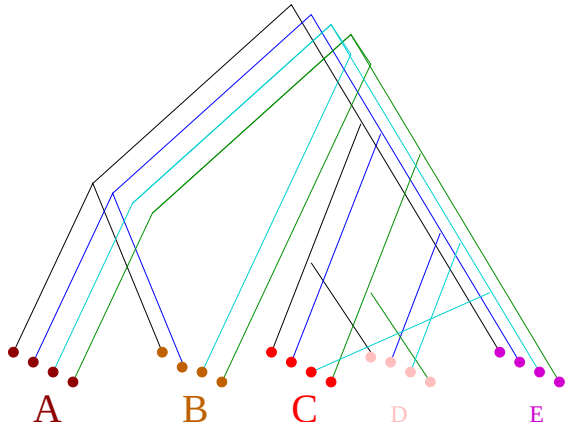
Tutorial:
Bayesian Concordance Analysis (BCA) using BUCKy



Cécile Ané

UW - Madison
Departments of Statistics and of Botany

NESCent, July 2014

Installation

Download: `http://www.stat.wisc.edu/~ane/bucky`
You can compile and install bucky if you

- have `gcc` installed on your computer,
- use Linux or Mac on the terminal,
- have a directory `~/bin` in your path of executable files

In a terminal:

1. change to the directory where `bucky-1.4.3.tgz` was placed,
2. unzip and untar the file, creating a directory tree and files:

```
tar xzf bucky-1.4.3.tgz
```

3. change to the source directory, compile the code, then move the executables to `~/bin` :

```
cd bucky-1.4.3/src/
make
mv mbsum bucky ~/bin/
```

Step 1: single-gene analyses

Use MrBayes to obtain the posterior probability of each topology τ_j for each locus i individually: $\mathbb{P}(\tau_j|D_i)$.

- You can use different parameters and models for each locus. In particular, coding genes could be analyzed on their AA alignments, non-coding regions could be analyzed on their DNA alignments.
- chloroplast DNA: should be analyzed as a single locus. Partitioned analysis to account for rate heterogeneity among genes, different substitution models among genes, etc., but a single tree.
- mitochondrial DNA: idem.

Step 1: single-gene analyses

Use MrBayes to obtain the posterior probability of each topology τ_j for each locus i individually: $\mathbb{P}(\tau_j|D_i)$.

- BUCKy assumes all genes share the same set of OTUs. (a version is available without this requirement, but is too memory-greedy at this point).
- Need: tree (.t) files. Consensus trees are not enough!
- Run `mbsum` to summarize the tree sample for each locus: to combine all runs from the same locus and calculate $\mathbb{P}(\tau|D_i)$ for all trees τ appearing at least once for that locus i.

Step 1: single-gene analyses

4. Download tutorial files and locate nexus files:

```
tar xzf buckyTutorial.tgz
cd buckyTutorial
ls ebersberger100nexus/
more ebersberger100nexus/9192.nex
```

5. Example: MrBayes analysis of locus `13830`

```
ls 13830/
more 13830/13830.nex
mb 13830/13830.nex
more 13830/13830.run1.t
mbsum -n 1001 -o 13830/13830.in 13830/13830.run?.t
more 13830/13830.in
```

MrBayes command file `13830/13830.nex`:

```
#nexus
begin mrbayes;
set autoclose=yes nowarn=yes;
execute ebersberger100nexus/13830.nex;

lset nst=2 rates=gamma;
prset brlenspr=Unconstrained:Exp(50.0);
mcmc nruns=2 temp=0.2 ngen=110000 burninfrac=0.0909
 Nchains=4 samplefreq=10 swapfreq=10 printfreq=50000
 mcmcdiagn=yes diagnfreq=50000 filename=13830/13830;
quit;
end;
```

