BUCKy

Bayesian Untangling of Concordance Knots
(applied to yeast and other organisms)

Version 1.2, 17 January 2008

Last updated: 11 November 2008

Departments of Statistics and of Botany
University of Wisconsin - Madison
Medical Sciences Center,
1300 University Ave. Madison, WI 53706, USA.

Introduction

BUCKy is a program to analyze a multi-locus data sets with Bayesian Concordance Analysis (BCA), as described in Ané *et al.* (2007). This method accounts for biological processes like hybridization, incomplete lineage sorting or lateral gene transfer, which may result in different loci to have different genealogies. With BCA, each locus is assumed to have a unique genealogy, and different loci having different genealogies. The *a priori* level of discordance among loci is controlled by one parameter α.

BCA works in two steps: First, each locus is to be analyzed separately, with MrBayes for instance. Second, all these separate analyses are brought together to inform each other. BUCKy will perform this second step. BUCKy comes into two separate programs: `mbsum` and `bucky`. The first program `mbsum` summarizes the output produced by MrBayes from the analysis of an individual locus. The latter, `bucky`, takes the summaries produced by `mbsum` and performs the second step of BCA. These two programs were kept separate because `mbsum` is typically run just once, while `bucky` might be run several times independently, with or without the same parameters.

Installation and Compilation

BUCKy is a command-line controlled program written is C++. It should be easily compiled and run on any Linux system or Mac OSX.

Installation (Mac OSX 10.4 users). Since Mac OSX 10.4 does not come with a C++ compiler, we can provide an executable file that compiled with OSX 10.4, upon request.

Installation (Linux or Mac OSX, version 10.3.9 or below). Pick a directory where you want the BUCKy code to be. This directory will be called `$BUCKY_HOME` in this documentation. Download the `bucky-1.2.tgz` file and put it in `$BUCKY_HOME`. To open the compressed tar file with the BUCKy source code and example data, do these commands:

```
cd $BUCKY_HOME
```

```
tar -xzvf bucky-1.2.tgz
```

This creates a directory named `bucky` with subdirectories `bucky/data` and `bucky/src`.

Compilation. If you have gcc installed, compile the software with these commands.

```
cd $BUCKY_HOME/bucky/src
make
```

This will compile programs `mbsum` and `bucky`. It is suggested that copies of `mbsum` and `bucky` be put in `~/bin` if this directory is in your path.

If you do not have gcc installed and the executable provided is not working on your system, you need to find the installer for gcc. On a Macintosh (version 10.3.9 or before), it may be in Applications/Installers/Developer Tools .

Running mbsum

Type this for a brief help message

```
mbsum --help
mbsum -h
```

Purpose and Output. It is advised to have one directory containing the MrBayes output of all individual locus analyses. Typically, in this directory each file of the form `*.t` is a MrBayes output file from one single locus. Use `mbsum` to summarize all files from the same locus. You want `mbsum` to create a file `<filename>.in` for each locus. The extension `.in` just means input (for later analysis by `bucky`). Output files `*.in` from `mbsum` will typically look like the following, containing a list of tree topologies and a tally representing the trees' posterior probabilities from a given locus (as obtained in the first step of BCA).

```
(1,(2,(3,(4,(5,((6,7),8))))));  24239
(1,(2,(3,(4,(5,(6,(7,8)))))));  15000
(1,(2,(3,(4,(5,((6,8),7))))));  2983
(1,(2,(3,((4,5),((6,7),8)))));  2590
(1,(2,((3,((6,7),8)),(4,5))));  2537
(1,(2,((3,(6,(7,8))),(4,5))));  1097
(1,(2,(3,((4,5),(6,(7,8))))));  995
(1,(2,(3,((4,5),((6,8),7)))));  163
(1,(2,(3,((4,((6,7),8)),5))));  145
(1,(2,((3,((6,8),7)),(4,5))));  96
(1,(2,((3,(4,5)),((6,7),8))));  66
(1,(2,(3,((4,(6,(7,8))),5))));  51
(1,(2,((3,(4,5)),(6,(7,8)))));  22
(1,(2,(3,((4,((6,8),7)),5))));  15
(1,(2,((3,(4,5)),((6,8),7))));  1
```

