A BUCKy Tutorial

Bret Larget
Departments of Botany
and of Statistics,
UW-Madison

Workshop on New Methods
for Phylogenomics and Metagenomics

What Does BUCKy Intend to do?

- BUCKy was conceived as a program to answer the question, what fractions of genomes (or genes within genomes) share the same evolutionary history?
- The specific way BUCKy is set to address this question is to jointly estimate many gene trees given data for each gene and prior information about the level of gene tree discordance.

What is a GTM?

- GTM is an acronym for a Gene-to-Tree-Map which can be represented as an array of tree topologies, one for each gene.

$$M = (T_1, T_2, \ldots, T_G)$$

where M is the map, T_i is the tree topology for the ith gene, and G is the number of genes.

Joint Posterior Distribution

- For data sets $X_1, X_2, \ldots, X_G$, one for each gene, BUCKy tries to compute $P(M | X_1, X_2, \ldots, X_G)$, the joint posterior distribution of the tree topology for all the genes given the data on all of the genes.

Form of Posterior Distribution

- Under assumptions of independence of parameters other than topology across different genes, this posterior has the form:

$$P(M|X) \propto P(M) \times \prod_i P(T_i \mid X_i)$$

for $M = (T_1, T_2, \ldots, T_G)$ and $X = (X_1, X_2, \ldots, X_G)$

Approximation

- BUCKy approximates the exact product of posterior probabilities of trees given single data sets with simple relative frequencies from MCMC samples on single genes.
- As a consequence, BUCKy can mislead when single gene posterior distributions are inaccurate.

Preliminary: Installation

If you:

(1) Have gcc installed on your computer;

(2) Are using Linux or the Terminal on a Mac;

(3) Have a directory ~/bin which is part of your path of executable files;

Then, you can compile and install bucky and mbsum with these steps.

In a terminal:

(1) Change directories to where the file bucky-1.4.2.tgz exists.

(2) Unzip and untar the file, creating a directory tree and files

```
tar zxf bucky-1.4.2.tgz
```

(3) Change to the source directory and compile the code.

```
cd bucky-1.4.2/src
make
```

(4) Move executables to ~/bin.

```
mv mbsum bucky ~/bin
```

Preliminary: Installation

Or, download a previously compiled binary from:

http://www.stat.wisc.edu/~ane/bucky .

Preliminary: Single Gene Samples

- Use MrBayes for each gene individually.
- You can use different parameters and models for each.
- BUCKy assumes that each gene has data for exactly the same species. More on dealing with this later if it is not true!
- We need the .t files for each gene. It is okay if there are more than one.

Preliminary: mbsum

- mbsum is a simple program that reads in one or more output .t files from MrBayes and creates a file with two parts:
    - a translate section which gives a list of species names and the number code for this species in trees:
    - a list of tree indices, trees, and their counts.

Example: mbsum output

```
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))))))); 31366
(1,(2,(3,(4,(5,((6,7),8)))))); 10461
(1,(2,(3,(4,(5,((6,8),7)))))); 7279
(1,(2,(3,((4,5),(6,(7,8)))))); 448
(1,(2,(3,((4,5),((6,7),8))))); 236
(1,(2,(3,((4,5),((6,8),7))))); 101
(1,(2,((3,(6,(7,8))),(4,5)))); 41
(1,(2,((3,((6,7),8)),(4,5)))); 37
(1,(2,(3,((4,(6,(7,8))),5)))); 12
(1,(2,((3,((6,8),7)),(4,5)))); 8
(1,(2,(3,((4,((6,8),7)),5)))); 7
(1,(2,((3,8),(4,(5,(6,7)))))); 2
(1,(2,(3,((4,((6,7),8)),5)))); 2
```

Example: Running mbsum

- Change directory to bucky-tutorial/TreeFiles
- Run mbsum on the .t files here, removing the first 501 trees from each.
- Save the output in a new file named y000.in.

```
mbsum -n 501 -o y000.in y000.run*.t
```

Input Files

- There should be a single input file for each gene.
- For this tutorial, input files are in the directory bucky-tutorial/InFiles .