Tree files from MrBayes in `13830/13830.run1.t`:

```
#NEXUS
[ID: 7867740670]
[Param: tree]
begin trees;
 translate
    1 human,
    2 chimp,
    3 gorilla,
    4 orang,
    5 rhesus;
 tree gen.0 = [&U] (4:1.0e-01,(2:1.0e-1,(5:1.0e-1,3:1.0e-01):1.0e-1):
                   1.0e-1,1:1.0e-1);
 tree gen.10 = [&U] ((4:4.4e-02,(5:4.4e-2,3:4.4e-2):4.4e-2):2.3e-2,2:
                    4.4e-2,1:3.2e-2);
 ...
 tree gen.109990 = [&U] ((2:6.3e-3,3:2.1e-2):1.4e-3,(4:6.4e-3,5:3.7e-2):
                        3.9e-3,1:8.7e-3);
 tree gen.110000 = [&U] ((2:2.6e-3,3:2.1e-2):1.4e-3,(4:4.4e-3,5:3.7e-2):
                        5.9e-3,1:8.7e-3);
end;
```

file with $\mathbb{P}(\tau|\text{gene } i)$ values from `mbsum`, to serve as input for `bucky` (in `13830/13830.in`:

```
translate
     1 human,
     2 chimp,
     3 gorilla,
     4 orang,
     5 rhesus;
((1,(2,3)),4,5); 16044
(((1,2),3),4,5); 3667
(((1,3),2),4,5); 255
((1,4),(2,3),5); 11
(1,((2,3),4),5); 10
(((1,2),4),3,5); 7
((1,2),(3,4),5); 3
(1,(2,(3,4)),5); 2
(1,((2,4),3),5); 1
```

Total sampled trees: 20,000. Best supported tree: `((1,(2,3)),4,5)` with posterior probability $16044/20000 = 0.8022$.

Step 1: single-gene analyses

Challenge: repeat 100's or 1000's MrBayes analyses + mbsum!

6. Create a directory where all the MrBayes results will go, then run the perl script `mb_analysis.pl`

```
mkdir ebersberger100mb
nohup perl mb_analysis.pl geneRowNb=1  Ngenes=10 fullrun &
nohup perl mb_analysis.pl geneRowNb=11 Ngenes=30 fullrun &
nohup perl mb_analysis.pl geneRowNb=41 Ngenes=30 fullrun &
nohup perl mb_analysis.pl geneRowNb=71 Ngenes=30 fullrun &
```

7. Check out results and log files with average SD of split frequencies:

```
ls ebersberger100mb/
ls ebersberger100mb/*.in
more ebersberger100mb/13986.in
more 10759to13986.log
```

Note: results from 2 separate random sets of 100 loci are available in `ebersberger100in/` and `ebersbergerAnother100in/`

log file from the analysis of the first 10 genes:

```
MrBayes estimation parameters:
 temperature      = 0.2
 nst              = 2
 rates            = gamma
 nb runs          = 2
 nb generations   = 110000
 sampling freq.   = 10
 mbsum burnin     = 1001
 mb    burnin     = 10000
 print frequency (.log) = 50000
 diagnostic freq (.mcmc)= 50000

MrBayes run: gene name, row, average SD of split freq:
        10759   1...    0.000848
        10809   2...    0.006081
        11959   3...    0.000601
        12293   4...    0.000071
        12562   5...    0.000247
        12565   6...    0.004525
        13676   7...    0.004850
        1381    8...    0.000990
        13830   9...    0.003724
        13986   10...   0.005798
Thu May 30 16:19:33 2013 -- It took 6 minute(s) and 13 second(s).
```

Step 2: Combining all genes for concordance analysis

8. Check `bucky`, get help about options, check with a short run:

```
bucky --help
bucky -n 1000 -o firsttry ebersberger100mb/*.in
bucky -n 1000 -o firsttry ebersberger100in/*.in
bucky -n 1000 -o firsttry ebersbergerAnother100in/*.in
```

9. If things went well, remove output files from this first try, create a directory for future results (bca: Bayesian Concordance Analysis):

```
rm firsttry*
mkdir bca100
```

Now for a longer run: n =100,000 generations, 2 chains (cold & heated). The default Dirichlet parameter is $\alpha = 1$.

```
bucky -n 100000 -c 2 -o bca100/a1 ebersberger100mb/*.in
ls bca100/
```

BUCKy output files

`.input` lists input files (loci) with their assigned ID

`.out` similar to screen output. Lists parameters values, reports average SD of mean sample-wide CF (to assess convergence) and acceptance probablities when swapping between cold/heated chains.