Syntax and Options. To run `mbsum` for a single data file, type:

```
mbsum [-h] [--help] [-n #] [-o filename] [--version] <inputfilename(s)>
```

For example, let's say an alignment `mygene.nex` was analyzed with MrBayes with two runs, and sampled trees are in files `mygene.run1.t` and `mygene.run2.t`. These two sample files include, say, 5000 burnin trees each. To summarize these 2 runs use

```
mbsum -n 5000 -o mygene mygene.run1.t mygene.run2.t
```

or more generally

```
mbsum -n 5000 -o mygene mygene.run?.t
```

Here is a description of the available options.

`-h` or `[--help]`	prints a help message describing the options then quits.
`-n #` or `[--skip #]`	This option allows the user to skips lines of trees before actually starting the tally tree topologies. The default is 0, i.e *no* tree is skipped. The same number of trees will be skipped in each input file.
`-o filename` or `--out filename`	sets the output file name. A single output file will be created even if there are multiple input files. The tally combines all trees (except skipped trees) found in all files.
`--version`	prints the program's name and version then quits.

Example: the raw data and output from MrBayes are provided for the very first gene in the set analyzed in Ané *et al.* (2007). They are located in `$BUCKY_HOME/bucky/data/yeast/y000/`. The tree files from MrBayes, named `y000.run1.t` through `y000.run4.t`, each contain 5501 trees. They can be summarized with:

```
mbsum -n 501 -o y000.in    $BUCKY_HOME/bucky/data/yeast/y000/y000.run?.t
```

Warnings. `mbsum` will overwrite a file named `filename` if such a file exists. Most importantly, `mbsum` assumes that the same translation table applies to all files, i.e. that taxon 1 is the same taxon across all genes, and taxon 2 is the same taxon across all genes, etc. This is okay as long as taxa appear in the same order in all alignment files. But if not, BCA would be screwed up with no warning (Ané *et al.*, 2007). The best way to avoid such a problem is to have all the alignments in a single nexus file. In the first step (single locus) analysis, MrBayes can be told to ignore all but a single locus, and this would be repeated for each locus.

Running bucky

To run `bucky`, type

```
bucky [-options] <summary_files>
```

For example, after creating all `.in` files with `mbsum` in the same directory, you can run bucky with the default parameters by typing this:

```
bucky *.in
```

Options.

`-o output-file-root`	This option sets the names of output files. Default is `run1`.
`-a alpha`	α is the *a priori* level of discordance among loci. Default α is 1.
`-n number-generations`	Use this option to increase the number of updates (default: 100,000). An extra number of updates will actually be performed for burnin. This number will be 10% of the desired number `n` of post-burning updates. The default, then, is to perform 10,000 updates for burnin, which will be discarded, and then 100,000 more updates.
`-h` or `--help`	Prints a help message describing options, then quits.
`-k number-runs`	Runs k independent analyses. Default is 2.
`-c number-chains`	Use this option to run Metropolis coupled MCMC (or MCMCMC), whereby hot chains are run in addition to the standard (cold) chain. These chains occasionally swap states, so as to improve their mixing. The option sets the total number of chains, including the cold one. Default is 1, i.e. no heated chains.
`-r MCMCMC-rate`	When Metropolis coupled MCMC is used, this option controls the rate r with which chains try to swap states: a swap is proposed once every r updates. Default is 100.
`-m alpha-multiplier`	Warm and hot chains, in MCMCMC, use higher values of α than does the cold chain. The cold chain uses the α value given by the option `-a`. Warmer chains will use parameters $m\alpha, m^2\alpha, \ldots, m^{c-1}\alpha$. Default m is 10. The independence prior corresponds to $\alpha = \infty$ so MCMCMC is not used with this prior.
`-s subsample-rate`	Use this option for thinning the sample. All post-burnin samples will be used for summarizing the posterior distribution of gene-to-tree maps, but you may choose to save just a subsample of these gene-to-tree maps. One sample will be saved every s updates. This option will have an effect only if option `--create-sample-file` is chosen. Default is 1: no thinning.
`-s1 seed1`	Default is 1234.
`-s2 seed2`	Default is 5678.

