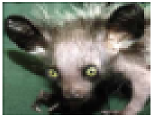
Beast II 101: Part 3



Remco R. Bouckaert

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

Department of Computer Science

University of Auckland & University of Waikato

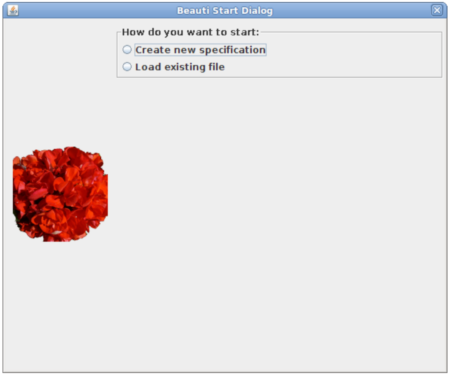
What is Beauti 2

A GUI for manipulating Beast 2 specifications

Features:

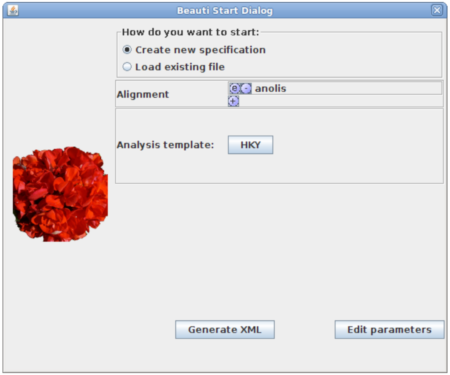
- Read/write XML specifications
- Customizable GUI through templates
- Interactive validation of specification
- Automatically picks up plug-ins from Add-ons
- Batch merging of alignments to XML specifications

Start up



Before editing anything, Beauti needs to know either alignments and template OR existing file.

Start up: select alignments



Beauti Start Dialog

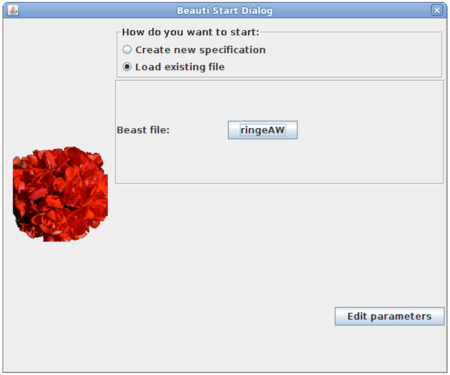
How do you want to start:
[✓ Create new specification]
Load existing file