`.cluster` posterior distribution on the # clusters, i.e. # of groups of genes that share the same tree.

`.gene` For each gene, lists its support for each tree from the individual analysis (input), and from the combined analysis.

`.concordance` Estimated population & concordance trees and more.

.input file

```
Gene Filename
=============================================
   0 ebersberger100mb/10759.in
   1 ebersberger100mb/10809.in
   2 ebersberger100mb/11959.in
  ...
  97 ebersberger100mb/9192.in
  98 ebersberger100mb/948.in
  99 ebersberger100mb/9917.in
=============================================
```

.out file

```
Bayesian Untangling of Concordance Knots (applied to yeast and other
...                                                              organisms)
  Parameter                | Usage           | Default Value | Value Used
  ------------------------------------------------------------------------
  alpha                    | -a number       | 1             | 1
  # of runs                | -k integer      | 2             | 2
  # of MCMC updates        | -n integer      | 100000        | 100000
  # of chains               | -c integer      | 1             | 2
  ...                      ...
  File with prune list     | -p pruneFile    |               |
  skip genes               | -sg             | false         | false
  Space optimization       | --opt-space     | false         | false
  ------------------------------------------------------------------------
...
Writing concordance factors to bca100/a1.concordance....done.
Average SD of mean sample-wide CF: 0.000861062
Writing single and joint gene posteriors to bca100/a1.gene....done.

MCMCMC acceptance statistics in run 1:
alpha1          <--> alpha2          accepted proposed proportion
1.000000        <--> 10.000000       148      1000     0.148000
...
Program ended at Fri May 31 15:20:49 2013
Elapsed time: 8 seconds.
```

.cluster file

```
mean #groups = 3.498
SD across runs = 0.022

credible regions for # of groups
probability region
------------------
  0.99       (3,5)
  0.95       (3,5)
  0.90       (3,5)
------------------

Distribution of cluster number in run 1:
 # of     raw     posterior
groups  counts probability
--------------------------
  2         58  0.00058000
  3      56318  0.56318000
  4      36348  0.36348000
  5       6828  0.06828000
  6        430  0.00430000
  7         18  0.00018000
--------------------------
```

.cluster file

Visualize # of clusters with bucky-tools on BUCKy webpage:

- `http://www.stat.wisc.edu/~ane/bucky/` then click to "Tools" tab on the left
- Upload your `.cluster` file.
- Read the file
- Bookmark your page, or write your file ID# down, for later use
- Options tab: choose "Clusters"
- Graph tab: histogram of posterior distribution of # clusters,

.gene file

```
Gene 0:
  numTrees = 3
  index topology              single     joint
      0 (((1,2),3),4,5);  0.993100  0.998890
      1 ((1,(2,3)),4,5);  0.003400  0.000600
      2 (((1,3),2),4,5);  0.003500  0.000510
Gene 1:
  numTrees = 14
  index topology              single     joint
      0 (((1,2),3),4,5);  0.863400  0.987010
      1 ((1,(2,3)),4,5);  0.036200  0.007490
      2 (((1,3),2),4,5);  0.032750  0.005420
      3 ((1,2),(3,4),5);  0.001600  0.000010
      4 (((1,2),4),3,5);  0.057050  0.000065
      5 ((1,(3,4)),2,5);  0.000050  0.000000
      6 (1,(2,(3,4)),5);  0.000050  0.000000
      7 (((1,4),2),3,5);  0.002450  0.000000
      9 (((1,4),3),2,5);  0.000100  0.000000
     10 ((1,(2,4)),3,5);  0.005150  0.000005
     11 (((1,3),4),2,5);  0.000150  0.000000
     12 (1,((2,3),4),5);  0.000050  0.000000
     13 ((1,3),(2,4),5);  0.000450  0.000000
     14 (1,((2,4),3),5);  0.000550  0.000000
...
```

.gene file

Visualization with bucky-tools:

