BayesCAT Manual

Heejung Shim

November 22, 2014

Contents

1 Overview

The **BayesCAT** software implements a joint model for co-estimating phylogeny and sequence alignment described in Shim2014. Traditionally, phylogeny and sequence alignment are estimated separately: first estimate a multiple sequence alignment and then infer a phylogeny based

on the sequence alignment estimated in the previous step. However, uncertainty in the alignment estimation is ignored, resulting, possibly, in overstated certainty in phylogeny estimates. The implemented joint model for co-estimating phylogeny and sequence alignment improves estimates from the traditional approach by accounting for uncertainty in the alignment in phylogenetic inferences.

Compared to alternative joint methods, our insertion and deletion (indel) model allows arbitrary-length overlapping indel events and a general distribution for indel fragment size. In addition, we explicitly model a completely history of indel events on the tree. Therefore, our approach enables us to infer more information about the indel process.

The implemented methods for joint estimation of phylogeny and sequence alignment

- infer phylogeny while accounting for uncertainty in the alignment
- summarize alignment samples
- infer more information about the indel process.

2 How to cite BayesCAT

Heejung Shim and Bret Larget (2014). BayesCAT: Bayesian Co-estimation of Alignment and Tree (Under review).

3 Other software packages which are required in BayesCAT

BayesCAT uses multiple software packages to summarize tree and alignment samples and check if Markov Chain converges. This section explains how to install those packages.

3.1 A software package: summarize

To summarize tree samples, we use 'summarize' software. We download summarize.tgz from BADGER's website, unzip, and compile a code:

```
mkdir tmp
cd tmp/
wget http://badger.duq.edu/download/summarize.tgz
tar -zxvf summarize.tgz
cd Summarize
```

```
make
```

Then, we can get 'summarize' binary file. We copy it to bin directory in BayesCAT package:

```
cp tmp/Summarize/summarize ~/BayesCAT/bin/
```

We can summarize tree MCMC samples using

```
~/BayesCAT/bin/summarize treesummaryfile > outputfile
```

3.2 A software package: FSA

We download the latest version of FSA from Sourceforge FSA project page, unzip, and build:

```
cd tmp/
wget http://sourceforge.net/projects/fsa/files/latest/download/fsa-1.15.9.tar.
    gz
tar -zxvf fsa-1.15.9.tar.gz
cd fsa-1.15.9
./configure
make
```

3.3 A software package: chart

To check if Markov Chain converges by using tree samples, we use 'chart' which is part of the Badger software package. We provide 'chart.C' in BayesCAT/script/C directory. We compile, and move binary file to bin directory:

```
cd ~/BayesCAT/scripts/C
g++ -l chart chart.C
mv chart ~/BayesCAT/bin/
```

4 How to run BayesCAT

If you followed the instruction in our github, you already have BayesCAT installed and have binary executable file BayesCAT in the directory 'BayesCAT/src/'. We start with making a directory to perform an analysis using a toy example.

```
cd BayesCAT
mkdir test1
cd test1
```

As a toy example, we use simulated data of five sequences which we provide in '/BayesCAT/-data/example/sequence.fasta'. We copy the toy example to the current directory and we can run **BayesCAT** on this dataset, using for example

```
cp ~/BayesCAT/data/example/sequence.fasta .
~/BayesCAT/src/BayesCAT -seqfile sequence.fasta -seed 4 -iterations 1000 -
   burnin 100 -samplingIV 10 -alpha_gamma 0.1538462 -alpha_kappa 0.5 -
   alpha_lambda 20 -alpha_A 0.90 -alpha_C 1.6 -alpha_G 0.85 -alpha_T 1.65 -
   alpha_r 2 -beta_r 98 -alpha_rd 4 -beta_rd 12
```

The sequence file containing multiple sequences as a FASTA format should be provided with the option **-seqfile**. A seed for random generators can be provided with the option **-seed**. The number of iterations for sampling and the number of iterations to be skipped as burn-in can be provided by the options **-iterations** and **-burnin**, respectively. For example, with '-iterations 1000 -burnin 100', **BayesCAT** runs total 1100 iterations and skips the first 100 iterations as a burn-in. Among 1000 iterations, we can use only a subset of samples for inference. For example, '-samplingIV 10' indicates that **BayesCAT** uses every 10th sample from 1000 iterations after burn in for inference (total 100 samples). When **-burnin** and **-samplingIV** are not provided, **BayesCAT** sets up burnin as 10% of iterations and samplingIV as min(number of iterations/1000, 1). Thus, as default, **BayesCAT** won't sample more than 1000. Prior distributions can be specified by using the options **-alpha_gamma** (prior on γ), **-alpha_-kappa** (prior on κ), **-alpha_lambda** (prior on λ), **-alpha_A**, **-alpha_C**, **-alpha_G**, **-alpha_T** (prior on π), **-alpha_r**, **-beta_r** (prior on r), and **-alpha_rd**, **-beta_rd** (prior on r_d).

