Boolean> m_pScaleAll =
    new Input<Boolean>("scaleAll",
        "if true, all elements of a parameter (not tree) are scal
        new Boolean(false));
```

Other plugins

```
public Input<Frequencies> m_freqs =
    new Input<Frequencies>("frequencies",
        "frequencies nucleotide letters");
```

Multiple inputs

```
public Input<List<Parameter>> m_pParameters =
    new Input<List<Parameter>>("parameter",
        "parameter, part of the state",
        new ArrayList<Parameter>());
```

Input validation

Default: OPTIONAL (see previous slide)

If input is REQUIRED:

```
public Input<Parameter> m_kappa =
    new Input<Parameter>("kappa",
        "kappa parameter in HKY model",
        Validate.REQUIRED);

public Input<List<Operator>> m_operators =
    new Input<List<Operator>>("operator",
        "operator for generating proposals in MCMC state space",
        new ArrayList<Operator>(), Validate.REQUIRED);
```

If input is XOR:

```
public Input<Tree> m_pTree =
    new Input<Tree>("tree",
        "if specified, all tree branch length are scaled");
public Input<Parameter> m_pParameter =
    new Input<Parameter>("parameter",
        "if specified, this parameter is scaled"
        , Validate.XOR, m_pTree);
```

State

- State is explicit in XML & as object (unlike Beast 1)
- Contains parameters and trees
- Operators work on the state

```
public double proposal(State state) throws Exception {...}
```

- Uncertainty calculated given state

```
public double calculateLogP(State state) throws Exception {...
```

- State can be interrogated on value of parameters/trees

```
state.getParameter(/**Parameter**/ p)
state.getTree(/**Tree**/ t)
```

Store/Restore

BEAST 1: push model

- Model
- Event handling
- Framework and plug-ins responsible
- Elegant, undocumented, requires lots of Panadol

BEAST 2: pull model

- store/restore
- sample number provided to prevent multiple (re)stores
- MCMC only tells its 'uncertainties' to store/restore, plug-ins responsible

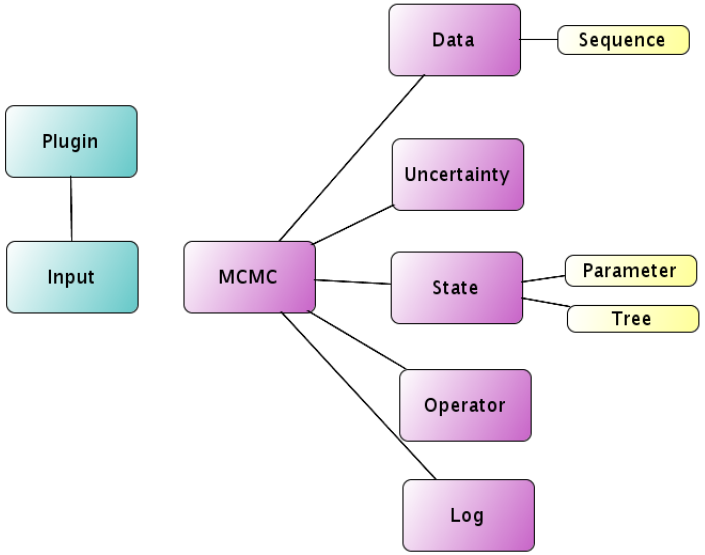
XML

```
<mcmc id="mcmc" chainLength="5000000">
    <state>
        <parameter id="hky.kappa" value="1.0" lower="0.0"/>
        <tree spec='util.ClusterTree' id='tree' clusterType='upgma'>
           <input name='taxa' idref='alignment'/>
        </tree>
    </state>

    <uncertainty id='likelihood' idref="treeLikelihood"/>

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

XML parsing/writing provided by the framework XML

Reserved elements

<mcmc >
<uncertainty >
<operator >
<log >
<data >
<sequence >
<state >
<parameter >
<tree >




```
<beast version='2.0' namespace='x.y.z:' >
<map name='elementName' >x.y.z.Class </map>
```

XML


```
<!-- The HKY substitution model (Hasegawa, Kishino & Yano, 1985)          -->
<input spec='HKY' id="hky">
    <parameter name='kappa' idref='hky.kappa'/>
    <input id='freqs' name='frequencies' spec='Frequencies'>
        <input name='data' idref='alignment'/>
    </input>
</input>

<!-- site model                                                            -->
<input spec='SiteModel' id="siteModel">
    <input idref='freqs' name='frequencies'/>
    <input name='substModel' idref='hky'/>
</input>

<input spec='TreeLikelihood' id="treeLikelihood">
    <input name='data' idref="alignment"/>
    <input name='tree' idref="tree"/>
    <input name='siteModel' idref="siteModel"/>
</input>
```