`--calculate-pairs`	Use this option to calculate the posterior probability that pairs of loci share the same tree. Default is to NOT use this option.
`--create-sample-file`	Use this option for saving samples of gene-to-tree maps. Default is to NOT use this option: samples are not saved. Saving all samples can slow down the program.
`--create-joint-file`	This option creates a `.joint` file. NOT created by default.
`--create-single-file`	This option creates a `.single` file. NOT created by default.
`--use-independence-prior`	Use this option to assume *a priori* that loci choose their trees independently of each other. This is equivalent to setting $\alpha = \infty$. Default is to NOT use this option.
`--use-update-groups`	Use this option to permit all loci in a group to be updated to another tree. Default is to use this option, because it improves mixing.
`--do-not-use-update-groups`	Use this option to disable the update that changes the tree of all loci in a group in a single update. Default is to NOT use this option. If both options `--use-update-groups` and `--do-not-use-update-groups` are used, only the last one is applied. No warning is given, but the file `.out` indicates whether group updates were enabled or disabled.

Output. Running `bucky` will create various output files. With defaults parameters, these output files will have names of the form `run1.*`, but you can choose you own root file name. The following output files describe the input data, input parameters, and progress history.

`.out`	Gives the date, version (1.2), input file names, parameters used, running time and progress history. If MCMCMC is used, this file will also indicate the acceptance history of swaps between chains.
`.input`	Gives the list of input files. There should be one file per locus.
`.single`	Gives a table with tree topologies in rows and loci in columns. The entries in the table are posterior probabilities of trees from the separate locus analyses. It is a one-file summary of the first step of BCA.

The following files give the full results as well as various result summaries. The goal of BCA this is to estimate the primary concordance tree. This tree is formed by all clades with concordance factors (CF) greater than 50%, and possibly other clades. The CF of a clade is the proportion of loci that have the clade. Sample-wide refers to loci in the sample and genome-wide refers to loci in the entire genome.

`.concordance`	Main output: this file first gives the primary concordance tree topology in parenthetical format and again the same tree with the posterior means of sample-wide CFs as edge lengths. This concordance tree is currently fully resolved, possibly including clades that are in less than 50% of gene trees. The user might want to unresolve those clades in case the conflicting clades have lower but similar concordance factors. The list of clades in the primary concordance tree follows, with information on their sample-wide and genome-wide CFs: posterior mean and 95% credibility intervals. Inference on genome-wide CFs assumes that loci were sampled at random from an infinite genome. Finally, the file gives the posterior distribution of sample-wide CFs of all clades, sorted by their mean CF. In this list however, CFs are expressed in number of loci instead of proportions.
`.cluster`	Gives the posterior distribution of the number of clusters, as well as credibility intervals. A cluster is a group of loci sharing the same tree topology. Loci in different clusters have different tree topologies.
`.pairs`	Gives an l by l similarity matrix, l being the number of loci. Entries are the posterior probability that two given loci share the same tree.
`.gene`	For each locus, gives the list of all topologies supported by the locus (index and parenthetical description). For each topology is indicated the posterior probability that the locus has this tree given the locus's data ('single' column) and given all loci's data ('joint' column).
`.sample`	Gives the list of gene-to-tree maps sampled by `bucky`. With n post-burnin updates and subsampling every s steps, this file contains n/s lines, one for each saved sample. Each line contains the number of accepted updates (to be compared to the number of genes * sub-sampling rate), the number of clusters in the gene-to-tree map (loci mapped to the same tree topology are in the same cluster), the log-posterior probability of the gene-to-tree map up to an additive constant followed by the gene-to-tree map. If there are l loci, this map is just a list of l trees. Trees are given by their indices. The correspondance between tree index and tree parenthetical description can be found in the `.gene` or `.single` file.
`.joint`	Gives a table with topologies in rows and loci in columns, similar to file `.single` file. Topologies are named by their indices as well as by their parenthetical descriptions. Entries are posterior probabilities (averaged across all runs) that each locus was mapped to each topology.