- go back to bucky-tools: `www.stat.wisc.edu/~ane/bucky/`
- upload and read your `.gene` file
- Options tab: choose "gene tree probabilities"
- Graph tab: explore options. In particular:
  - Gene menu: select gene 1, gene 2, etc. Most favor tree 1.
  - back to Options tab: gene tree probability options.
    increase # genes to show 100 genes.
    back to Graphs tab.
  - click on the bar to see tree 1.
  - select gene 98: low-informative gene.
    change probability type from "single" to "joint"
  - select gene 56 or 17: switch from "single" to "joint"

.concordance file

First section: estimated population tree (with/without branch lengths) and estimated concordance tree (with/without concordance factors):

```
translate
 1 human,
 2 chimp,
 3 gorilla,
 4 orang,
 5 rhesus;

Population Tree:
(((1,2),3),4,5);
Primary Concordance Tree Topology:
(((1,2),3),4,5);

Population Tree, With Branch Lengths In Estimated Coalescent Units:
(((1:10.000,2:10.000):1.050,3:10.000):4.675,4:10.000,5:10.000);
Primary Concordance Tree with Sample Concordance Factors:
(((1:1.000,2:1.000):0.765,3:1.000):0.993,4:1.000,5:1.000);
```

.concordance file

To draw concordance tree and population tree:

- quality figures for publication: FigTree or other tree drawing software. Copy and paste the trees from .concordance file.
- exploratory: bucky-tools `www.stat.wisc.edu/~ane/bucky/` upload and read your `.concordance` file.
  Options tab: Concordance and Population trees
  Graph tab: view both trees

.concordance file

Second section: More information on concordance factors.

```
Four-way partitions in the Population Tree:
   sample-wide CF, coalescent units and Ties(if present)
{1; 2|3; 4,5}   0.767, 1.050,
{1,2; 3|4; 5}   0.994, 4.675,
```

- Recall: sample-wide versus genome-wide concordance factors.
- CF of 4-way partitions: average CFs of all quartets defining an edge. Here:
  `{1; 2|3; 4,5}` is defined by quartets $\{12|34\}$ and $\{12|35\}$.
  `{1,2; 3|4; 5}` is defined by $\{13|45\}$ and $\{23|45\}$.
- Associated branch length in coalescent units for quartets:

$$t = -\log\left(\frac{3}{2}(1 - \mathrm{CF})\right)$$

.concordance file

next: concordance factors of splits with credibility intervals.

```
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} 0.993(0.970,1.000) 0.986(0.945,1.000)       0.001
{1,2|3,4,5} 0.765(0.670,0.860) 0.759(0.625,0.878)       0.001

Splits NOT in the Primary Concordance Tree but with estimated
                                                    CF > 0.05:
{1,4,5|2,3} 0.124(0.050,0.220) 0.125(0.034,0.242)        0.001
{1,3|2,4,5} 0.107(0.040,0.190) 0.108(0.030,0.215)        0.001

Average SD of mean sample-wide CF: 0.001
```

- The 2 splits conflicting with 12|345 (not in the concordance tree) have overlapping credibility intervals, both sample-wide and genome-wide: compatible with ILS.
- Credibility intervals for genome-wide CF are wider than for sample-wide CF. The difference goes away with more loci.

.concordance file

- visualize CF and their credibility intervals: online bucky-tools.
  Options tab: "Splits"
  Shows which clades are conflicting, and alternative placements.
- non-overlapping intervals as evidence for gene flow or hybridization:

For a bipartition in the concordance tree, take the 2 alternative and conflicting clades. If their CFs have non-overlapping credibility intervals, this is evidence that ILS is *not* the only cause of discordance.

.concordance file

Last section: lists each split, complete posterior distribution for its CF, separately from each run to assess convergence.

```
All Splits:
{1,2,3|4,5}
#Genes  count in run(s) 1 through 2, Overall probability, Overall
                                                  cumulative probability
    91           0           1     0.000005    0.000005
    92           1           2     0.000015    0.000020
    93          11           8     0.000095    0.000115
    94          44          22     0.000330    0.000445
    95         236         158     0.001970    0.002415
    96        1202         898     0.010500    0.012915
    97        4497        3445     0.039710    0.052625
    98       12529       10292     0.114105    0.166730
    99       25266       26870     0.260680    0.427410
   100       56214       58304     0.572590    1.000000

mean CF =   0.993 (proportion of loci)
        =  99.337 (number of loci)
99% CI for CF = (96,100)
95% CI for CF = (97,100)
90% CI for CF = (97,100)
...
```