5 How to summarize outputs from BayesCAT

5.1 Summarize tree samples

'REStree' contains tree samples and we can summarize them using **summarize** for example

```
~/BayesCAT/bin/summarize REStree > tree.sum
```

'tree.sum' shows posterior probabilities for tree topologies and clades. Please see manual for summarize for more options and detailed description of their output.

5.2 Summarize multiple sequence alignments

'RESAlignment' contains multiple sequence alignment samples. First, we convert multiple sequence alignment samples to pairwise alignment samples using perl script 'compute.Pair.Post.Prob.pl' provided in 'BayesCAT/scripts/perl/', for example,

```
perl ~/BayesCAT/scripts/perl/compute.Pair.Post.Prob.pl RESAlignment alignment.
    sum
```

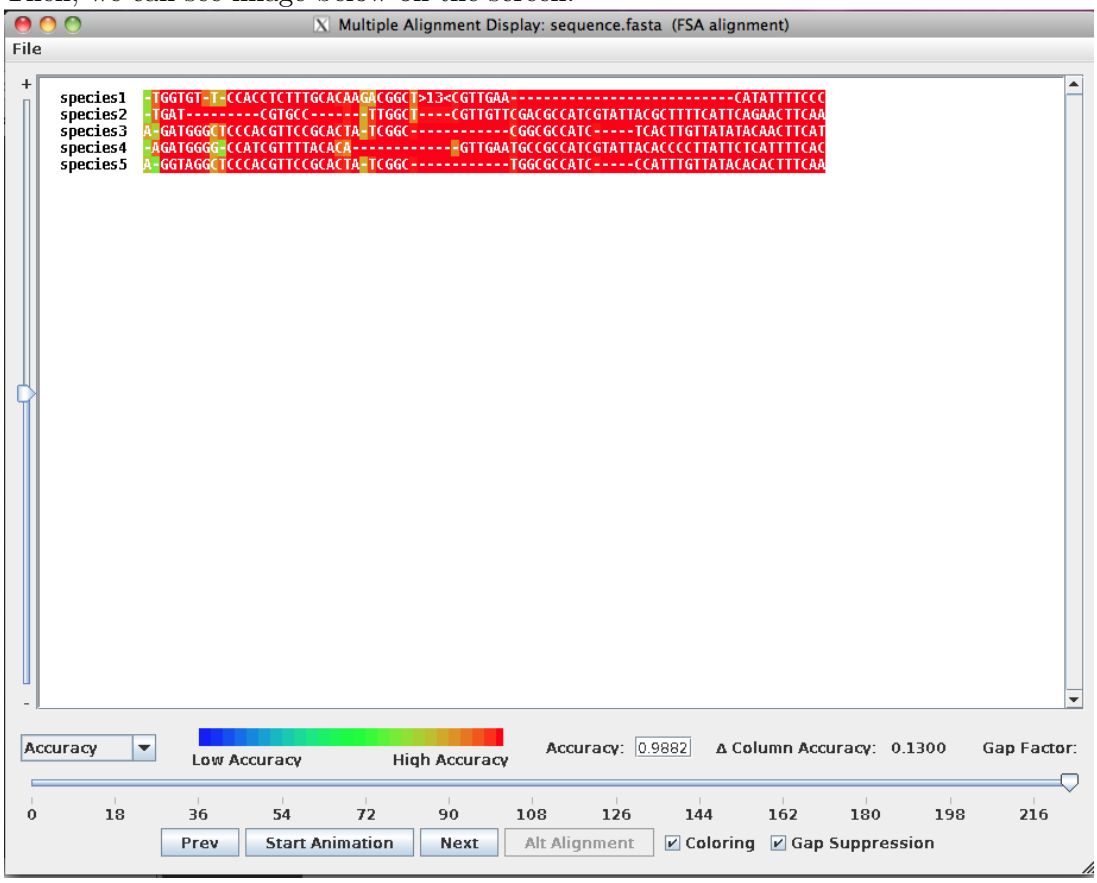
Then, 'alignment.sum' contains pairwise alignment samples. Now we can summarize/visualize multiple sequence alignment using the software package **FSA**. For example

```
~/tmp/fsa-1.15.9/src/main/fsa --gui --load-probs alignment.sum sequence.fasta
```

Then, we can see **FSA** generates two files, 'sequence.fasta.probs' and 'sequence.fasta.gui'. Now we can summarize/visualize alignment samples using for example

```
java -jar ~/tmp/fsa-1.15.9/display/mad.jar sequence.fasta
```

Then, we can see image below on the screen:



Please see FSA website for more options and detailed description of their output.