Alignment: anolis

Analysis template: HKY

Select one or more alignments
Select an analysis template

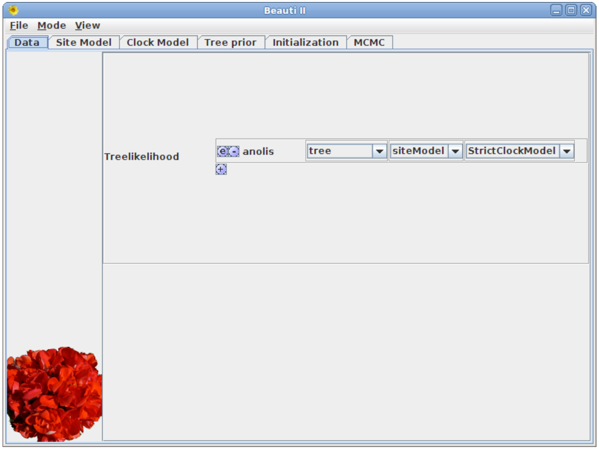
Start up: select existing file


Beauti Start Dialog

How do you want to start:
- ○ Create new specification
- ● Load existing file

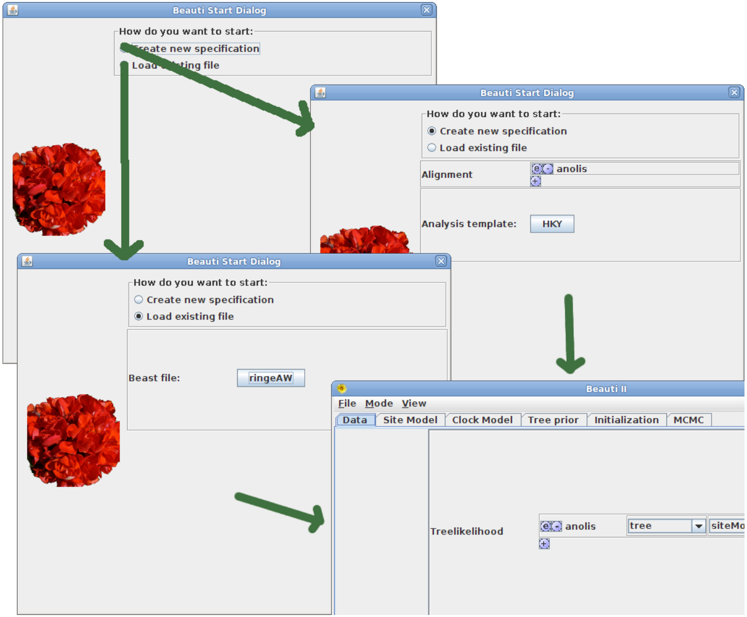
Beast file: ringeAW

Edit parameters



Familiar panel based user interface for configuring specification

Flow



Beauti Start Dialog

How do you want to start:
- Create new specification
- Load existing file

Beauti Start Dialog

How do you want to start:
- [✓ Create new specification]
- Load existing file

Alignment: anolis

Analysis template: HKY

Beauti Start Dialog

How do you want to start:
- Create new specification
- [✓ Load existing file]

Beast file: ringeAW

Treelikelihood: anolis | tree | siteMo

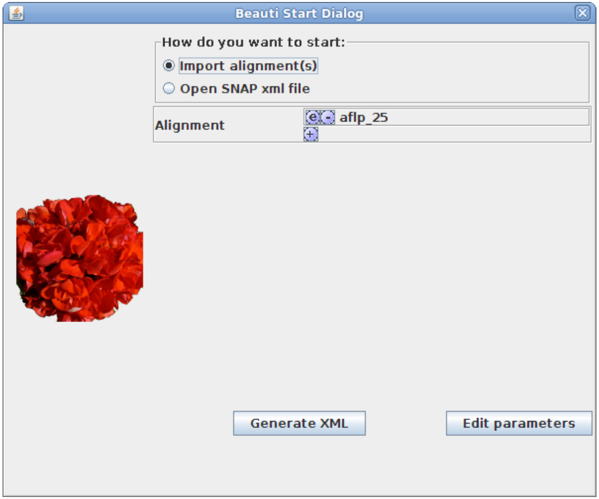
No new alignment selection once editing is started

Start up

```
java beast.app.beauti.Beauti [options]
where options can be one of the following:
-template [template file]
-nex [nexus data file]
-xmldat [beast xml file]
-xml [beast file]
-out [output file name]
-exitaction [writexml|usetemplate|usexml]
```

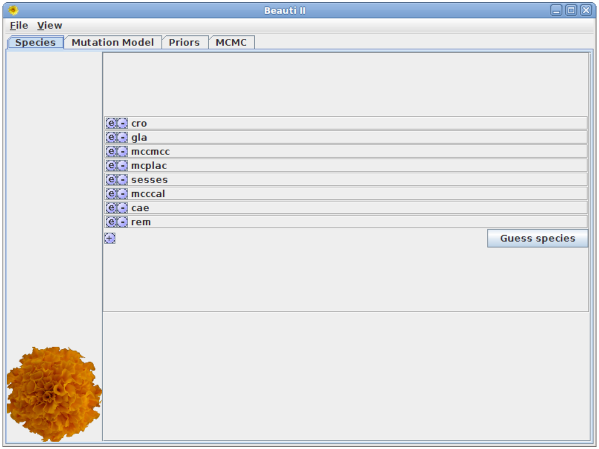
Select proper command line functions to short cut the flow
Multiple alignment files allowed
Batch merging of alignment with template

Start up: another template



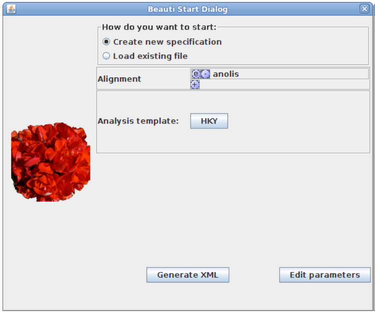
Template for SNP and AFLP analysis
Customised labels, template button invisible

Edit: another template



Custom menu
Only subset of panels used

Spot the differences

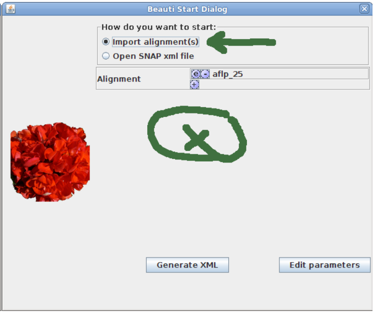

Beauti Start Dialog

How do you want to start:
- (●) Create new specification
- ( ) Load existing file