{1,4,5|2,3}

#Genes	count in run(s) 1	through 2,	Overall probability,	Overall cumulative probability
0	232	71	0.001515	0.001515
1	138	104	0.001210	0.002725
2	279	248	0.002635	0.005360
3	640	572	0.006060	0.011420
4	1320	1238	0.012790	0.024210
5	2087	2090	0.020885	0.045095
6	3365	3284	0.033245	0.078340
7	4722	4743	0.047325	0.125665
8	6173	6105	0.061390	0.187055
9	7778	7304	0.075410	0.262465
10	8713	8531	0.086220	0.348685
11	9128	9084	0.091060	0.439745
12	9229	9239	0.092340	0.532085
13	8764	8773	0.087685	0.619770
14	8172	8009	0.080905	0.700675
15	6869	7090	0.069795	0.770470
16	5754	5895	0.058245	0.828715
17	4819	4845	0.048320	0.877035
18	3624	3626	0.036250	0.913285
19	2617	2903	0.027600	0.940885
20	1852	2083	0.019675	0.960560
21	1356	1475	0.014155	0.974715
22	907	1071	0.009890	0.984605
23	575	678	0.006265	0.990870
24	366	402	0.003840	0.994710
25	207	257	0.002320	0.997030
26	148	124	0.001360	0.998390
27	75	68	0.000715	0.999105
28	46	41	0.000435	0.999540
29	20	26	0.000230	0.999770
30	13	16	0.000145	0.999915
31	5	2	0.000035	0.999950
32	3	1	0.000020	0.999970
33	4	2	0.000030	1.000000

mean CF = 0.124 (proportion of loci)
= 12.386 (number of loci)
99% CI for CF = (2,25)
95% CI for CF = (5,22)
90% CI for CF = (6,20)

Options for BUCKy: `bucky -h`

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
CF cutoff for display	-cf number	0.05
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
taxon set	-p prune-file	common taxa
skip genes with fewer taxa	-sg	false
Space optimization	--opt-space	false
do not build population tree?	--no-population-tree	false
grid-size for genomewide CF	--genomewide-grid-size	1000
help	-h OR --help	
version	--version	

Options for BUCKy

```
bucky -a 5 -k 4 -c 2 -n 100000 -s1 5420 -s2 7521 -o bca100/a5
                                                ebersberger100mb/*.in
```

- `-a 5` sets $\alpha = 5$. Recall that $\mathbb{P}(\tau_i = \tau_j|\alpha) = (1 + \alpha/T)/(1 + \alpha)$ for 2 random genes *i* and *j*. With $\alpha = 5$, $T = 15$ (5 taxa), we get $\mathbb{P}(\tau_i = \tau_j|\alpha) \approx 0.22$. This is higher than $1/15 \approx 0.07$ but lower than expected under coalescent with most extreme ILS on a single branch only $(1/3)$.
- `-k 4` sets 4 independent runs. Better for assessing convergence.
- `-c 2` sets 2 chains: one cold, one heated. Can help convergence, but not always.
- `-n 100000` to run 100,000 MCMC generations (+10% burnin)
- `-s1 5420` and `-s2 7521` set the random seeds
- `-o bca100/a5` sets the root name for output files

Options for BUCKy

```
bucky -a 5 -k 4 -c 2 -n 100000 -s1 5420 -s2 7521 -o bca100/a5
                                                ebersberger100mb/*.in
```

10. Run the command then:
    - check .out file to assess convergence.
    - How did the concordance tree changed compared to $\alpha = 1$?
    - How did the concordance factors changed? Change in branch lengths (coalescent units) in the estimated population tree?

α infinite

$\alpha =$ infinity is equivalent to independence among gene trees: no sharing of topological information among loci, like in a consensus approach. Most extreme a priori level of discordance.

