Phenotype-Genotype Branch-Site Model

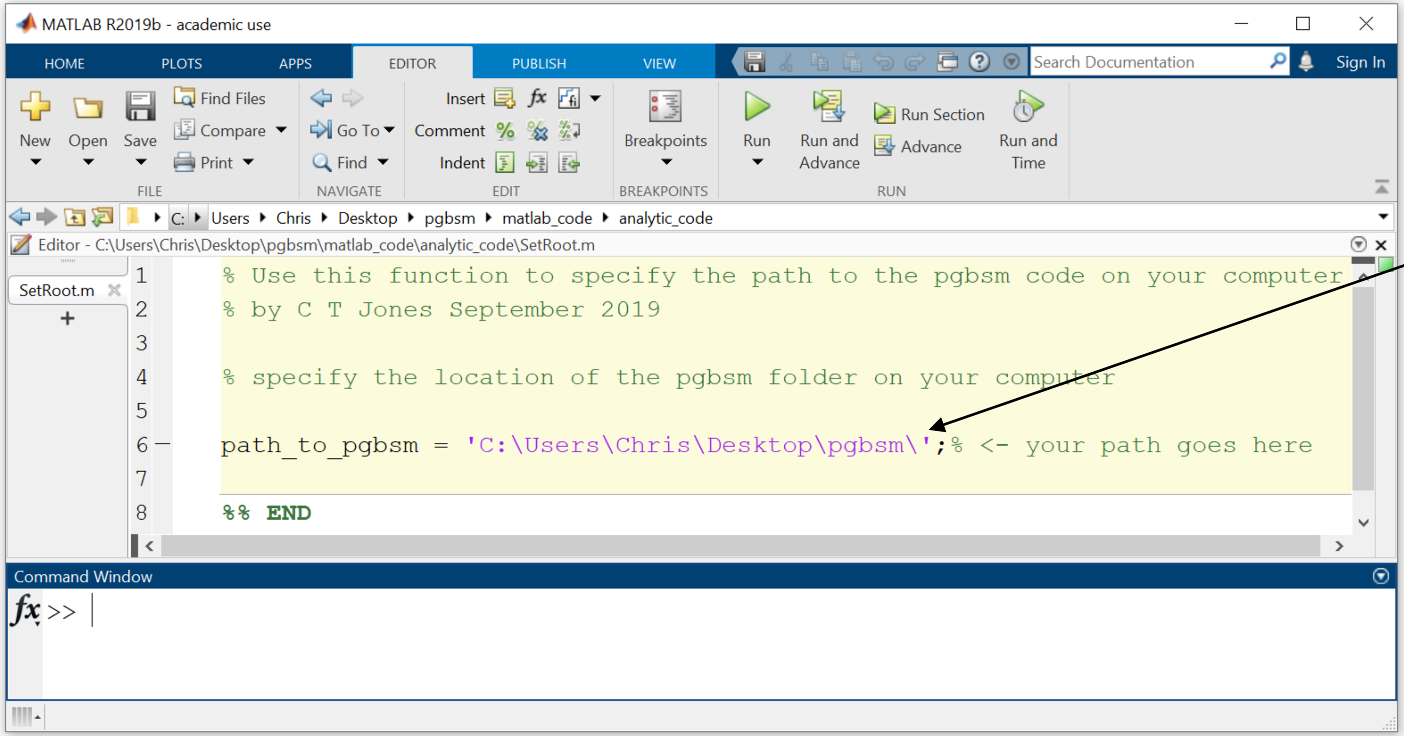
Users Guide Version 1.00

By C T Jones, Autumn 2019

cjones2@dal.ca

Setting up the code:

1. The folder pgbsm contains all scripts and data necessary to both simulate alignments using **MSmmtDNA** and to fit your own data to **RaMoSS** and to the **PG-BSM**.
2. The first step in setting up the code is to put the pgbsm folder somewhere on your computing system.
3. Next, you must tell Matlab where to look for scripts by editing the script called SetRoot.m found in the following three folders generating_code, analytic_code, and formatting_code all of which are contained in the folder matlab_code.



```
% Use this function to specify the path to the pgbsm code on your computer
% by C T Jones September 2019

% specify the location of the pgbsm folder on your computer

path_to_pgbsm = 'C:\Users\Chris\Desktop\pgbsm\';% <- your path goes here

%% END
```

You must set your path in all three versions of SetRoot.m

The text in this Power Point document is color-coded: green indicates a folder, red a script and blue a data file.

- **MSmmtDNA** = Mutation-Selection Mammalian Mitochondrial DNA – a generating model that was formulated to mimic a real alignment of mammalian mtDNA (Jones et al. 2018).
- **RaMoSS** = Random Mixture of Static and Switching Sites – a codon substitution model designed to account for a mixture of sites evolving with a constant dN/dS rate ratio and sites evolving with variations in dN/dS over time (i.e., heterotachy, Jones et al. 2017).
- **PG-BSM** = Phenotype-Genotype Branch-Site Model – a codon substitution model designed to detect codon sites that underwent adaptive evolution in conjunction with changes in a discrete phenotype.

The folder matlab_code contains three folders, each with its own set of code:

1. The folder generating_code contains scripts simulation_setup.m and simulate_alignments.m. The first script simulation_setup.m is used to specify parameters for a simulation, including tree topology and branch lengths, taxa names, the number of codon sites, the number of alignments to be generated, and the proportion of sites to be generated with changes in their site-specific landscapes. These parameters and more can be changed by editing the script directly. Running the script produces a data file called setupData.mat, a text file called taxon_labels.txt, and a figure showing your tree with taxon labels and the branches over which the phenotype was made to change. Data files are stored in the folder simulated_alignments. The second script simulate_alignments.m is used to generate the alignments. It reads setupData.mat and produces two text files, phenotype_map.txt and tree_data.txt. It also generates alignments according to your specifications and stores them as text files with numerical labels seqfile_nnn.txt, where nnn is a number between 001 and 999. These are all stored in the folder simulated_alignments.

2. The folder analytic_code contains scripts process_my_simulations.m, process_my_data.m, and visualize_my_data.m. Running the script process_my_simulations.m will process the alignment files stored in simulated_alignments. Each alignment will be fitted to the null PG-BSM and three versions of the alternate PG-BSM designed to test for branch-wise, clade-wise and reverse clade-wise sites. See Jones et al. 2019 for a detailed description of the PG-BSM. The script process_my_data.m fits your own real-data alignment to both RaMoSS and the PG-BSM. The script visualize_my_data.m can be used to visualize the results of the fit of these models to your data.

3. The folder formatting_code contains the script format_my_data.m. This is used to convert your data files into the specific format required for the script process_my_data.m. The script format_my_data.m also makes assumptions about the format of your data. Details are provided on the next few slides.

Input data format:

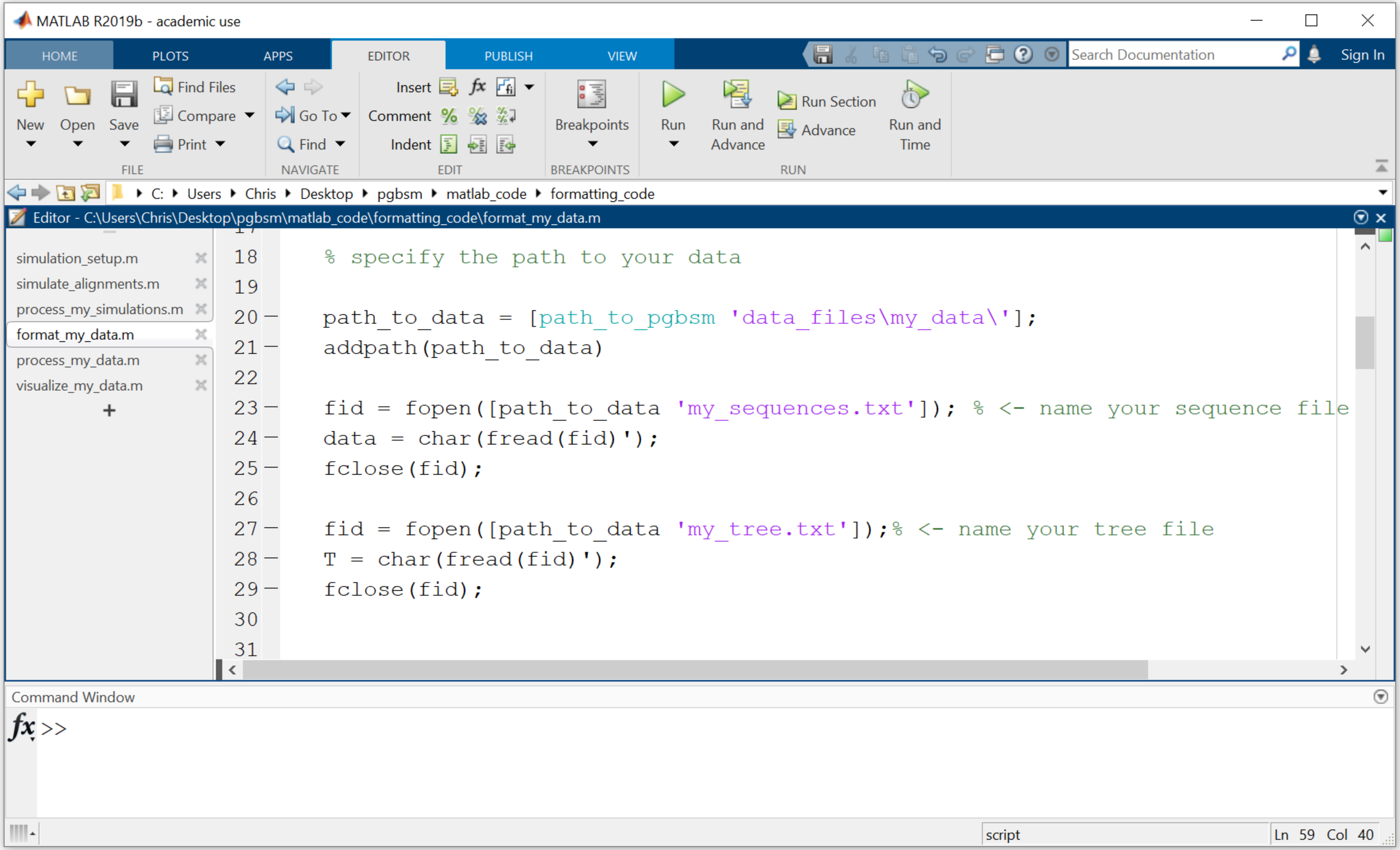
The analytic code requires taxa to be indicated in the tree by numbers 1 to nL (where nL is the number of sequences or leaf nodes in the tree) and sequences to appear in the same order in which they occur in the tree. The script format_my_data.m was written to help make your data meet these requirements. It assumes the following common tree format:

```
(((harpsichord:0.25,piano forte:0.25):0.55,(acoustic guitar:0.30,electric guitar:0.25):0.60):1.75,((trumpet:0.25,trombone:0.25):0.65,(cornet:0.25,tuba:0.50):0.50):1.75);
```

The branch lengths that appear in the tree (the numbers following each colon) provide an initial estimate for the first analytic model fitted to the alignment, the null PG-BSM. Branch lengths can be set to any initial value. The sequence file is assumed to be formatted as follows, where 8 indicates the number of sequences and 15 the number of nucleotides in each sequence.

```
8 15

tuba TTC CAC ACT TCA CAA

piano forte TTC CAC ACT TCA CAA

cornet TTC CAT ACT TCA CAA

electric guitar TTC CAC ACC TCA CAA

harpsichord ACC TAC AAG TTA CTG

trombone  ACC TAC AAA TTA CTA

acoustic guitar  TTC CAC ACT CTC CAA

trumpet  TTC CAC ACT CTG CAA
```

Your tree and sequence data must be stored in the folder my_data in text files that you name yourself. The same names must appear in the script format_my_data.m near the top of the script, as indicated here.



Editor - C:\Users\Chris\Desktop\pgbsm\matlab_code\formatting_code\format_my_data.m

```
% specify the path to your data

path_to_data = [path_to_pgbsm 'data_files\my_data\'];
addpath(path_to_data)

fid = fopen([path_to_data 'my_sequences.txt']); % <- name your sequence file
data = char(fread(fid)');
fclose(fid);

fid = fopen([path_to_data 'my_tree.txt']);% <- name your tree file
T = char(fread(fid)');
fclose(fid);
```

```
>>
```

The script format_my_data.m changes taxa names to numbers 1 to nL.

```
(((harpsichord:0.25,piano forte:0.25):0.55,(acoustic guitar:0.30,electric guitar:0.25):0.60):1.75,((trumpet:0.25,trombone:0.25):0.65,(cornet:0.25,tuba:0.50):0.50):1.75);

(((1:0.25,2:0.25):0.55,(3:0.30,4:0.25):0.60):1.75,((5:0.25,6:0.25):0.65,(7:0.25,8:0.50):0.50):1.75);
```

It also writes the sequences in the order they appeared in your tree file.

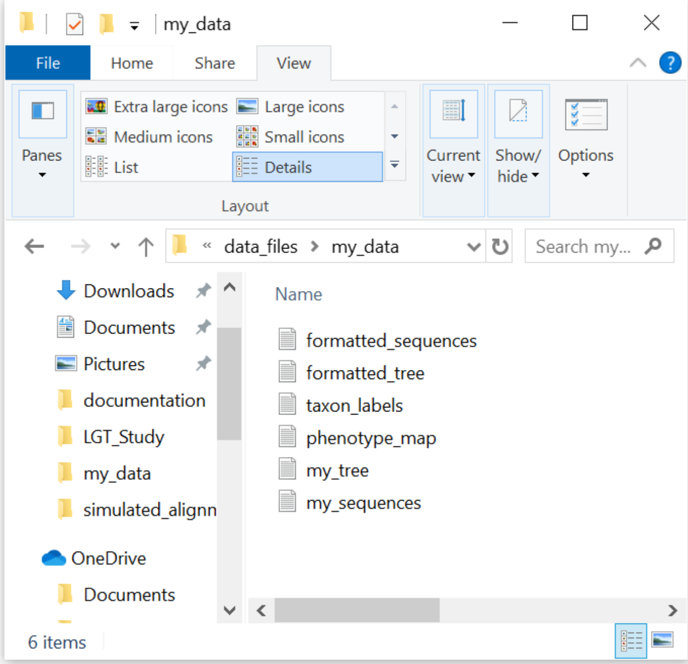
```
8 15
tuba TTC CAC ACT TCA CAA
piano forte TTC CAC ACT TCA CAA
cornet TTC CAT ACT TCA CAA
electric guitar TTC CAC ACC TCA CAA
harpsichord ACC TAC AAG TTA CTG
trombone  ACC TAC AAA TTA CTA
acoustic guitar  TTC CAC ACT CTC CAA
trumpet  TTC CAC ACT CTG CAA
```

```
8 15
harpsichord ACC TAC AAG TTA CTG
piano forte TTC CAC ACT TCA CAA
acoustic guitar  TTC CAC ACT CTC CAA
electric guitar TTC CAC ACC TCA CAA
trumpet  TTC CAC ACT CTG CAA
trombone  ACC TAC AAA TTA CTA
cornet TTC CAT ACT TCA CAA
tuba TTC CAC ACT TCA CAA
```