Alignment: anolis

Analysis template: HKY

Generate XML | Edit parameters


Beauti Start Dialog

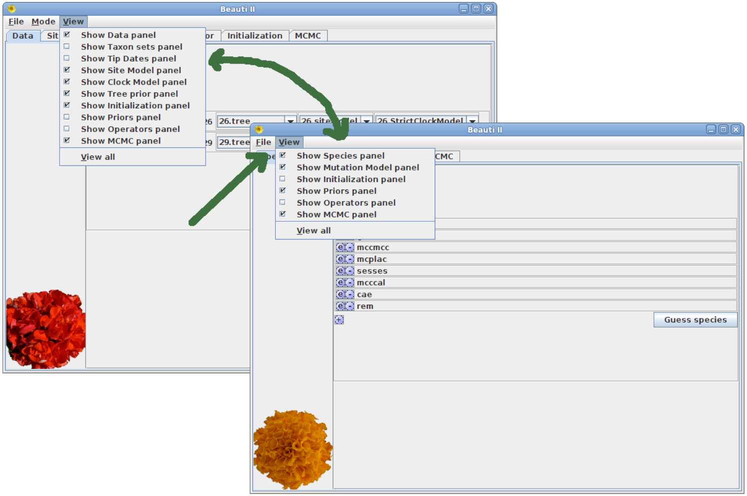
How do you want to start:
- (●) Import alignment(s)
- ( ) Open SNAP xml file

Alignment: aflp_25

Generate XML | Edit parameters

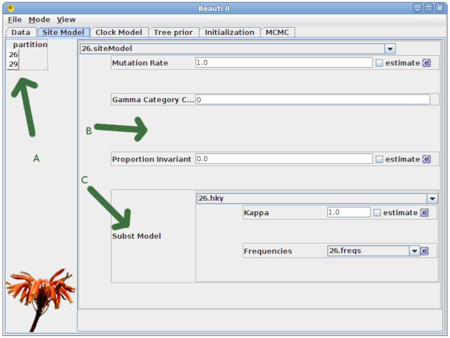
Customized labels
Button visibility

Spot the differences



Customized menus: visibility and label names
Customized panels

Configurable


Beauti II

File Mode View

Data | Site Model | Clock Model | Tree prior | Initialization | MCMC

partition
26
29

26.siteModel

Mutation Rate 1.0 estimate

Gamma Category C... 0

Proportion Invariant 0.0 estimate

Subst Model

26.hky

Kappa 1.0 estimate

Frequencies 26.freqs

A: Partitionable or not

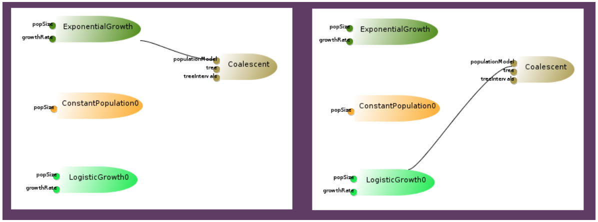
B: Custom behaviour: gamma shape only shown when categories at least 2

C: Expand inputs of a plugin

D: Hide irrelevant inputs

What Beauti does

- Contain a large number of Plugin objects, to ensure a somewhat sensible set of choices
- User changes link in the model graph



- User changes values of primitive (String, Integer, Boolean, Double) inputs
- Automatically update links, e.g. Operator to MCMC

Expert mode allows creation of new Plugin object, but the user is on its own as far as validation is concerned

How to configure Beauti

1: XML template

2: Custom InputEditor classes

Beauti templates

A Beauti template is an XML specification with extra features

- `<plate var='n' range='alignments'>` 'macros'
- `<data id="alignments"/>` for merging alignments
- `<mergepoint>`/`<mergewith>` for merging sub-templates
- `<beauticonfig spec='beast.app.beauti.BeautiConfig'>` for customizing GUI components

Main-templates define type of analysis, e.g. vanilla alignment analysis, *BEAST, Snap

Sub-templates define parts that go anywhere in a main template, e.g. substitution or branch rate models.

