Tutorial for bppsuite and testnh

Laurent Guéguen
(laurent.gueguen@univ-lyon1.fr)

May 4, 2021

In this tutorial, we will see several examples of usage of programs from bppsuite and testnh. Bppsuite and testnh programs depend on bio++ libraries, and all are available there:
`http://biopp.univ-montp2.fr/wiki/index.php/Main_Page`
as well as on Github:
`http://github.com/BioPP`

We will give examples how to simulate sequences, perform maximum likelihood estimates of models and branch lengths, perform ancestral reconstruction and substitution mapping.

All the options are described in bppsuite and testnh manuals, see github repositories.

1 Homogeneous simple model

Maximum Likelihood

In `hom_ML.bpp`, we define an homogeneous model on DNA evolution. The results are stored in directory `hom_Res`, defined through a macro. Make sure this directory exists before executing `bppml`. Then the command is:

```
bppml param=hom_ML.bpp
```

Ancestral sequence inference

Given a process and data, it is possible to infer ancestral sequence probabilities, using program `bppancestor`. Usually this is done after the estimate of the best process given the data. In our case, we use the parameter file `hom_Anc.bpp`

```
bppancestor param=hom_Anc.bpp
```

Substitution Mapping

Given a process and data, it is possible to infer the counts of ancestral events on sites and/or branches, using the program `mapnh` (from `Testnh` suite). Usually this is done after the estimate of the best process given the data.

Any type of events can be counted, see the manual of Testnh for details. Here, we count the substitutions from AT to GC & from GC to AT, and the counts are output as branch lengths in trees.

```
mapnh param=hom_Map_GC.bpp
```

It is also possible to count events that a similar process could have done on the same history, as used for the estimates of dN & dS. In the file `hom_Map_GC_Norm.bpp`, the lines

```
nullProcessParams = zz=1

output.counts = PerBranch(prefix=$(RESULT_DIR)/counts_, splitNorm=true)
```

assess how the parameters of the similar process are set in comparison to the original one, and "splitNorm=true" means to output specific trees for these "theoretical" counts (in `_norm.dnd` trees).

Here `zz=1` has no meaning, just to use the same process as the original. But, it could have been:

```
nullProcessParams = HKY85.theta_1=0.5
```

to set a model with no tendency towards AT vs GC.

2 Non-homogeneous simple model

Now, we define several models of the tree, with specific assignations to branches. For example, in file `nonhom_ML.bpp`.

In this case, two models are assigned to different branches. In the case of trees read from newick files, the branches are numbered in the reading order of the description string. In NHX format, branches are explicitly numbered, which is much easier to handle.

Aliases

We see in the declaration of second model

```
model2 = T92(theta=HKY85.theta_1)
```

that theta parameters of both models are linked, which means they are constrained to share the same value (here equilibrium GC content).

Another way to alias parameters is through the line:

```
likelihood.alias = HKY85.kappa_1 -> T92.kappa_2
```

which means that kappa parameter model HKY85 is equal to kappa parameter of model T92.

Substitution rates

In this example, we define a distribution of substitution rates:

```
rate_distribution1 = Invariant(dist=Gamma(n=4))
```

which is here a Gamma distribution with 4 classes and a specific probability of null rate for constant sites.

The *a posteriori* probabilities of the 5 classes of this distribution are written in the info file `aln1.infos_1`. We can see that for non-constant sites the *a posteriori* probabilities of the first class (the invariant class) are quasi null.

3 Computing on several data

3.1 Shared process

In `multidata_ML.bpp`, we compute the likelihood of the same process as before on two alignments. In the result file `multiData_Res/aln1.params.txt`, the line

```
result=phylo1 + phylo2
```

means that the result (and optimized) log-likelihood is the sum of log-likelihoods on both alignments. It means that the result is the same as the log-likelihood of the concatenation of both alignments.

This result calculation is the default behaviour, but it is possible to define its own calculation, such as:

```
result=phylo1 + phylo2/2
```

3.2 Several processes

If we want to compute likelihood on several data with different trees, we have to define several processes. See `multiProc_ML.bpp`.

Both optimized trees are in the output directory. In this case, all branch lengths are optimized independently. If we want to share branch lengths between trees, we alias them. In this case, NHX format is more convenient. See `multiProcNHX_ML.bpp`, where the matching branches of trees from files `aln1.nhx` and `aln2.nhx` are linked.

4 Mixing processes

It is also possible to mix several processes, in a predetermined or probabilistic (mixture or HMM) way. In the following examples, both processes differ on the topology of the tree.

In `partition_ML.bpp`, an *a priori* partition of the data is proposed, defined in `process3`, which uses `process1` and `process2`.

In `mixture_ML.bpp`, on each position the likelihood is the mean of the likelihood of both processes, averaged through optimized probabilities. To know which process best fits each site, in `mixture_Res/infos`, the site log-likelihoods and *a posteriori* probabilities are displayed.

In `autocorr_ML.bpp`, the modeling is also a probabilistic mixture of both models, but with a between sites auto-correlation for each process.