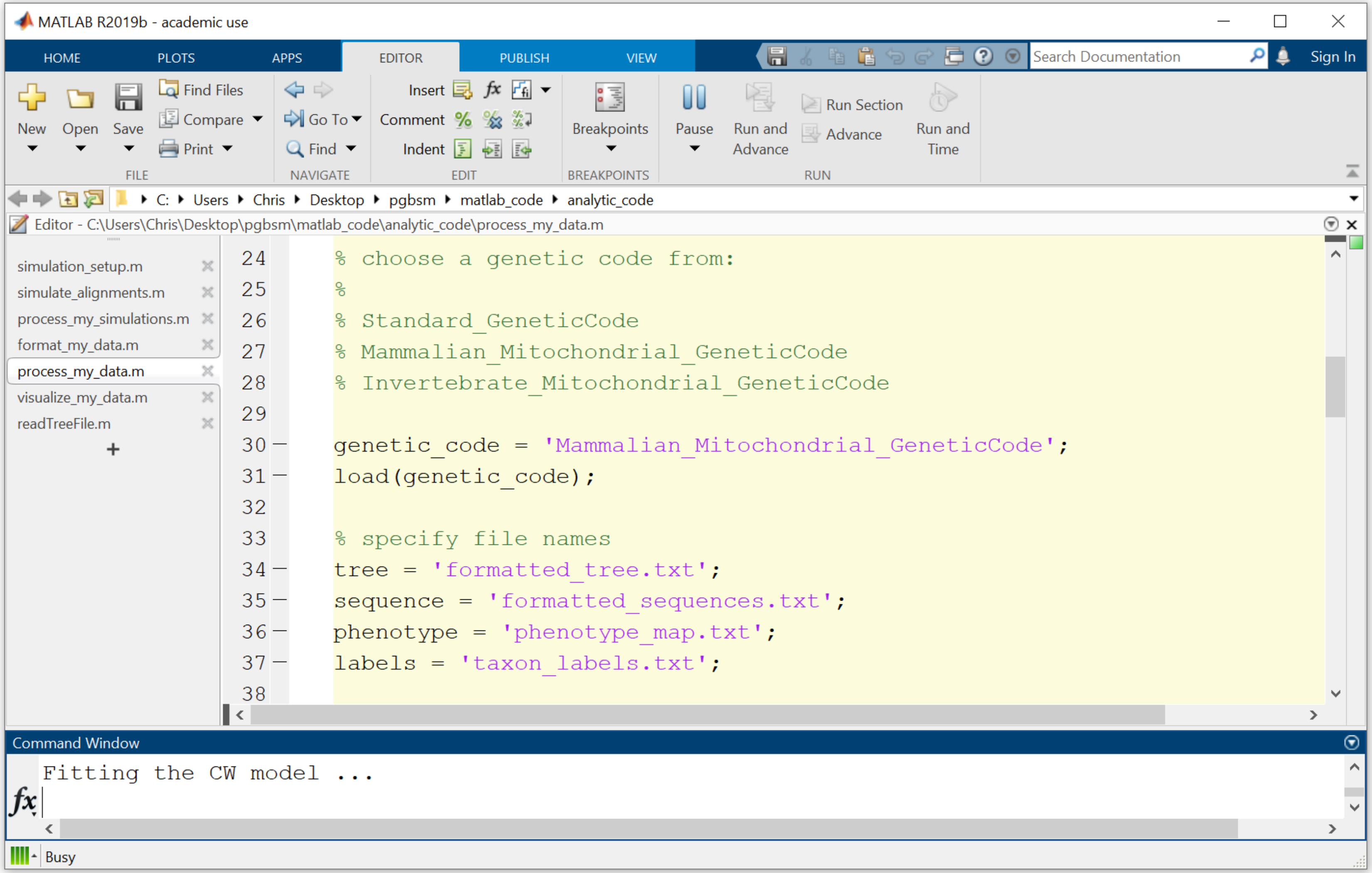
The new tree is stored in a file called formatted_tree.txt and the sorted sequences in formatted_sequences.txt. Both are stored in the folder my_data. The code also writes a list of taxon names in the file taxon_labels.txt. The analytic code that fits the PG-BSM to your data requires a phenotype map like this: `phenotypeMap = 1  1  1  1  2  2  1  1` The numbers indicate different values for a discrete phenotype. You will need to make a text file with this information and put it in the folder my_data.

Processing your data:

The script process_my_data.m fits your own alignment to both RaMoSS and the PG-BSM. Before running the code first check that all necessary files are contained in the folder my_data. The names of these files must appear at the top of process_my_data.m. You will also need to select a genetic code.



formatted_sequences
formatted_tree
taxon_labels
phenotype_map
my_tree
my_sequences



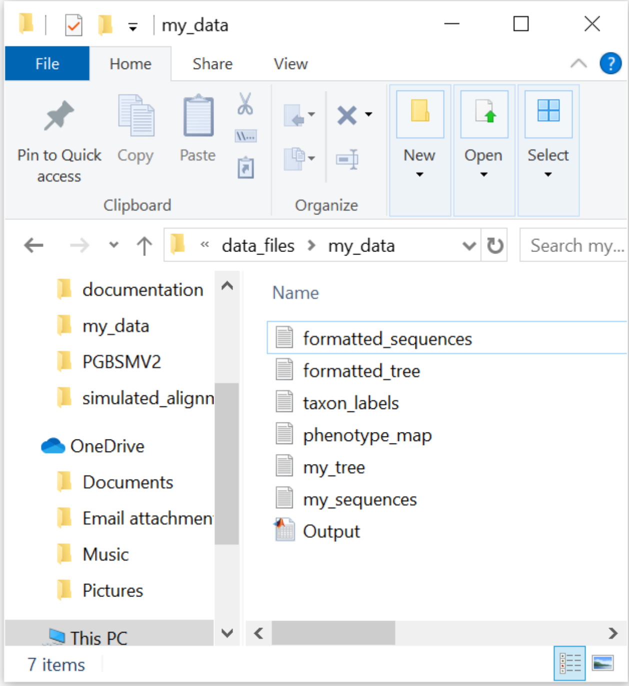
Editor - C:\Users\Chris\Desktop\pgbsm\matlab_code\analytic_code\process_my_data.m

```
% choose a genetic code from:
%
% Standard_GeneticCode
% Mammalian_Mitochondrial_GeneticCode
% Invertebrate_Mitochondrial_GeneticCode

genetic_code = 'Mammalian_Mitochondrial_GeneticCode';
load(genetic_code);

% specify file names
tree = 'formatted_tree.txt';
sequence = 'formatted_sequences.txt';
phenotype = 'phenotype_map.txt';
labels = 'taxon_labels.txt';
```

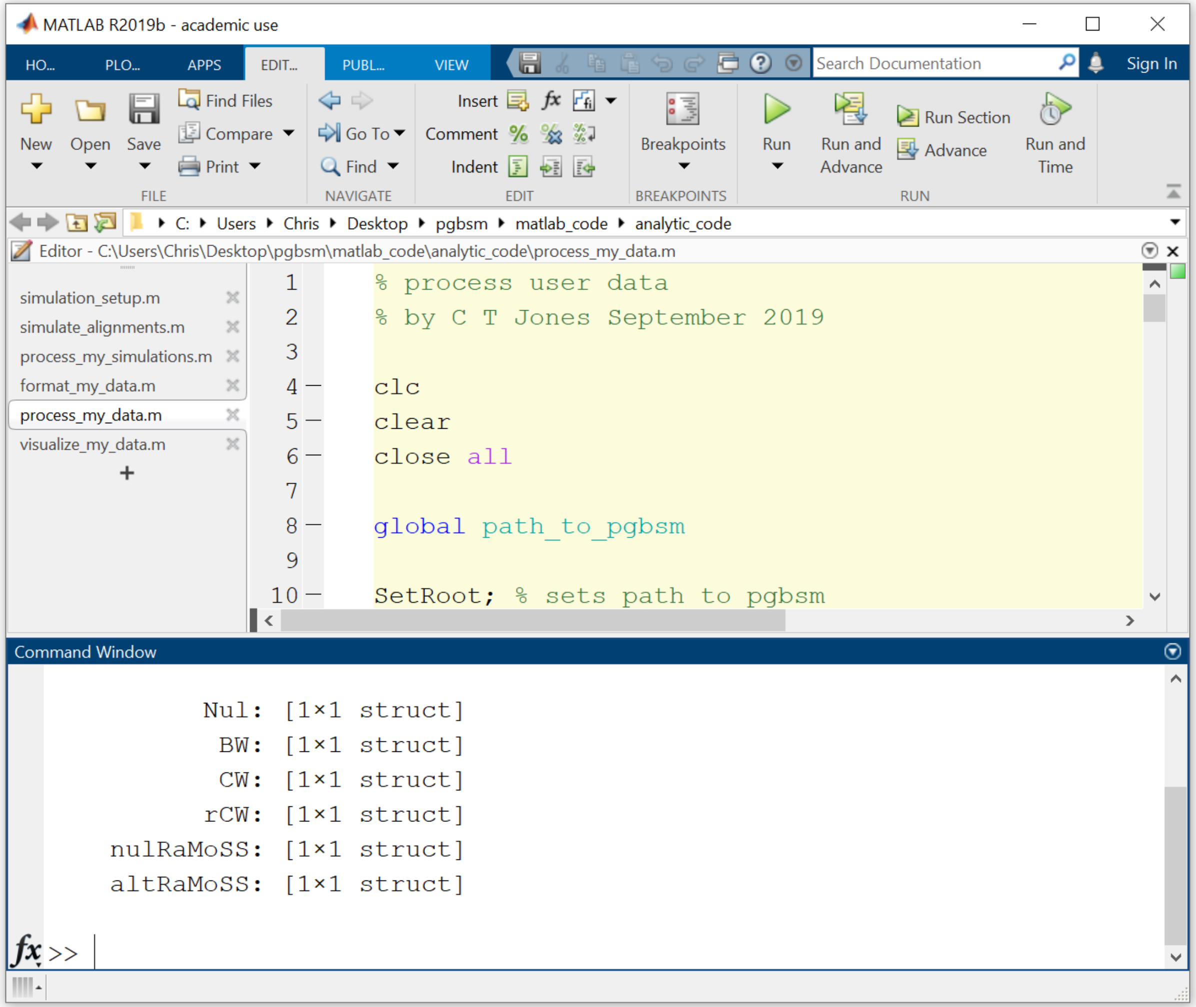
```
Fitting the CW model ...
```

Model Output:

The results of the fit of RaMoSS and the PG-BSM to your data are stored in Output.mat as a data structure with one field for each fitted model. Output.mat can be found in the folder my_data.