XML


```
    <!-- The sequence alignment (each sequence refers to a taxon above).         -->
    <!-- ntax=6 nchar=768                                                         -->
    <!-- npatterns=69                                                             -->
    <data id="alignment" dataType="nucleotide">
        <sequence taxon="human">
AGAAATATGTCTGATAAAAGAGTTACTTTGATAGAGTAAATAATAGGAGCTTAAACCCCCTTATTTCTACTAGGACTATGAGAATCGAAC
        </sequence>
        <sequence taxon="chimp">
AGAAATATGTCTGATAAAAGAATTACTTTGATAGAGTAAATAATAGGAGTTCAAATCCCCTTATTTCTACTAGGACTATAAGAATCGAAC
        </sequence>
        <sequence taxon="bonobo">
AGAAATATGTCTGATAAAAGAATTACTTTGATAGAGTAAATAATAGGAGTTTAAATCCCCTTATTTCTACTAGGACTATGAGAGTCGAAC
        </sequence>
        <sequence taxon="gorilla">
AGAAATATGTCTGATAAAAGAGTTACTTTGATAGAGTAAATAATAGAGGTTTAAACCCCCTTATTTCTACTAGGACTATGAGAATTGAAC
        </sequence>
        <sequence taxon="orangutan">
AGAAATATGTCTGACAAAAGAGTTACTTTGATAGAGTAAAAAATAGAGGTCTAAATCCCCTTATTTCTACTAGGACTATGGGAATTGAAC
        </sequence>
        <sequence taxon="siamang">
AGAAATACGTCTGACGAAAGAGTTACTTTGATAGAGTAAATAACAGGGGTTTAAATCCCCTTATTTCTACTAGAACCATAGGAGTCGAAC
        </sequence>
    </data>
```

XML

Reserved attributes:

```
<input id='myId'
      idref='otherId'
      name='inputName'
      spec='x.y.z.MyClass' />
```

Resolving plug-in class:

- specified in spec attribute
- if not, get from element2class map
- if not, use element name (and hope it shows up in the namespace somewhere).

XML

Resolving input name:

- specified in name attribute

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

- if not, use (non-reserved) attribute name

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

- if not, use element name

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

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

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

XML

Resolving input value:

- if idref is specified, use the referred object

```
<input idref="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

Parsing rules: Processing non reserved attributes

```
<input otherAttribute="xyz" />
```

equals

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

Processing non reserved element names