`plate` element in templates

Plate behaves like a macro

```
<plate var='n' range='a,b,c'>
<input id='$(n)'/>
</plate>
```

is interpreted as

```
<input id='a'/>
<input id='b'/>
<input id='c'/>
```

For example

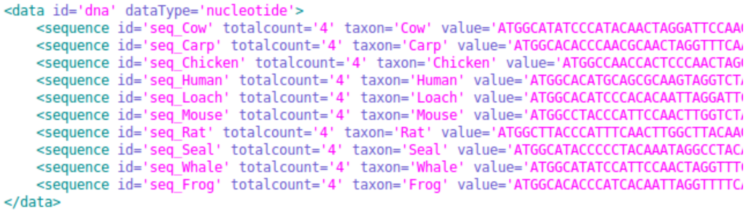
```
<plate var='n' range='#alignments'>
        <input spec='HKY' id='$(n).hky'>
            <kappa idref='$(n).hky.kappa'/>
            <frequencies id='$(n).freqs' spec='Frequencies'
                <data idref='$(n)'/>
            </frequencies>
        </input>
</plate>
```

Merging alignments

Single alignment merging:

```
<data id="#alignments"/>
```

becomes



```
<data id='dna' dataType='nucleotide'>
    <sequence id='seq_Cow' totalcount='4' taxon='Cow' value='ATGGCATATCCCATACAACTAGGATTCCAA
    <sequence id='seq_Carp' totalcount='4' taxon='Carp' value='ATGGCACACCCAACGCAACTAGGTTTCA
    <sequence id='seq_Chicken' totalcount='4' taxon='Chicken' value='ATGGCCAACCACTCCCAACTAG
    <sequence id='seq_Human' totalcount='4' taxon='Human' value='ATGGCACATGCAGCGCAAGTAGGTCT
    <sequence id='seq_Loach' totalcount='4' taxon='Loach' value='ATGGCACATCCCACACAATTAGGATT
    <sequence id='seq_Mouse' totalcount='4' taxon='Mouse' value='ATGGCCTACCCATTCCAACTTGGTCT
    <sequence id='seq_Rat' totalcount='4' taxon='Rat' value='ATGGCTTACCCATTTCAACTTGGCTTACAA
    <sequence id='seq_Seal' totalcount='4' taxon='Seal' value='ATGGCATACCCCCTACAAATAGGCCTAC
    <sequence id='seq_Whale' totalcount='4' taxon='Whale' value='ATGGCATATCCATTCCAACTAGGTTT
    <sequence id='seq_Frog' totalcount='4' taxon='Frog' value='ATGGCACACCCATCACAATTAGGTTTTC
</data>
```

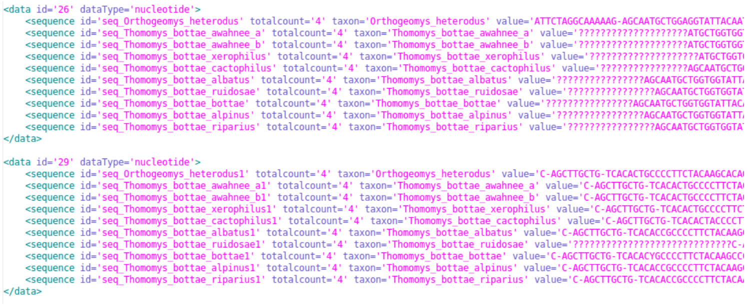
and everywhere in the template `#alignments` becomes `dna`