- formatted_sequences
- formatted_tree
- taxon_labels
- phenotype_map
- my_tree
- my_sequences
- Output



```
% process user data
% by C T Jones September 2019

clc
clear
close all

global path_to_pgbsm

SetRoot; % sets path to pgbsm
```

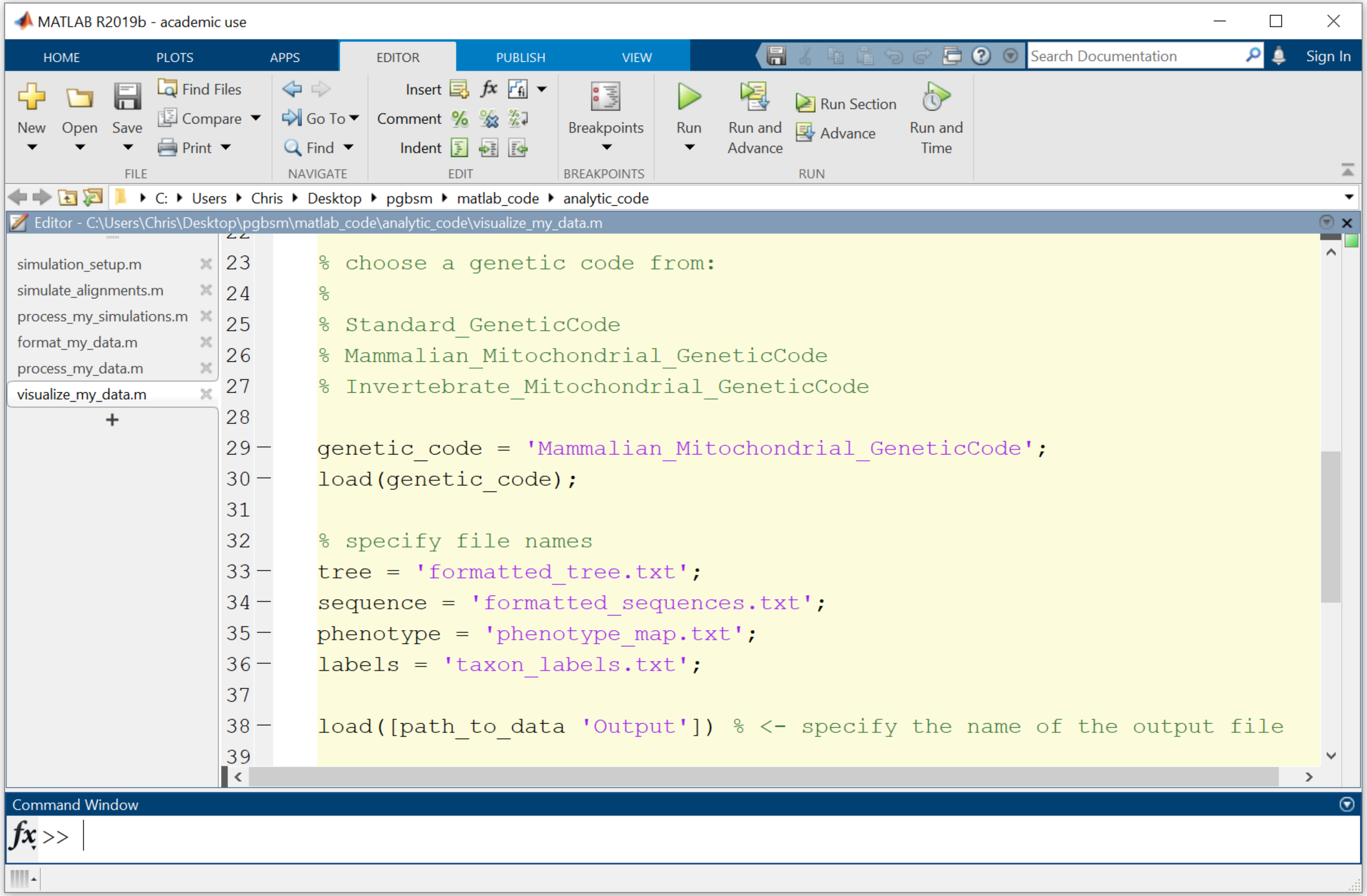
```
          Nul: [1×1 struct]
           BW: [1×1 struct]
           CW: [1×1 struct]
          rCW: [1×1 struct]
    nulRaMoSS: [1×1 struct]
    altRaMoSS: [1×1 struct]

fx >>
```

- Nul = null PG-BSM
- BW = alt PG-BSM for branch-wise sites
- CW = alt PG-BSM for clade-wise sites
- rCW = alt PG-BSM for reverse clade-wise sites
- nulRaMoSS = null RaMoSS
- altRaMoSS = alt RaMoSS

Each model structure includes a log-likelihood score and a vector of maximum-likelihood estimates for all model parameters. The BW, CW and rCW model structure also include POST, a matrix of posterior probabilities.

Visualizing Output:

Model output can be displayed graphically by running the script visualize_my_data.m. This code also generates a text file with all of the data in Output.mat. Before running this code, you must specify the genetic code and file names at the top of the script.



Editor - C:\Users\Chris\Desktop\pgbsm\matlab_code\analytic_code\visualize_my_data.m

```
    % choose a genetic code from:
    %
    % Standard_GeneticCode
    % Mammalian_Mitochondrial_GeneticCode
    % Invertebrate_Mitochondrial_GeneticCode

    genetic_code = 'Mammalian_Mitochondrial_GeneticCode';
    load(genetic_code);

    % specify file names
    tree = 'formatted_tree.txt';
    sequence = 'formatted_sequences.txt';
    phenotype = 'phenotype_map.txt';
    labels = 'taxon_labels.txt';

    load([path_to_data 'Output']) % <- specify the name of the output file
```

Visualizing Output:

Running visualize_my_data.m will generate 7 figures and 4 data files.

Figures include the following:

1. A comparison of the log-likelihoods for the 5 fitted models: the null PG-BSM, BW PG-BSM, CW PG-BSM, rCW PG-BSM, null RaMoSS and alterative RaMoSS.
2. A bar plot showing the log-likelihood ratios testing for BW sites, CW sites, rCW sites, and covarion-like switching. All tests involve a single parameter and are conducted assuming a chi-squared distribution with 1 degree of freedom.
3. Three bar plots showing posterior probabilities P(BW), P(CW), and P(rCW) for all sites in the alignment.
4. A plot comparing branch-length estimates for the five models.
5. Three trees showing the most likely history of the phenotype under the BW PG-BSM, CW PG-BSM, and rCW PG-BSM models.

The four data files include three files that contain the site patterns most consistent with the BW, CW and rCW processes after controlling the false discovery count to 1 false discovery per process, and an file with all model output. These files, BW.txt, CW.txt, rCW.txt and Output.txt, are stored in the folder my_data.

Examples of all figures and files are shown on the next several slides.

Figure 1: The log-likelihood for each model minus that for the best fitting model (the model with no bar). Shorter bars indicate better fit.