```
<myElement />
```

==

```
<input spec='myElement' name='myElement' />
```

unless 'spec' is a specified attribute, then that overrides, likewise for 'name'

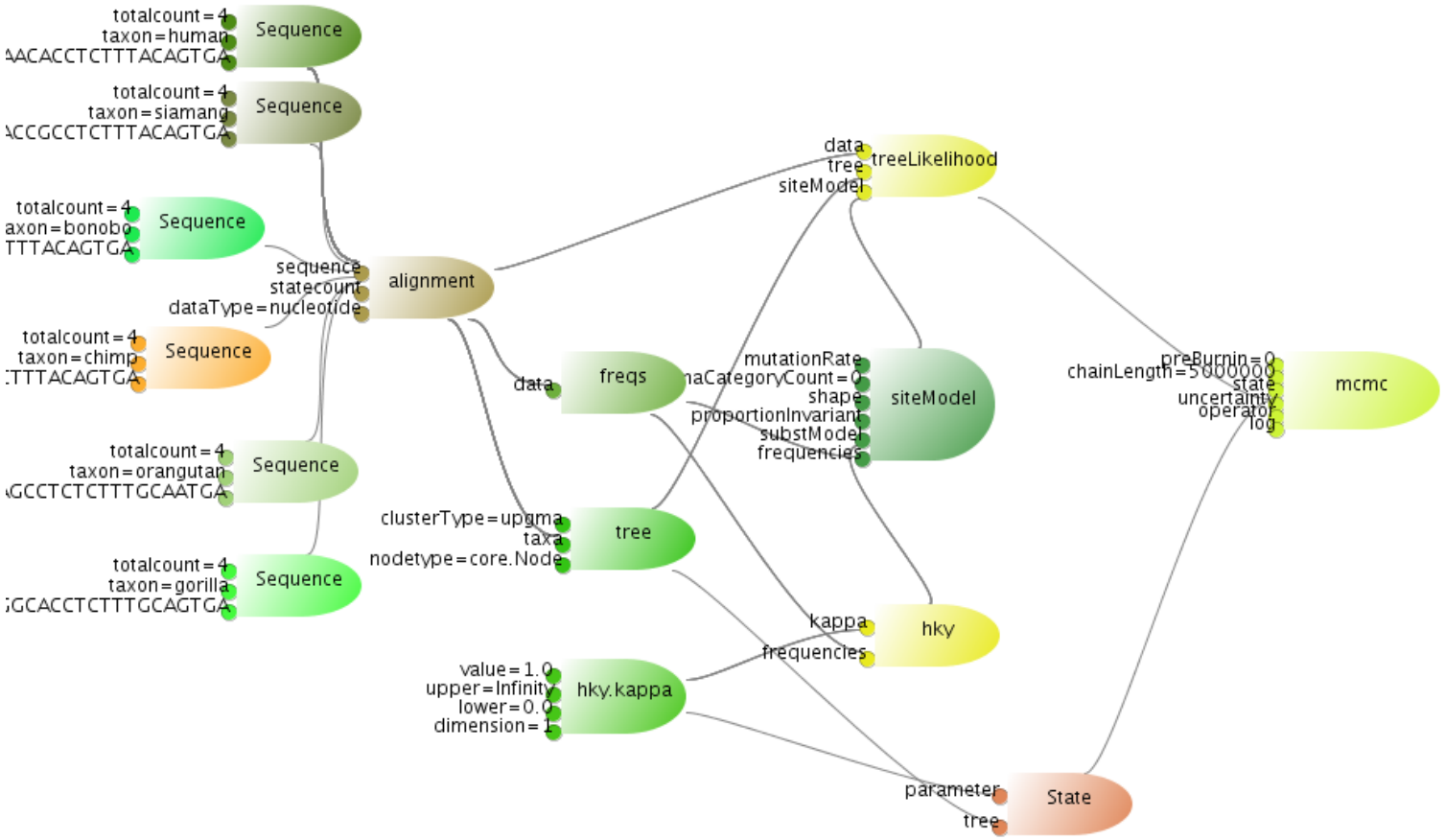
Processing of text content (only when there are no enclosing elements)

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

==

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

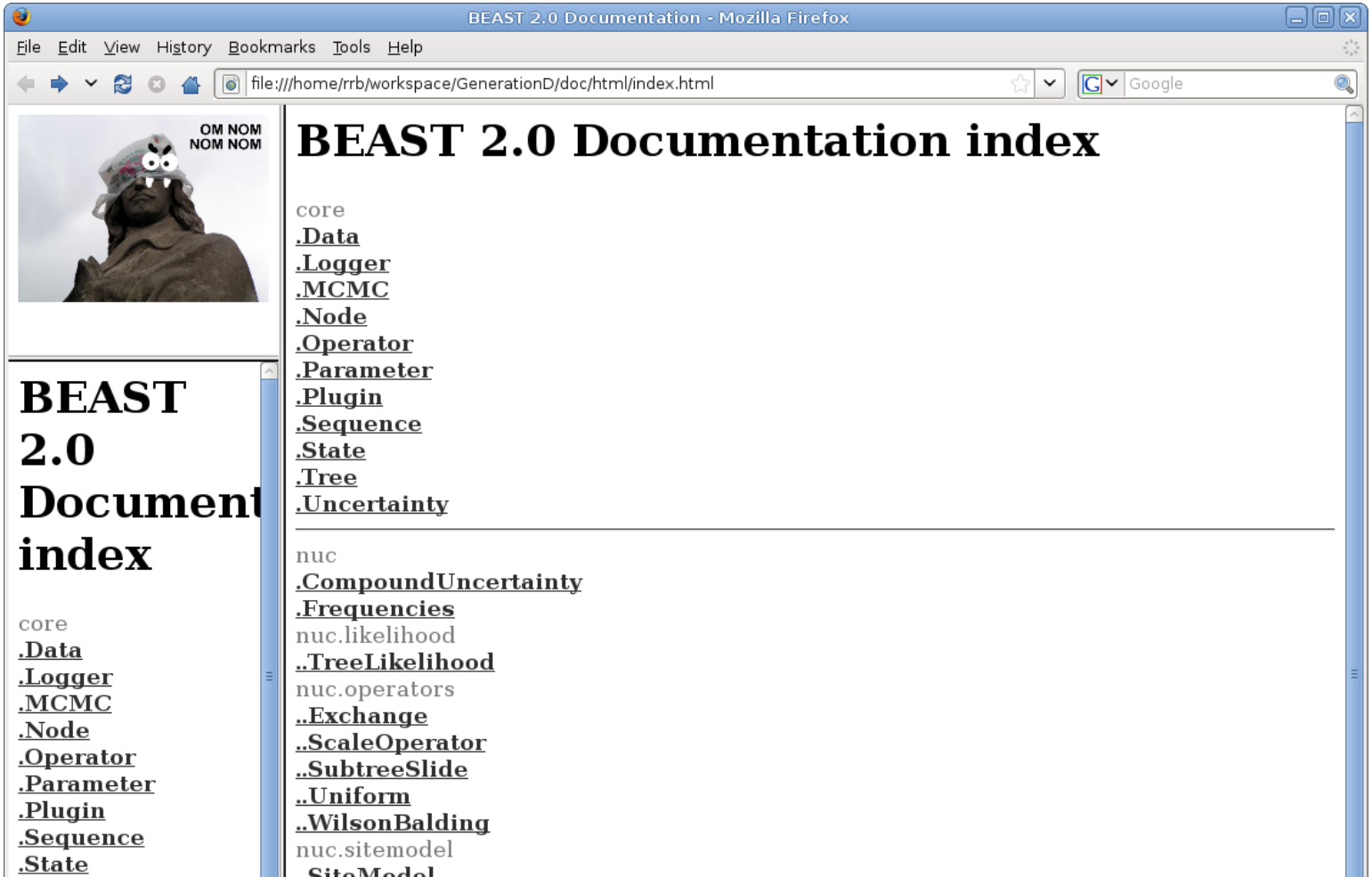
Model builder



Documentation

XML documentation provided through

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



Documentation

BEAST 2.0 Documentation - Mozilla Firefox

File Edit View History Bookmarks Tools Help

file:///home/rrb/workspace/GenerationD/doc/html/index.html

BEAST 2.0 Document index

core
.Data
.Logger
.MCMC
.Node
.Operator
.Parameter
.Plugin
.Sequence
.State
.Tree
.Uncertainty

nuc
.CompoundUncerta
.Frequencies
nuc.likelihood
..TreeLikelihood
nuc.operators
..Exchange
..ScaleOperator
..SubtreeSlide
..Uniform
..WilsonBalding

BEAST 2.0 Documentation: nuc.substitutionmodel.HKY

Specifies transition probability matrix for a given distance

HKY85 (Hasegawa, Kishino & Yano, 1985) substitution model of nucleotide evolution.

Reference:

Hasegawa, M., Kishino, H and Yano, T. 1985. Dating the human-ape splitting by a molecular clock of mitochondrial DNA. Journal of Molecular Evolution 22:160-174.

Inputs:

kappa

type: core.Parameter
kappa parameter in HKY model
Required input

frequencies

type: nuc.Frequencies
frequencies of characters
Required input

Documentation

BEAST 2.0 Documentation - Mozilla Firefox