Options for BUCKy

- BUCKy is run from the command line.
- The program is usually called with multiple options.
- The program is called as follows.

```
mbsum [options] [gene files]
```

Running BUCKy

```
cd bucky-tutorial
bucky -a 1 -k 4 -n 1000000 -c 4 -s1 23546 -s2 4564 -o yeast InFiles/*.in
```

1. `-a 1` sets alpha to 1
2. `-k 4` sets 4 separate runs
3. `-n 1000000` sets that many MCMC generations
4. `-c 4` sets 4 chains, one cold and three hot
5. `-s1 23546 -s2 4564` sets random seeds
6. `-o yeast` sets the root name for output files

More on BUCKy options

- -a alpha --- set alpha parameter
    - alpha = 0 is equivalent to disallowing discordance among gene trees
    - alpha = infinity is equivalent to independence among genes
    - the probability that two specific genes share the same tree is about $1/(1+\alpha)$ if α is much smaller than the size of tree space
    - use tool from BUCKy web page to visualize prior distribution on number of clusters

More on BUCKy options

- -k number --- sets number of chains
  - good to do more than one to informally check convergence
- -n number --- sets number of MCMC updates
  - Do enough for thorough mixing (millions?)
  - 10% extra automatic for burn-in

More on BUCKy Options

- -c number --- number of chains
- -r number --- rate to swap chains
- -m number ---multiplier for hot chains
  - For each run, bucky will run one or more chains.
  - Additional chains are run with a larger alpha.
  - At specified rate, BUCKy tries to swap chains
  - This can help mixing when mixing with desired alpha is too slow.

BUCKy Output Files

- BUCKy produces these files
  - .out --- screen output and other information
  - .input --- list of input files (one for each gene)
  - .gene --- summary of information for each gene
  - .cluster --- summary of the number of clusters (different trees)
  - .concordance --- summary of concordance among gene trees

.gene File

- Separate entry for each gene
- Shows trees for each gene
- Single is the probability of the tree given only the data in the gene
- Joint is the probability of the tree given the data in all of the genes (for specified prior concordance)

```
Gene 0:
  numTrees = 13
  index topology                              single     joint
      0 ((((((1,2),3),4),5),6),7,8);  0.627320  0.999783
      1 ((((((1,2),3),4),5),7),6,8);  0.145580  0.000148
      2 (((((1,2),3),(4,5)),6),7,8);  0.008960  0.000000
      3 ((((((1,2),3),5),4),6),7,8);  0.000240  0.000000
      4 (((((1,2),3),4),5),(6,7),8);  0.209220  0.000003
      5 (((((1,2),(4,5)),3),6),7,8);  0.000820  0.000000
      6 ((((((1,2),3),5),4),7),6,8);  0.000140  0.000000
      9 ((((1,2),(4,5)),3),(6,7),8);  0.000740  0.000066
     10 (((((1,2),3),5),4),(6,7),8);  0.000040  0.000000
     12 (((((1,2),3),(4,5)),7),6,8);  0.002020  0.000000
     13 (((((1,2),(4,5)),3),7),6,8);  0.000160  0.000000
     14 ((((1,2),3),(4,5)),(6,7),8);  0.004720  0.000000
     15 (((1,2),(4,(5,(6,7)))),3,8);  0.000040  0.000000
```

.cluster

- Summarizes distribution of number of clusters for each run

```
mean #groups = 2.024
SD across runs = 0.006

credible regions for # of groups
probability region
------------------
  0.99      (2,3)
  0.95      (2,2)
  0.90      (2,2)
------------------

Distribution of cluster number in run 1:
 # of    raw    posterior
groups  counts probability
--------------------------
  2     982478  0.98247800
  3      17521  0.01752100
  4          1  0.00000100
--------------------------

Distribution of cluster number in run 2:
 # of    raw    posterior
groups  counts probability
--------------------------
  2     972060  0.97206000
  3      27788  0.02778800
  4        152  0.00015200
--------------------------
```