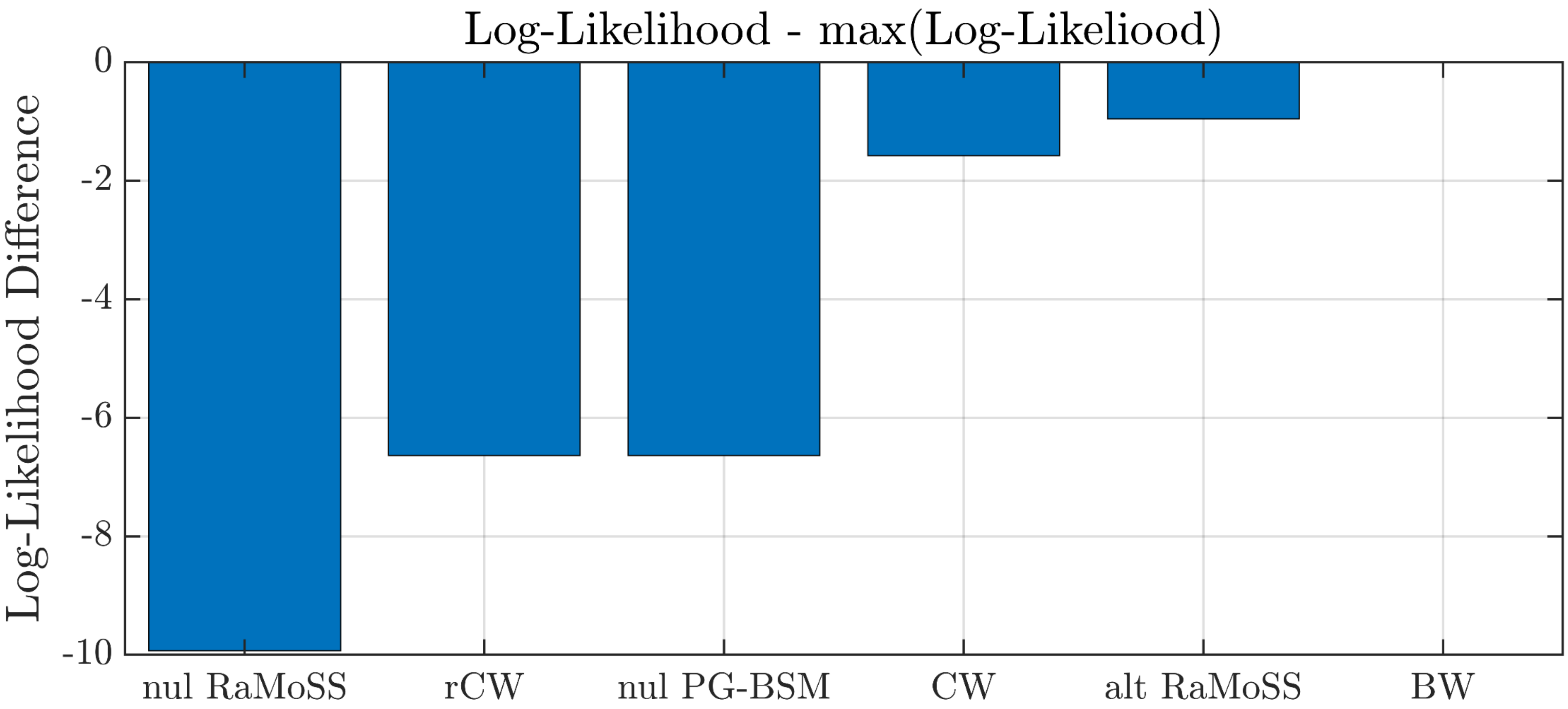


Figure 2: Model contrasts testing for BW, CW, rCW, and Covarion-like sites (using RaMoSS). All tests are chi-squared with 1 degree of freedom. The red line indicates the critical value for a 1% test (6.63).

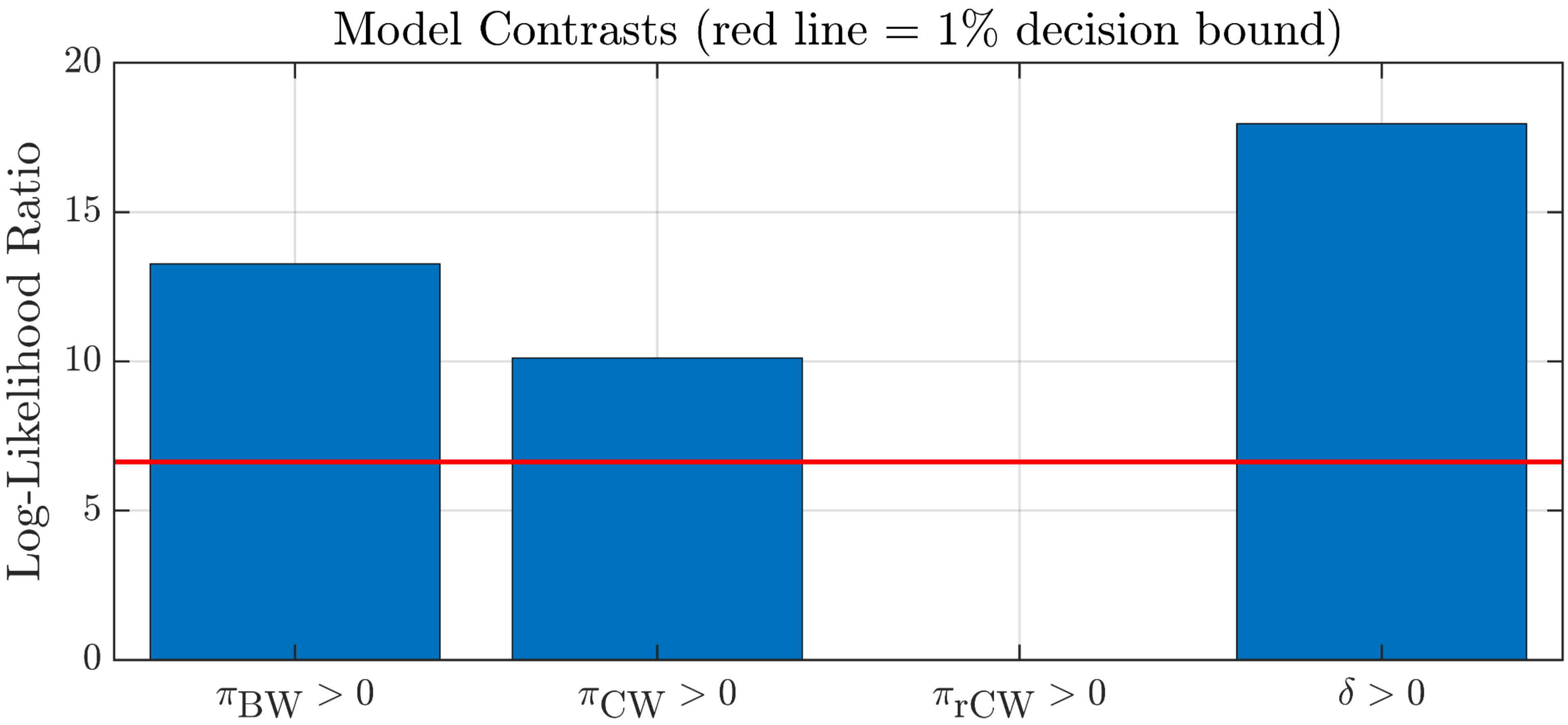


Figure 3: Poster probabilities for all sites under each version of the alternative PG-BSM.

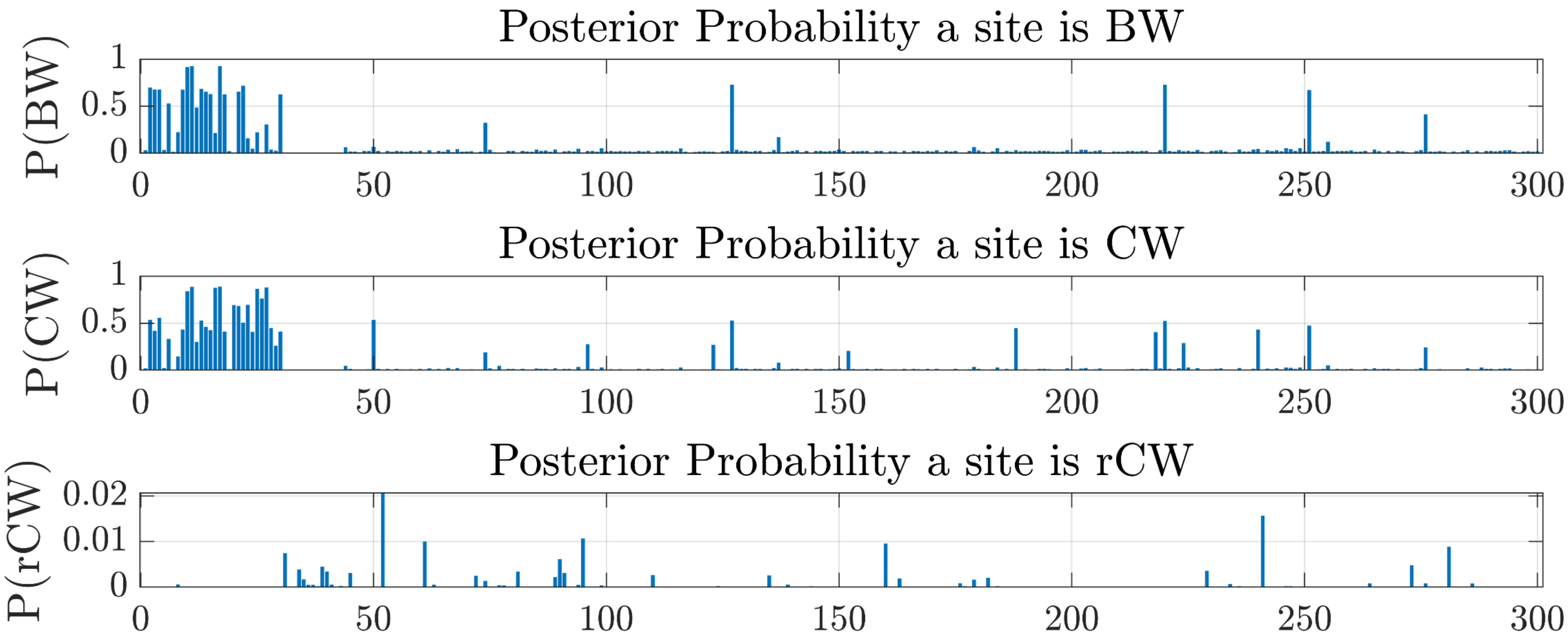


Figure 4: A comparison of branch-length estimates under the five fitted models.

