Phylogenetic Analysis of Molecular Data (Botany 563)

Computer Lab 11: Bayesian Concordance Analysis using BUCKy

References:
1. BUCKy manual, at http://www.stat.wisc.edu/~ane/bucky/index.html
2. Ané C., B. Larget, D.A. Baum, S.D. Smith, A. Rokas (2007). Bayesian estimation of concordance among gene trees. Molecular Biology and Evolution 24(2), 412-426.
3. Rokas, A., B. L. Williams, N. King and S. B. Carroll (2003). Genome-scale approaches to resolving incongruence in molecular phylogenies. Nature 425: 798-804.

Abbreviations: BCA=Bayesian Concordance Analysis; CF=concordance factor

Learning objective:

Get acquainted with Bayesian Concordance Analysis, as implemented in BUCKy.
Familiarize yourself with the program FigTree for tree visualization and manipulation.

Dataset: 106 genes for 8 species of yeast
The yeast data that we will analyze is provided with the program BUCKy and organized as follows. The directory **bucky/data/yeast/** contains 106 folders named **y000** to **y105**, one for each gene. In each of these folders, there is a nexus file with the data for one gene (run2.nex), and a .t file that was generated by MrBayes for that gene (run2.nex.t). The eight taxa used are:

```
Taxon 1 -> Scer
Taxon 2 -> Spar
Taxon 3 -> Smik
Taxon 4 -> Skud
Taxon 5 -> Sbay
Taxon 6 -> Scas
Taxon 7 -> Sklu
Taxon 8 -> Calb
```

About BCA and BUCKy:

Bayesian Concordance Analysis (BCA), as described in Ané et al. (2007) is implemented in the package BUCKy. By allowing for the possibility that each locus has tracked its own history, BCA allows to identify biological processes that may account for different loci to have different genealogies, like hybridization, incomplete lineage sorting, or lateral gene transfer. The a priori level of discordance among loci is controlled by a single parameter (alpha: α). Crudely, BCA is a three-step process, which involves three programs, two of which are part of the package BUCKy:
1. Bayesian Analysis of individual loci —> program: MrBayes
2. Summary of the **.t** files produced from MrBayes ––> program: mbsum
3. Running last step ––> program: bucky

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. An interactive display to get this distribution, which can serve as a tool for the choice of α, can be accessed at http://bigfork.botany.wisc.edu/concordance/

Running mbsum

Typically, in this directory each file of the form *.t is an output file from a single locus analysis in MrBayes. All files from the same locus can be summarized by mbsum; it will create a file with the extension .in (this just means input for later analysis by bucky). Output files from mbsum will typically contain 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). An output file from mbsum will look like this:

```
translate
1 Scer,
2 Spar,
3 Smik,
4 Skud,
5 Sbay,
6 Scas,
7 Sklu,
8 Calb;
(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
```

Running bucky

After input files created by mbsum are ready, the names of these files can be given as arguments to bucky (the file names can be written into a file, which in turn can be given to bucky). For example, after creating all the **.in** files with mbsum in the same directory, you can run bucky with the default parameters by typing this: ./bucky *.in

The options for bucky are the following:

```
Options:
Parameter                | Usage                         | Default Value
--------------------------------------------------------------------------
alpha                    | -a number                     | 1
# of runs                | -k integer                    | 2
# of MCMC updates        | -n integer                    | 100000
# of chains              | -c integer                    | 1
MCMCMC Rate              | -r integer                    | 100
alpha multiplier         | -m number                     | 10
subsample rate           | -s integer                    | 1
output root file name    | -o name                       | run1
input file list file     | -i filename                   |
random seed 1            | -s1 integer                   | 1234
random seed 2            | -s2 integer                   | 5678
create sample file       | --create-sample-file          | false
create joint file        | --create-joint-file           | false
create single file       | --create-single-file          | false
use independence prior   | --use-independence-prior      | false
calculate pairs          | --calculate-pairs             | false
use update groups        | --use-update-groups           | true
use update groups        | --do-not-use-update-groups    |
help                     | -h OR --help                  |
version                  | --version                     |
--------------------------------------------------------------------------
```

For a description of each option refer to the BUCKy manual. **Note:** There is a limit to 32 taxa for bucky.

