Estimate of dN and dS with Bio++ programs

We present a tutorial to estimate of dN and dS by normalized stochastic mapping, with some programs developed on Bio++ libraries.

Programs

Bio++ libraries are fully described on the wiki site:

http://biopp.github.io

The methods presented here depend on two separate sets of programs, **bppsuite** and **testnh**.

This site explains how to install all libraries and programs.

An apptainer image is available there:
`https://pbil.univ-lyon1.fr/bpp-doc/images/bpp_debian.sif`.

All programs share the same syntax and options. In the next examples, minimum configuration files are provided, but all options of the programs are described in the bppsuite manual and in the testnh manual.

Estimate on simulated sequences

Simulating sequence evolution

The program to simulate the evolution of sequences is `bppseqgen`, and we can call like this:

```
bppseqgen THETA_ROOT=0.7 THETA_EQ=0.3 param=seq.bpp
```

In the configuration file `seq.bpp`, one unique model is used on all the branches of the tree described in file `tree.dnd`.

The model used is the standard model of Yang & Nielsen for codons (see the bio++ link for a description of the parametrization), with F1X4 parametrization of the codon distribution. `THETA_ROOT` is the GC content at the root of the tree, and `THETA_EQ` is the equilibrium GC content. The proportions of C and G are equal, as well as the proportions of A and T.

The output alignment is in file `simul_seq.fa`.

Maximum likelihood estimate of model and branch lengths

From a given alignment and a given topology, we estimate the most likely model and branch lengths with program `bppml`:

```
bppml SEQ=simul_seq.fa param=ml.bpp
```

`tree_init.dnd` is the starting tree for the optimization procedure. The only important feature in this tree is the topology, since it is not optimized. The branches are numbered in the order they appear in the file, for example here: `(((0,(1,2)3)4,(5,6)7)8,9);` (for user declaration, use Nhx format (see bppsuite manual)). The optimized tree and model are respectively in files `tree_ml.dnd` and `model_ml.params`.

Estimates of dN and dS

Finally, to compute the normalized expected dN and dS of the tree, we use `mapnh` with the optimal tree and model obtained previously. Several outputs are available for the estimates of dN and dS.

Estimates per branch

We get one tree file per type of substitution, with branch lengths replaced by the branch specific estimates.

```
mapnh SEQ=simul_seq.fa MODEL=model_ml TREE=tree_ml \
                       param=map_per_branch.bpp
```

The dN and dS counts on all branches are respectively in tree files `simul_seq.fa.counts_dN.dnd` and `simul_seq.fa.counts_dS.dnd`.

Here is an example of usage in R :

```
library(ape)

treedN=read.tree(file="simul_seq.fa.counts_dN.dnd")
treedS=read.tree(file="simul_seq.fa.counts_dS.dnd")

plot(treedS)

## length of the edges
treedN$edge.length
treedS$edge.length

## primate edges are numbered 2, 3, 4, 5, 6
## dN and dS on these edges

sum(treedN$edge.length[2:6])
sum(treedS$edge.length[2:6])

## estimated dN/dS on the primate clade

sum(treedN$edge.length[2:6])/sum(treedS$edge.length[2:6])
```

Counting per site

It is also possible to get the estimates per site:

```
mapnh SEQ=simul_seq.fa MODEL=model_ml TREE=tree_ml \
                       param=map_per_site.bpp
```

The output is in file `simul_seq.fa_sites.count.sged`, with a column for dS and a column for dN.

Counting per site per branch

It is also possible to get the estimates per site per branch:

```
mapnh SEQ=simul_seq.fa MODEL=model_ml TREE=tree_ml \
                       param=map_per_site_per_branch.bpp
```

The dN and dS counts on all branches and sites are respectively in files `simul_seq.fa_branch_sites_dN.count` and `simul_seq.fa_branch_sites_dS.count`, with columns numbered with branch numbers.

Estimate with stationarity assumption

We perform similar estimates, but with the assumption of stationarity:

```
bppml SEQ=simul_seq.fa param=ml_stat.bpp

mapnh SEQ=simul_seq.fa MODEL=model_stat_ml TREE=tree_stat_ml \
                       PREF=_stat param=map_per_branch.bpp
```

And as before, in R:

```
treedNstat=read.tree(file="simul_seq.fa_stat.counts_dN.dnd")
treedSstat=read.tree(file="simul_seq.fa_stat.counts_dS.dnd")

## length of the edges
treedNstat$edge.length
treedSstat$edge.length

## primate edges are numbered 2, 3, 4, 5, 6
## dN and dS on these edges

sum(treedNstat$edge.length[2:6])
sum(treedSstat$edge.length[2:6])

## estimated dN/dS on the primate clade

sum(treedNstat$edge.length[2:6])/sum(treedSstat$edge.length[2:6])
```

Simulation and estimate in non-homogeneous modeling

Here is an example of simulation and estimate with non-homogeneous modeling. One model is used for the branches in the primate clade (numbers 0 to 4), and another one for the other branches (numbers 5 to 9).

To reduce the number of parameters for the estimate, we assume that ω as well as equilibrium proportions of C/(C+G) and A/(A+T) are equal in both models.

```
bppseqgen THETA_ROOT=0.7 THETA_EQ_PRIM=0.7 THETA_EQ_OTHER=0.3 \
                         param=seq_nonhom.bpp

bppml SEQ=simul_seq_nonhom.fa param=ml_nonhom.bpp

mapnh SEQ=simul_seq_nonhom.fa MODEL=model_nonhom_ml \
      TREE=tree_nonhom_ml param=map_per_branch.bpp
```

And in R:

```
treedNnonhom=read.tree(file="simul_seq_nonhom.fa.countsdN.dnd")
treedSnonhom=read.tree(file="simul_seq_nonhom.fa.countsdS.dnd")

## length of the edges
treedNnonhom$edge.length
treedSnonhom$edge.length

## primate edges are numbered 2, 3, 4, 5, 6
## dN and dS on these edges

sum(treedNnonhom$edge.length[2:6])
sum(treedSnonhom$edge.length[2:6])

## estimated dN/dS on the primate clade

sum(treedNnonhom$edge.length[2:6])/sum(treedSnonhom$edge.length[2:6])
```