5.3 Summarize number of splits, edge length, and number of indel events

'RESmnumIinAll', 'RESmnumDinAll', and 'RESmedgeLen' contain number of insertion, number of deletion, and edge length for each (sampled) split. We can summarize them using a Script 'get.summary.numID.edgelen.R' provided in '/BayesCAT/scripts/R/' for example

```
Rscript ~/BayesCAT/scripts/R/get.summary.numID.edgelen.R RESmnumIinAll
    RESmnumDinAll RESmedgeLen splits.sum 5
```

Here, 5 is number of sequences. The output file 'split.sum' contains four columns. The first column is the split identifier. For example the identifier **11000** indicates the split which splits the first two sequences from the other three sequences. The second column contains a posterior

probability for each split. The third and fourth columns contain posterior means of the number of indel events and the edge length given occurrence of each split.

Please see the description at the beginning of the script '/BayesCAT/scripts/R/get.summary.numID.edgelen.R' for detailed explanation for usage and options.

5.4 Summarize indel fragment sizes

'RESIleninAll' and 'RESDleninAll' contain samples of insertion and deletion fragment sizes. We can summarize them using a Script 'get.summary.fragmentSize.R' provided in '/BayesCAT/scripts/R/' for example

```
Rscript ~/BayesCAT/scripts/R/get.summary.fragmentSize.R RESIleninAll
   RESDleninAll indel.len.sum
```

The output file 'indel.len.sum' contains three rows, and each row contains a posterior estimate of realized indel (in the 1st row; realized insertion in the 2nd row; realized deletion in the 3rd row) fragment size distribution.

Please see the description at the beginning of the script '/BayesCAT/scripts/R/get.summary.fragmentSize.R' for detailed explanation of usage.

5.5 Summarize parameters

'RESmGamma', 'RESmKappa', 'RESmP', 'RESmLambda', 'RESmMu', 'RESmR', 'RESmRi', and 'RESmRd' contain samples of parameters, γ, κ, π, λ, μ, r, r_i, and r_d. We can summarize them using a Script 'get.summary.param.R' provided in '/BayesCAT/scripts/R/' for example

```
Rscript ~/BayesCAT/scripts/R/get.summary.param.R RESmGamma gamma.sum 95 1
Rscript ~/BayesCAT/scripts/R/get.summary.param.R RESmP pi.sum 95 4
```

The script takes three arguments (input file, creditable interval, and number of parameters in the input file). The first output file 'gamma.sum' contains four columns: mean, median, and 95% CI for γ. The second output file 'pi.sum' contains four rows: each of rows contains mean, median, and 95% CI for each of π.

Please see the description at the beginning of the script '/BayesCAT/scripts/R/get.summary.param.R' for detailed explanation of usage.

5.6 Summarize other quantities

'RESmnumD', 'RESmnumI', and 'RESmtotalEdgeLen' contain samples for number of deletion, number of insertion, and total sum of branch lengths in a tree. We can summarize (mean, median, CI) them using a script 'get.summary.param.R'.

6 How to check if Markov Chain converges

We can run multiple Markov Chains with different seeds and check if Markov Chains converge using tree samples from multiple runs by **chart**. **chart** is part of the Badger software package and one already has a binary executable file in 'BayesCAT/bin/' directory if one followed instruction in the previous section. Suppose we run three Markov Chains with different seeds on directories 'BayesCAT/test1/', 'BayesCAT/test2/', and 'BayesCAT/test3/'. First we summarize tree samples using **summarize** for example

```
cd ~/BayesCAT
~/BayesCAT/bin/summarize test1/REStree > test1/tree.sum
~/BayesCAT/bin/summarize test2/REStree > test2/tree.sum
~/BayesCAT/bin/summarize test3/REStree > test3/tree.sum
```

Then, we can check if Markov Chain converges using those summary files for example

```
~/BayesCAT/bin/chart test1/tree.sum test2/tree.sum test3/tree.sum
```

Please see manual for chart for more options and detailed description of its output.