Merging alignments

Multi alignment merging:

```
<data id="#alignments"/>
```

becomes



```
<data id='26' dataType='nucleotide'>
    <sequence id='seq_Orthogeomys_heterodus' totalcount='4' taxon='Orthogeomys_heterodus' value='ATTCTAGGCAAAAAG-AGCAATGCTGGAGGTATTACAA
    <sequence id='seq_Thomomys_bottae_awahnee_a' totalcount='4' taxon='Thomomys_bottae_awahnee_a' value='???????????????????ATGCTGGTGG
    <sequence id='seq_Thomomys_bottae_awahnee_b' totalcount='4' taxon='Thomomys_bottae_awahnee_b' value='???????????????????ATGCTGGTGG
    <sequence id='seq_Thomomys_bottae_xerophilus' totalcount='4' taxon='Thomomys_bottae_xerophilus' value='???????????????????ATGCTGGT
    <sequence id='seq_Thomomys_bottae_cactophilus' totalcount='4' taxon='Thomomys_bottae_cactophilus' value='???????????????AGCAATGCTG
    <sequence id='seq_Thomomys_bottae_albatus' totalcount='4' taxon='Thomomys_bottae_albatus' value='???????????????AGCAATGCTGGTGGTATT
    <sequence id='seq_Thomomys_bottae_ruidosae' totalcount='4' taxon='Thomomys_bottae_ruidosae' value='???????????????AGCAATGCTGGTGGTA
    <sequence id='seq_Thomomys_bottae_bottae' totalcount='4' taxon='Thomomys_bottae_bottae' value='???????????????AGCAATGCTGGTGGTATTAC
    <sequence id='seq_Thomomys_bottae_alpinus' totalcount='4' taxon='Thomomys_bottae_alpinus' value='???????????????AGCAATGCTGGTGGTATT
    <sequence id='seq_Thomomys_bottae_riparius' totalcount='4' taxon='Thomomys_bottae_riparius' value='???????????????AGCAATGCTGGTGGTA
</data>

<data id='29' dataType='nucleotide'>
    <sequence id='seq_Orthogeomys_heterodus1' totalcount='4' taxon='Orthogeomys_heterodus' value='C-AGCTTGCTG-TCACACTGCCCCTTCTACAAGCACA
    <sequence id='seq_Thomomys_bottae_awahnee_a1' totalcount='4' taxon='Thomomys_bottae_awahnee_a' value='C-AGCTTGCTG-TCACACTGCCCCTTCTA
    <sequence id='seq_Thomomys_bottae_awahnee_b1' totalcount='4' taxon='Thomomys_bottae_awahnee_b' value='C-AGCTTGCTG-TCACACTGCCCCTTCTA
    <sequence id='seq_Thomomys_bottae_xerophilus1' totalcount='4' taxon='Thomomys_bottae_xerophilus' value='C-AGCTTGCTG-TCACACTGCCCCTTC
    <sequence id='seq_Thomomys_bottae_cactophilus1' totalcount='4' taxon='Thomomys_bottae_cactophilus' value='C-AGCTTGCTG-TCACACTACCCCT
    <sequence id='seq_Thomomys_bottae_albatus1' totalcount='4' taxon='Thomomys_bottae_albatus' value='C-AGCTTGCTG-TCACACCGCCCCTTCTACAAG
    <sequence id='seq_Thomomys_bottae_ruidosae1' totalcount='4' taxon='Thomomys_bottae_ruidosae' value='???????????????????????????C-
    <sequence id='seq_Thomomys_bottae_bottae1' totalcount='4' taxon='Thomomys_bottae_bottae' value='C-AGCTTGCTG-TCACACYGCCCCTTCTACAAGCC
    <sequence id='seq_Thomomys_bottae_alpinus1' totalcount='4' taxon='Thomomys_bottae_alpinus' value='C-AGCTTGCTG-TCACACCGCCCCTTCTACAAG
    <sequence id='seq_Thomomys_bottae_riparius1' totalcount='4' taxon='Thomomys_bottae_riparius' value='C-AGCTTGCTG-TCACACCGCCCCTTCTACA
</data>
```

and everywhere in the template `#alignments` becomes `26,29`

`mergepoint` **and** `mergewith` **elements**

Main templates define `mergepoint`s with ids

Sub-templates define `mergwith` and point to the `mergepoint`s in main template

Typical usage of sub-templates

- specify a new substitution model,
- specify prior distributions on parameters of the model
- specify operators on parameters of the model

`mergepoint` **and** `mergewith` **Example**

Main-template

```
<plate var='n' range='#alignments'>
    <mergepoint id='substitutionmodel'/>
</plate>
```

Sub-template

```
<mergewith point='substitutionmodel'>
    <input spec='WAG' id='$(n).WAG'/>
</mergewith>
```

Interpretation of main template

```
<plate var='n' range='#alignments'>
    <input spec='WAG' id='$(n).WAG'/>
</plate>
```

Configuring Beauti GUI

In template:
`<beauticonfig spec='beast.app.beauti.BeautiConfig'>`
for customizing

- which panels are shown at start up
- which menus are visible in menubar
- which buttons are visible
- which inputs are hidden
- which inputs are expanded inline
- which labels are used

See files in `beast2/templates` directory for details

Developing custom GUI components

From a developers view: everything is a Plugin

Beauti is a tool for

- connecting inputs with Plugins
- configuring inputs

In Beauti, a panel for a Plugins shows a list of InputEditors.

To create customized behaviour for inputs of specific types, override InputEditor

InputEditor

Beauti uses the input editor associated with the in type of the InputEditor

```
public class MyInputEditor extends InputEditor {

    /** tell the type of input that this Input
        Editor applies to **/
        @Override
    public Class<?> type() {
        return MyPlugin.class;
    }

    /** custom implementation **/

}
```

See code for gory details of custom implementation possibilities

`beast.app.beauti` packages for examples

ListInputEditor

Dealing with list of inputs: extend ListInputEditor and implement `type()` and `baseType()`

```
public class MyInputEditor extends ListInputEditor {

    /** tell the type of input that this Input
        Editor applies to **/
        @Override
        public Class<?> type() {
                return List.class;
        }
        @Override
        public Class<?> baseType() {
                return Operator.class;
        }

    /** custom implementation **/

}
```

All done!

Go forth and develop new Plugins and Beauti templates now!