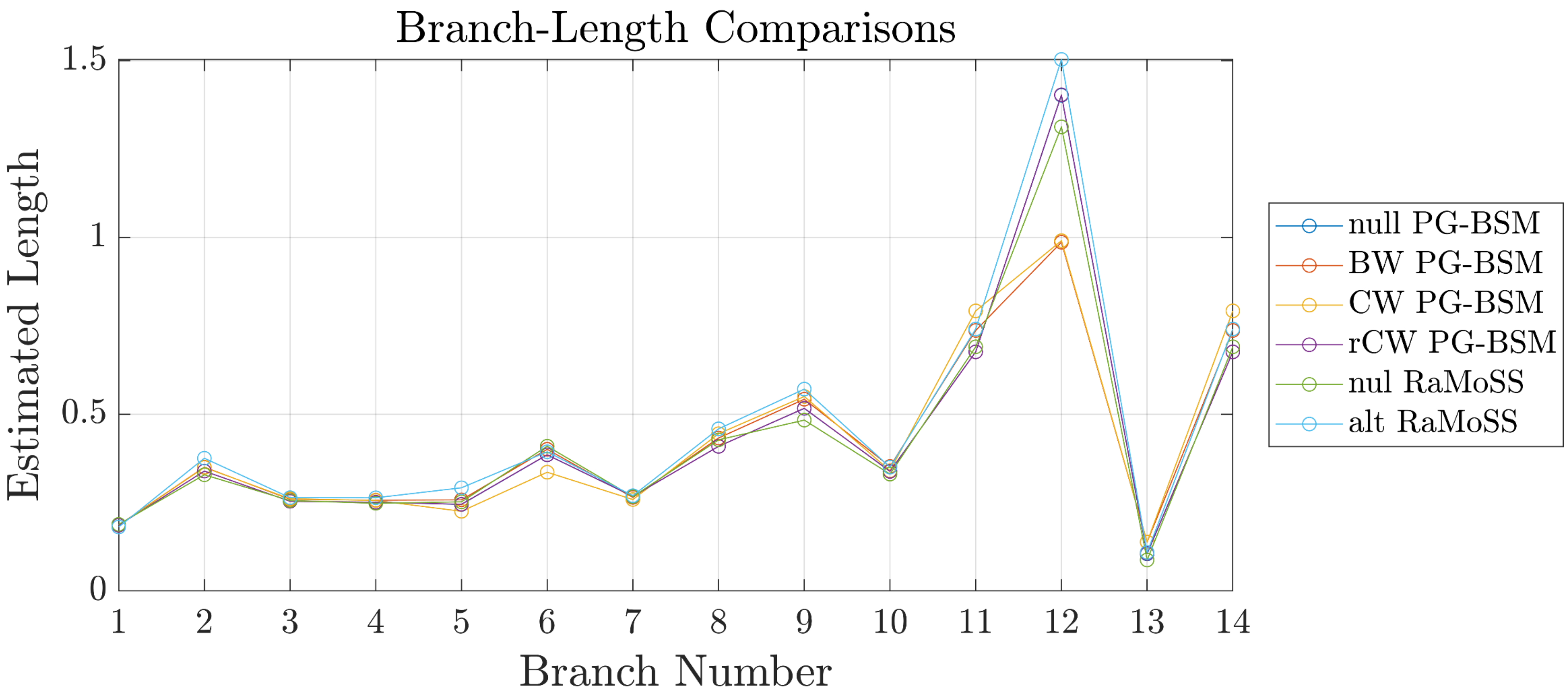


Figure 5: Estimates under the BW PG-BSM.

Numbers next to each taxon name indicated the assigned phenotype.

Bold branches indicate those over which the phenotype most likely changed (indicating the most likely history of the phenotype z).

Post(z) is the posterior probability of z.

Note that Post(z) is computed by running the visualize_my_data.m script and does not appear in Output.mat or Output.txt.

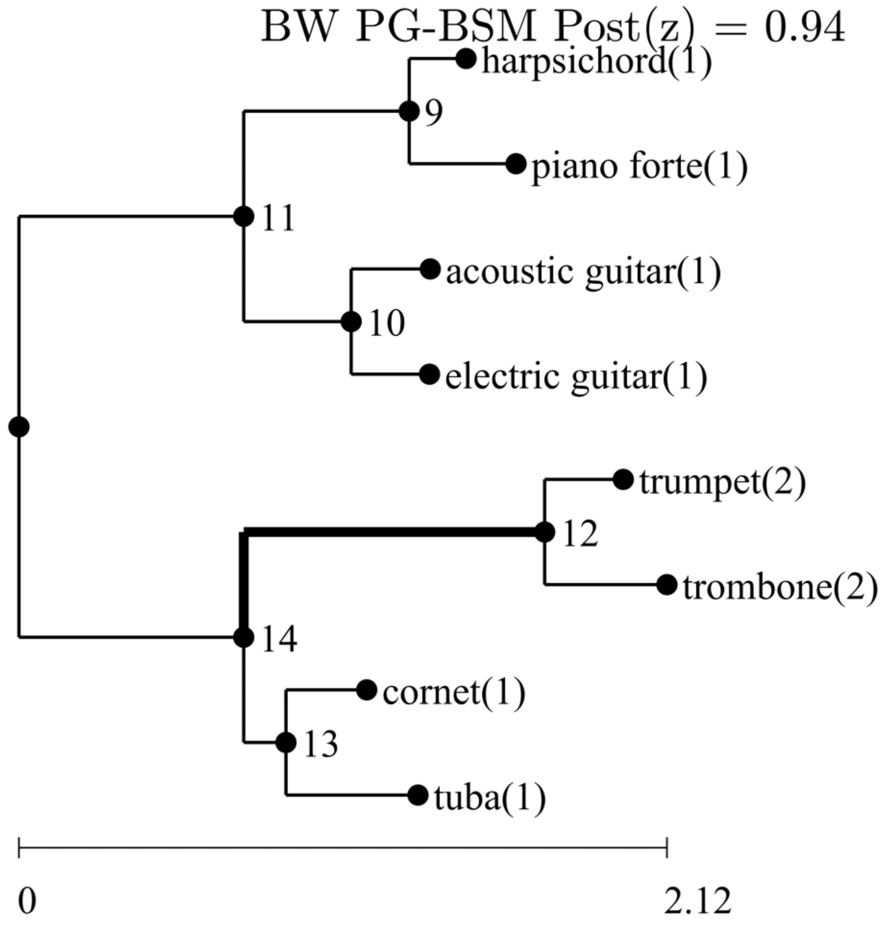


Figure 6: Estimates under the CW PG-BSM.

Numbers next to each taxon name indicated the assigned phenotype.

Bold branches indicate those over which the phenotype most likely changed (indicating the most likely history of the phenotype z).

Post(z) is the posterior probability of z.