Output from bucky

Running bucky will create various output files. With default parameters, these files will have the prefix "run1". The output files are the following:

.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 postburnin 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 correspondence 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 the .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.

<u>Task 1</u>. Prior distribution on number of distinct trees as a function of alpha

Go to http://bigfork.botany.wisc.edu/concordance/ and explore how the expected number of distinct trees changes for our 106 genes and 8 taxa given different values of alpha.

How many distinct trees should we expect for:
alpha = 0.001? _________
alpha = 1? _________
alpha = 10? _________
alpha = 1000? _________

<u>Task 2</u>. Running mbsum

mbsum is run in the terminal. To open the terminal: go to Applications/Utilities/Terminal

To change your working location to the bucky folder, <u>TYPE</u>: cd desktop/bucky

Goal: generate one input file for each .t file and store it in its corresponding folder, so that in the end we will have 106 folders each with: a) run2.nex, b) run2.nex.t, and c) run2.nex.in

We could run mbsum on each file individually by typing:

```
src/mbsum -n 5000 -o data/yeast2/y001/run2.nex.in data/yeast2/y001/run2.nex.t
```

What is our burnin specified by this command? __________

But we would need to run mbsum 106 times!!!

To automate this process, we will take advantage of Perl, and use the script in the file mbsum100.pl

All we need to do to generate the 106 run2.nex.in files is **<u>TYPE</u>**: perl mbsum100.pl (Perl is good! ☺)

Check that the desired .in files were generated.

<u>Task3</u>. Running bucky

Bucky is also run from the Terminal.

Check that your working directory is desktop/bucky by **<u>TYPING</u>** pwd

To run bucky on the 106 files provided in the yeast folder, simply <u>TYPE</u> (all in one line):

```
src/bucky -a 1.0 -n 150000 -k 4 -c 3 --create-sample-file --create-joint-file --calculate-pairs -s
1000 data/yeast2/y???/run2.nex.in
```

Refer to the options table provided in a previous section to answer the following:
Q3.1) What is the value of the alpha parameter being used? _________
Q3.2) How many MCMCMC chains are we running? _________

Task 4. Get acquainted with the different output files

To answer the following questions, you will need to look at the different output files generated by bucky. You can open and edit these files using TextEdit.

Q4.1) How many topologies were sampled for gene 57? ____; and for gene 104? ____ (Hint: .gene file)

Q4.2) How many different clusters were identified in this analysis? (Hint: .cluster file) ________

Q4.3) Look at the .concordance file. Are there any splits NOT in the Primary Concordance Tree but with estimated CF>0.05?________

How many? _________

Write such splits, with their respective mean CFs and the number of genes that, on average, yielded each of these splits.

Q4.4) Look for a topology that has the clade (6,7) and give the numbers of two genes that support it. (Hint: .joint file)

Task 5. Visualizing the Primary Concordance Tree

5.1) Open the .concordance file. Look for the "Primary Concordance Tree with Sample Concordance Factors", and paste it in the appropriate place in the file bcayeast.tree .

5.2) Open the file bcayeast.tree in the program FigTree.
CFs in the .concordance file are depicted as branch lengths, so to visualize them you need to check the box "branch lengths" (make sure that the option "branch lengths raw" is selected from the drop-down menu). What do the depicted values represent? How would you get the mean CFs?

FigTree is a nice tree visualization and manipulation program. Feel free to explore the different options.

Task 6. Does alpha affect the CFs for this dataset?

Re-run bucky, but now with an alpha of infinity. For this, you need to TYPE:

```
src/bucky --use-independence-prior -n 150000 -k 4 -c 3 --create-sample-file --create-joint-file --
calculate-pairs -s 1000 data/yeast2/y???/run2.nex.in
```

Did you find any difference in the CFs under alpha of infinity?