File Edit View History Bookmarks Tools Help

file:///home/rrb/workspace/GenerationD/doc/html/index.html

BEAST 2.0 Documentation: nuc.substitutionmodel.HKY

Specifies transition probability matrix for a given distance

HKY85 (Hasegawa, Kishino & Yano, 1985) substitution model of nucleotide evolution.

Reference:

Hasegawa, M., Kishino, H and Yano, T. 1985. Dating the human-ape splitting by a molecular clock of mitochondrial DNA. Journal of Molecular Evolution 22:160-174.

Inputs:

kappa

type: core.Parameter

kappa parameter in HKY model

Required input

frequencies

BEAST 2.0 Document index

core
.Data
.Logger
.MCMC
.Node
.Operator
.Parameter
.Plugin
.Sequence
.State
.Tree
.Uncertainty

nuc
.CompoundUncerta
.Frequencies
nuc.likelihood
..TreeLikelihood
nuc.operators
..Exchange
..ScaleOperator
..SubtreeSlide
..Uniform
..WilsonBalding

Documentation

BEAST 2.0 Documentation - Mozilla Firefox

File Edit View History Bookmarks Tools Help

file:///tmp/index.html

java author annotation

TODO list Wednesday 02nd ... | java author annotation - Go... | BEAST 2.0 Documentation

BEAST 2.0 Document index

core
.Data
.Logger
.MCMC
.Node
.Operator
.Parameter
.Plugin
.Sequence
.State
.Tree
.Uncertainty

nuc
.CompoundUncerta
.Frequencies
nuc.likelihood
..TreeLikelihood
nuc.operators
..Exchange
..ScaleOperator
..SubtreeSlide
..Uniform

BEAST 2.0 Documentation: core.Uncertainty

implemented by the following
nuc.CompoundUncertainty
nuc.likelihood.TreeLikelihood
snap.likelihood.SnAPPrior
snap.likelihood.SnAPTreeLikelihood

Not documented!!!

Inputs:

<none>

BEAST 2.0