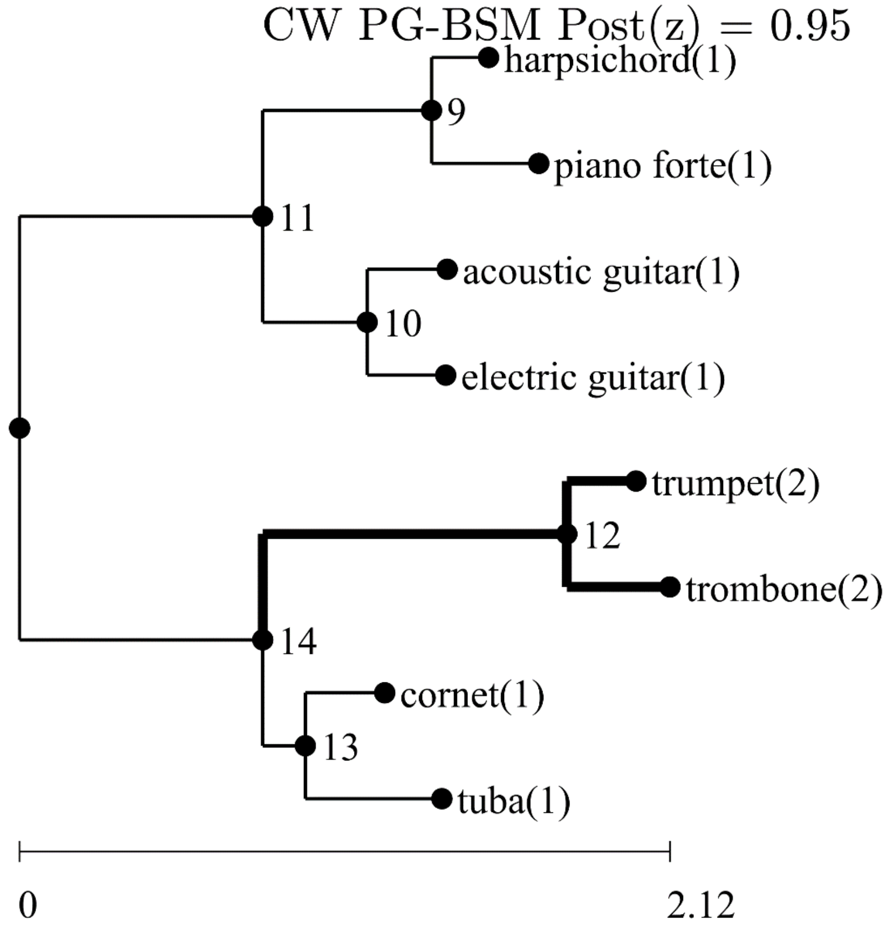
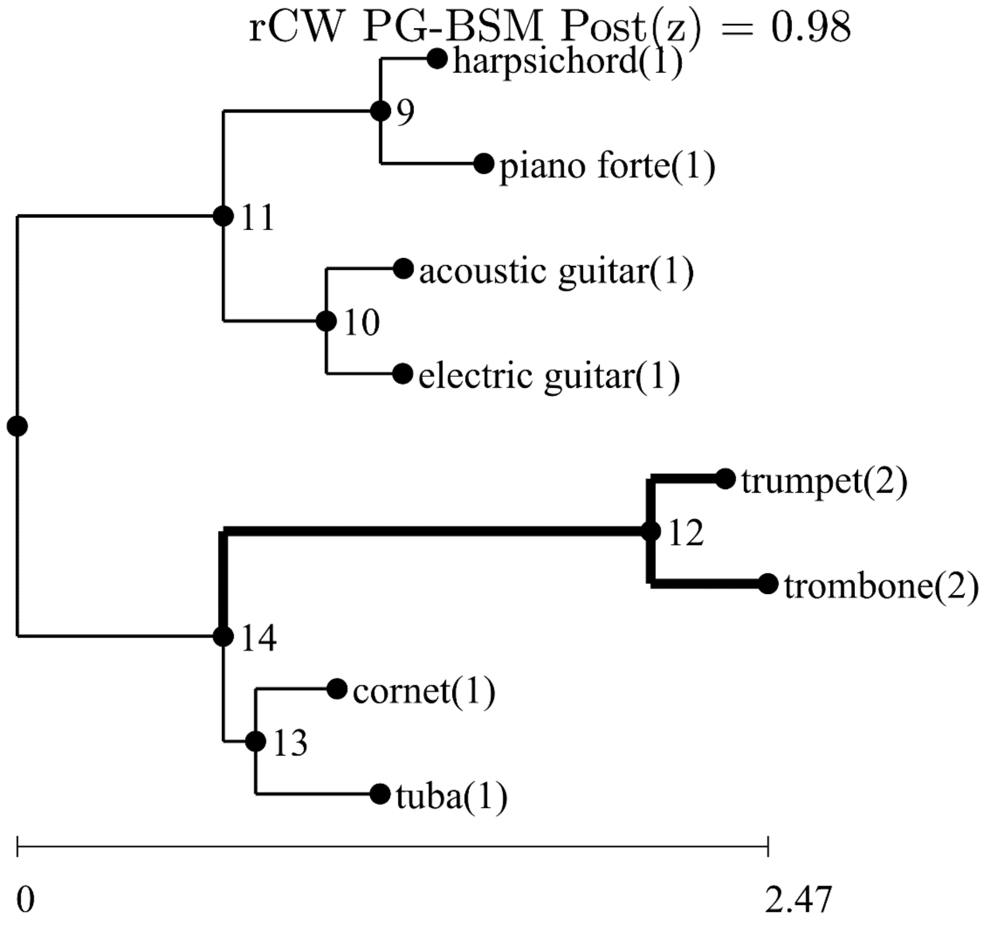


Figure 7: Estimates under the rCW PG-BSM.

Numbers next to each taxon name indicated the assigned phenotype.

Bold branches indicate those over which the phenotype most likely changed (indicating the most likely history of the phenotype z).

Post(z) is the posterior probability of z.



Three text files BW.txt, CW.txt, and rCW.txt contain site patterns inferred to have undergone the BW, CW and rCW process after application of false discovery count control to limit the count to 1 false discovery of each type. Sites consistent with the BW process are shown here.

GTA(V) ATC(I) AAC(N) TTA(L) CTC(L) harpsichord(1)

GTA(V) ATC(I) AAC(N) TTA(L) CTA(L) piano forte(1)

GTA(V) ATC(I) AAC(N) CTA(L) TTA(L) acoustic guitar(1)

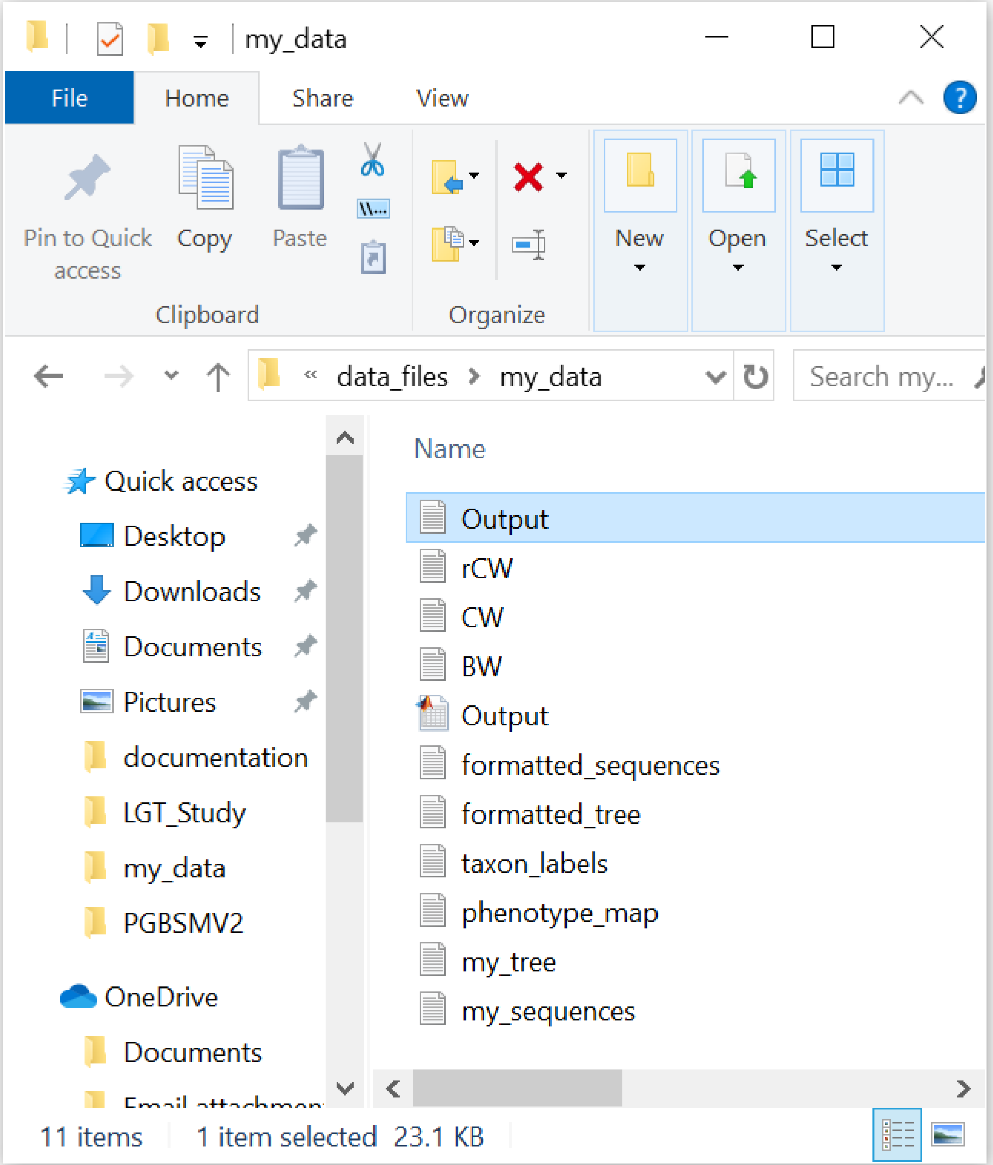
GTA(V) ATC(I) AAC(N) CTA(L) CTA(L) electric guitar(1)