Examples

The example data provided with the program is organized as follows: directory `$BUCKY_HOME/bucky/data/example1/` contains 10 folders named `ex0` to `ex9`, one for each locus. These 10 folders contain a single file each, named `ex.in`, which was created by `mbsum`. For analyzing these data, one can use the default parameters and type

```
bucky $BUCKY_HOME/bucky/data/example1/ex?/ex.in
```

The question mark will match any character (any digit 0 to 9 in particular), so that all 10 locus files will be used for the analysis. The following command will run `bucky` with $\alpha = 5$, no MCMCMC, group updates disabled, 2 independent runs (default), one million updates and user-defined seeds (keep this command on a single line).

```
bucky -n 1000000 -a 5 -s1 745203 -s2 905423 --do-not-use-update-groups
                                        $BUCKY_HOME/bucky/data/example1/ex?/ex.in
```

A look at the file `run1.concordance` shows that the clades (19,20) and (18,20) both have an estimated CF of 0.50 but that this estimate differed greatly between runs because its standard deviation is 0.707. Scrolling down the file indicates that the first run gave a 100% concordance factor to clade (18,20) all the time while the second run gave it a 0% concordance factor all the time. So the two runs are in very strong disagreement. This poor mixing is fixed by using the option `--use-update-groups` (or by not using the `--do-not-use-update-groups` option!).

The yeast data analyzed in Ané *et al.* (2007) is provided with the program and organized as follows. The directory `$BUCKY_HOME/bucky/data/yeast/` contains 106 folders named `y000` to `y105`, one for each gene. In each of these folders, a file created by `mbsum` and named `run2.nex.in` contains the data from one gene. For analyzing these data with $\alpha = 2.5$, $n = 1,000,000$ updates, $k = 4$ independent runs, $c = 4$ chains (one cold and 3 hot chains), saving samples once every 1000 updates, and for keeping a similarity matrix among genes, one would type (on a single line)

```
bucky -a 2.5 -n 1000000 -k 4 -c 4 --create-sample-file --calculate-pairs
                            $BUCKY_HOME/bucky/data/yeast/y???/run2.nex.in
```

Version history

Changes to version 1.1. The main output file (`.concordance`) contains the primary concordance tree in parenthetical format. It also displays a more detailed summary for all splits with mean concordance factor above 0.10. A bug was fixed in the list of splits belonging to the primary concordance tree. Inference on genome-wide concordance factors is included. The help message is improved with a better display of available options and default parameter values.

The following output files, deemed unnecessary, are no longer produced: `.gene`, `.top`, `.topologies` and `.splits`. Output file named `.genepost` in version 1.1 is now named `.gene` in version 1.2. Output files `.joint` and `.single` are not produced unless requested by the user.

Changes to version 1.2. Independent runs are implemented, with information on the standard deviation of clade's CF across runs. A bug was fixed with the group update.

General notes

There is a limit to 32 taxa.

Choosing the *a priori* level of discordance α. To select a value based on biological relevance, the number of taxa and number of genes need to be considered. For example, the user might have an *a priori* for the proportion of loci sharing the same genealogy. One can turn this information into a value of α since the probability that two randomly chosen loci share the same tree is about $1/(1+\alpha)$ if α is small compared to the total number of possible tree topologies. Also, the value of α sets the prior distribution on the number of distinct locus genealogies in the sample. Go to the website `www.stat.wisc.edu/~larget/bucky.html` for an interactive display of this distribution, which can serve as a tool for the choice of α.

First step of BCA: individual locus analysis. Any model of sequence evolution can be selected for any locus: there need not be one model common to all loci. Some loci can be protein alignments, others DNA alignments, and other morphological characters.

Missing sequences. If some loci have missing sequences, i.e. missing taxa, then rows of missing data (????) need to be inserted in place of the missing taxon's sequence. A more efficient way to deal with missing sequences will be implemented in a future version of `bucky`.

References

Ané, C., B. Larget, D. A. Baum, S. D. Smith, and A. Rokas. 2007. Bayesian estimation of concordance among gene trees. Molecular Biology and Evolution **24**:412–426.

Ané, C., B. Larget, D. A. Baum, S. D. Smith, and A. Rokas. 2007. Erratum for: Bayesian estimation of concordance among gene trees. Molecular Biology and Evolution **24**:1575.