```
~> java -cp bin 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:
-seed <int> : sets random number seed (default 127)
-threads <int> : sets number of threads (default 1)
```

BEAST 2.0

HKY + nucleotide data, 'random' initial tree
2000 samples in debug mode, scaling if required, auto optimise on, same logging, operators. etc.
testMCMC.xml – great apes

```
6 taxa
768 sites
69 patterns
10M samples  single thread
        Beast 1.6/java  Beast1.6/native Beast 2.0
real    3m55.056s       3m38.670s       2m31.794s
user    3m54.839s       3m38.670s       2m32.162s
sys     0m0.392s        0m0.428s        0m0.476s
```

BEAST 2.0

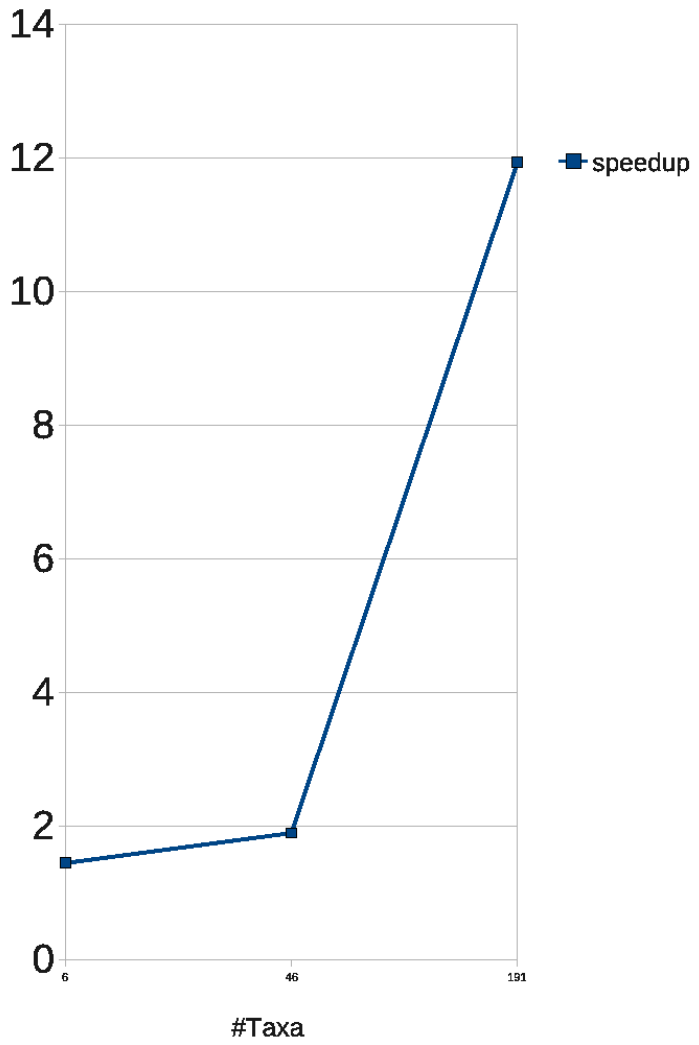
testMCMC.xml –

```
46 taxa
1363 sites
199 patterns
500K samples

Beast1.6/native core
real    1m56.097s
user    1m58.683s
sys     0m0.332s

Beast 2.0 + auto
real    0m56.843s
user    0m59.264s
sys     0m0.428s
```

BEAST 2.0

testMCMC.xml –

```
191 taxa
606 sites
228 patterns
100K samples

Beast1.6/native
real    8m32.418s
user    8m33.060s
sys     0m0.476s

Beast 2.0
real    0m43.962s
user    0m46.051s
sys     0m0.432s
```



Summary Beast 2.0

Extensibility

- Everything is a plug-in
- Plug-in implementation:
  - Add @Description annotation
  - Specify Inputs
  - Implement initAndValidate
  - Optional: getCitation, store, restore
- Get xml-parsing, Input validation, xml-documentation for free
- Store/restore more transparent
- Explicit state object

Documentation

- Framework 'forces' proper documentation habits
- Automatic documentation generation

Performance

- Up to 1 order of magnitude better performance