11. Analyze the data with $\alpha = \infty$, check convergence, trees, and any change in CFs and branch lengths compared to $\alpha =$ 1 or 5.

```
bucky --use-independence-prior -k 4 -n 100000 -o bca100/aInf
                                                ebersberger100mb/*.in
```

Heated chains

```
bucky -a 0.1 -c 3 -m 5 -r 50 -n 100000 -s1 5420 -s2 7521
                                -o bca100/a01 ebersberger100mb/*.in
```

- When mixing is hard to obtain, heated chains can help. Cost: additional computation for each chain. In BUCKy, heated chains use higher α's than the cold chain: easier to mix.
- `-c 3` to use 1 cold chain and 2 heated chains.
- `-r 50` to attempt chain swapping every 50 generations.
- `-m 5` to use values of 5α, $5^2\alpha$, etc. for heated chains. Here the cold chain uses a Dirichlet prior with $\alpha = 0.1$ and the 2 heated chains use $\alpha = 0.5$ and 2.5.

12. Run the command then check the acceptance probabilities when trying to swap chains. If too low: lower the α multiplier (`-m` option). If too high: increase the multiplier.

More info on clusters

13. Use `--calculate-pairs` to get an extra output file, with posterior probabilities that 2 genes have the same gene: $\mathbb{P}(\tau_i = \tau_j | D, \alpha)$, for every pair of genes. Large matrix (100 by 100 here) with 1's on the diagonal.

```
bucky -a 1 -k 3 -n 100000 --calculate-pairs -s1 5414 -s2 1752
                                -o bca100/a1 ebersberger100mb/*.in
more bca100/a1.pairs
```

Cost: more computation time.

More info on clusters

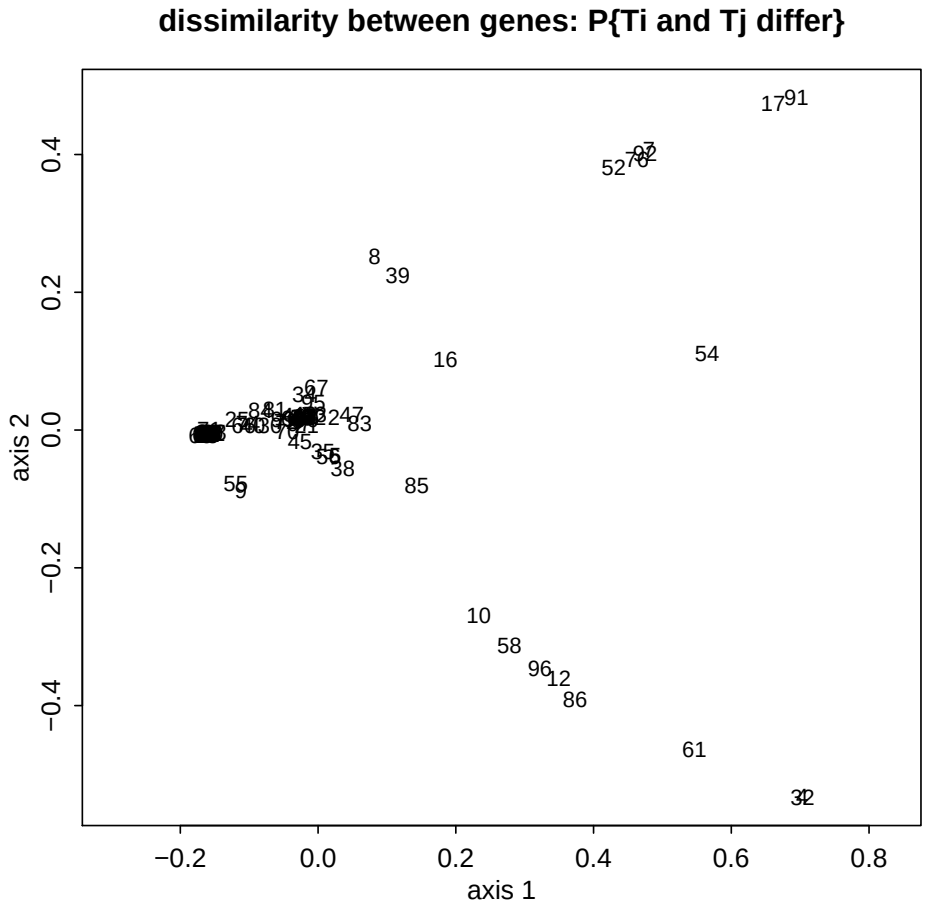
14. open `bca100/a1.pairs` in Excel, then in R to perform NMDS and visualize the values $\mathbb{P}(\tau_i = \tau_j | D, \alpha)$, treated as a similarities. Distance between gene *i* and gene *j*: $1 - \mathbb{P}(\tau_i = \tau_j)$. R code:

```
# read in the matrix P(Ti=Tj|data,alpha):
mat = read.table("bca100/a1.pairs", row.names=1)
mat=as.matrix(mat)
# set column names to gene IDs. Actual gene names would be best...
colnames(mat) = rownames(mat)
mat             # Just to check. Big! 100x100 matrix
diag(mat)       # check that diagonal has ones only.
# now NMDS calculations:
mds = cmdscale(1-mat)
str(mds) # matrix of 100 rows and 2 columns: for the 2 axes.
plot(mds[,1], mds[,2], type="n", xlab="axis 1", ylab="axis 2",
     asp=1, # to keep aspect ratio = 1 between both axes
     main="dissimilarity between genes: P{Ti and Tj differ}")
text(mds[,1], mds[,2], rownames(mds), cex=0.6)
```