.concordance

```
Splits in the Primary Concordance Tree: sample-wide and genome-wide mean CF (95% credibility), SD of
mean sample-wide CF across runs
{1,2,3,4,5|6,7,8} 1.000(1.000,1.000) 0.991(0.966,1.000)     0.000
{1,2|3,4,5,6,7,8} 1.000(1.000,1.000) 0.991(0.967,1.000)     0.000
{1,2,3|4,5,6,7,8} 0.941(0.906,0.962) 0.933(0.869,0.978)     0.000
{1,2,3,4|5,6,7,8} 0.941(0.906,0.962) 0.932(0.868,0.978)     0.000
{1,2,3,4,5,6|7,8} 0.941(0.906,0.962) 0.933(0.867,0.978)     0.000

Splits NOT in the Primary Concordance Tree but with estimated CF > 0.050:
{1,2,3,6,7,8|4,5} 0.059(0.038,0.094) 0.059(0.017,0.121)     0.000
{1,2,4,5|3,6,7,8} 0.059(0.038,0.085) 0.058(0.016,0.119)     0.000
{1,2,3,4,5,8|6,7} 0.059(0.038,0.085) 0.059(0.017,0.120)     0.000
```

.concordance

```
All Splits:
{1,2,3,4,5|6,7,8}
#Genes      count in run(s) 1 through 4, Overall probability, Overall cumulative probability
   103          0          0         58          0     0.000015    0.000015
   104          0          1          7         87     0.000024    0.000038
   105        117        129        161        165     0.000143    0.000181
   106     999883     999870     999774     999748     0.999819    1.000000

mean CF =   1.000 (proportion of loci)
        = 106.000 (number of loci)
99% CI for CF = (106,106)
95% CI for CF = (106,106)
90% CI for CF = (106,106)
```

.concordance

```
{1,2,3|4,5,6,7,8}
#Genes      count in run(s) 1 through 4, Overall probability, Overall cumulative probability
    86          0          0          5          0     0.000001    0.000001
    87          0          0         12          0     0.000003    0.000004
    88          0          0          6          1     0.000002    0.000006
    89          3          1         12         23     0.000010    0.000016
    90          9          2         10         27     0.000012    0.000028
    91         52         14         22         46     0.000034    0.000061
    92        100         81         49         98     0.000082    0.000143
    93        351        320        323        268     0.000316    0.000459
    94       1399       1365       1412       1091     0.001317    0.001775
    95       5287       5239       5273       4843     0.005161    0.006936
    96      19119      17688      18873      16846     0.018131    0.025067
    97      53703      51645      52046      49587     0.051745    0.076813
    98     120841     120342     119981     118385     0.119887    0.196700
    99     205640     212025     208334     207422     0.208355    0.405055
   100     255104     258592     260203     261258     0.258789    0.663844
   101     212900     209695     208838     215144     0.211644    0.875489
   102     101661      99021      99904     102864     0.100862    0.976351
   103      23363      23167      24143      21560     0.023058    0.999409
   104        468        803        554        537     0.000590    1.000000

mean CF =   0.941 (proportion of loci)
        =  99.772 (number of loci)
99% CI for CF = (95,103)
95% CI for CF = (96,102)
90% CI for CF = (97,102)
```

Cautions

- BUCKy assumes that the single gene posterior distributions are estimated perfectly by the samples; if a gene has mostly trees with very low sample counts, BUCKy will be misleading.
- Be extra careful if there are many taxa.

Cautions

- BUCKy assumes discordant trees are randomly drawn from all possible trees.
- Real mechanisms that cause discordance (hybridization, lateral gene flow, incomplete lineage sorting) result in trees that share many clades.
- BUCKy may underestimate true discordance, especially when tree space is large.