More info on clusters

Large cluster: gene trees matching the species tree.

Check `.gene` file.



Genes on the top right corner (e.g. 17,91) favor `(chimp,gorilla)`; genes on the bottom right (e.g. 4, 32, 61) favor `(human,gorilla)`.

Selecting a subset of taxa

Use option `-p filename` to specify which taxa should be used for analysis. To exclude taxon 'Rhesus', use file `HCGOtaxonlist.nex`, which should simply have a translate section:

```
translate
        1 human,
        2 chimp,
        3 gorilla,
        4 orang;
```

Selecting a subset of taxa

15. exclude rhesus macaque, keep the other 4 taxa only, then check estimated trees, CFs and branch length (concordance units) along the lineage ancestor to HC:

```
bucky -a 1 -n 100000 -p HCGOtaxonlist.nex -o bca100/a1_HCGO
                                          ebersberger100mb/*.in
more bca100/a1_HCGO.concordance
```

16. Now exclude Orangutan only, to use Rhesus instead as outgroup. Does the change in outgroup affect the estimated tree/CF/branch lengths?

```
bucky -a 1 -n 100000 -p HCGRtaxonlist.nex -o bca100/a1_HCGR
                                          ebersberger100mb/*.in
more bca100/a1_HCGR.concordance
```

Missing data: Selecting taxa or loci

Artificial data for which some taxa lack some sequences, in:

```
ls ebersberger5in_missingtaxa

more ebersberger5in_missingtaxa/187.in
more ebersberger5in_missingtaxa/253.in
more ebersberger5in_missingtaxa/948.in
```

Create a new directory for the results from this data set:

```
mkdir bca5
```

Missing data: Selecting taxa or loci

By default, `bucky` prunes all taxa that are missing one or more genes. With option `-sg` all taxa are kept, instead `bucky` skips any gene with some missing taxa.

17. Execute the following commands. The first one does not work because there are only 2 taxa left that do not lack any sequence (2 taxa is not enough).

    The second one runs: all taxa are kept but only 1 gene is kept (complete taxon sampling): useless.
    Check the list of included genes in `.input` file.

```
bucky -n 100000 -o bca5/a1 ebersberger5in_missingtaxa/*.in

bucky -n 100000 -o bca5/a1 -sg ebersberger5in_missingtaxa/*.in
more bca5/a1.input
```

Missing data: Selecting taxa or loci

18. Combine options `-p file` to impose which taxa to keep, and `-sg` to skip genes that do not have all taxa in the specified list.

    Check the list of retained genes: `.input` file (3 out of 5 here).

```
bucky -n 100000 -sg -p HCGRtaxonlist.nex -o bca5/a1_HCGR
                                    ebersberger5in_missingtaxa/*.in

more bca5/a1_HCGR.input
```