TAC(Y) CAA(Q) GGC(G) GTT(V) GTC(V) trumpet(2)

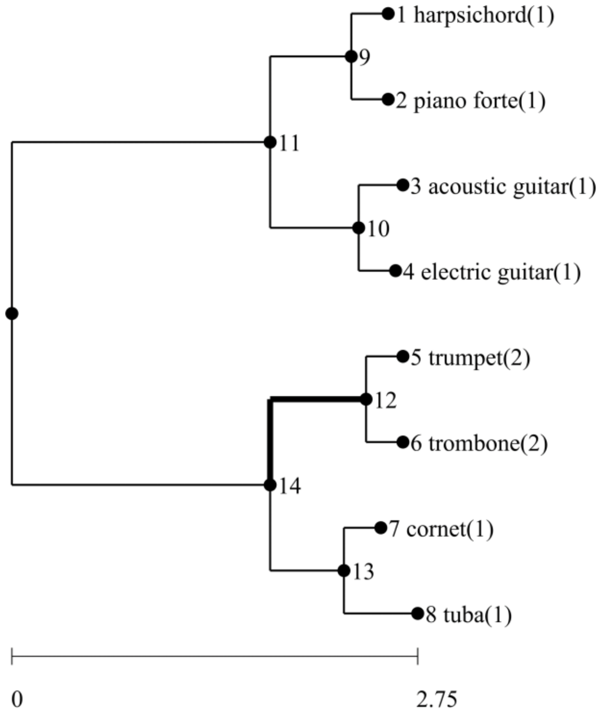
TAC(Y) CAA(Q) GGC(G) GTC(V) GTC(V) trombone(2)

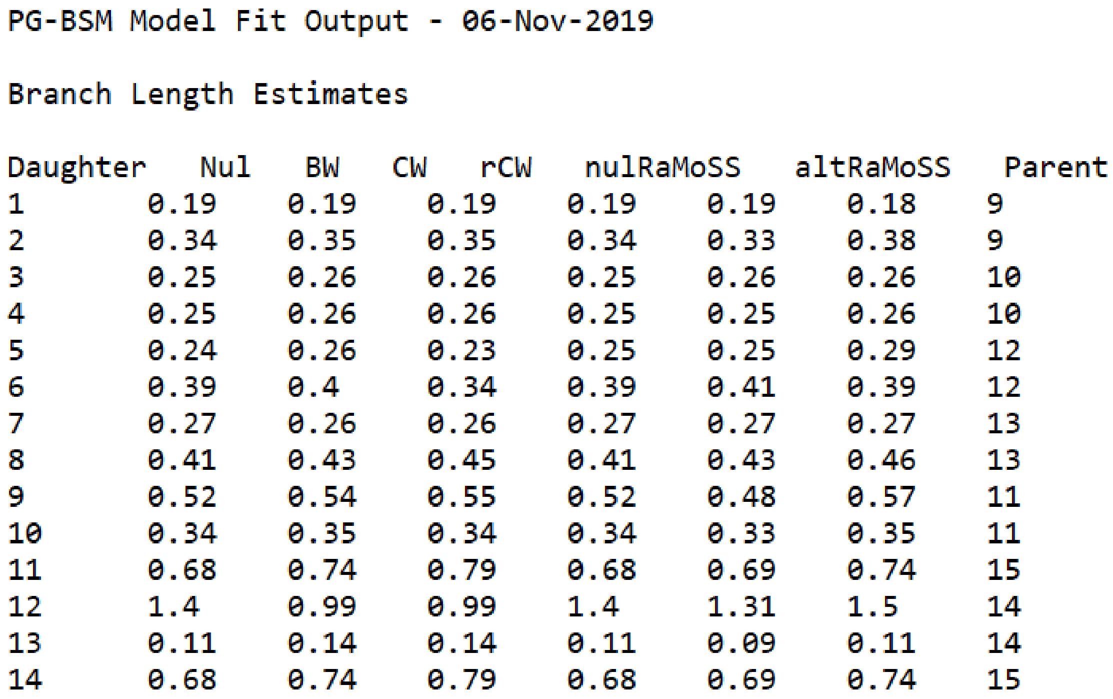
GTT(V) ATC(I) AAT(N) CTA(L) CTC(L) cornet(1)

GTC(V) ATC(I) AAC(N) CTA(L) CTT(L) tuba(1)



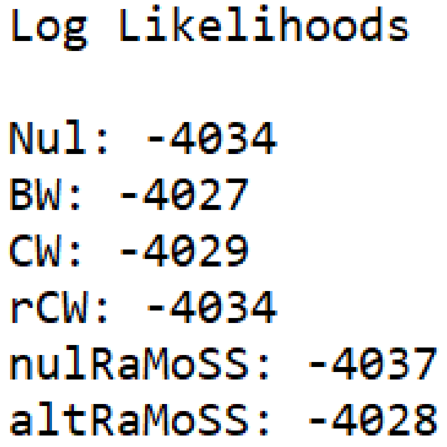
The file Output.txt contains the results of model fits. At the top of the file are branch length estimates for the six fitted model (the tree is shown on the left for reference but does not appear in the text file).



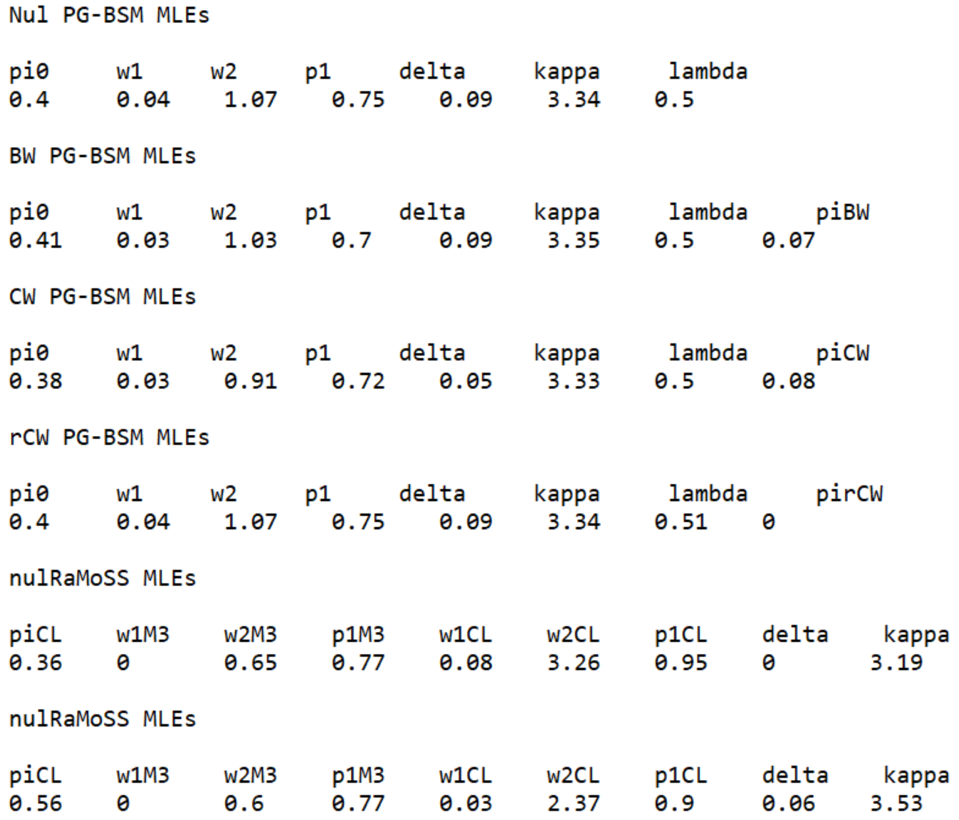


```
PG-BSM Model Fit Output - 06-Nov-2019

Branch Length Estimates

Daughter    Nul     BW     CW     rCW    nulRaMoSS    altRaMoSS    Parent
1         0.19   0.19   0.19   0.19   0.19   0.18   9
2         0.34   0.35   0.35   0.34   0.33   0.38   9
3         0.25   0.26   0.26   0.25   0.26   0.26   10
4         0.25   0.26   0.26   0.25   0.25   0.26   10
5         0.24   0.26   0.23   0.25   0.25   0.29   12
6         0.39   0.4    0.34   0.39   0.41   0.39   12
7         0.27   0.26   0.26   0.27   0.27   0.27   13
8         0.41   0.43   0.45   0.41   0.43   0.46   13
9         0.52   0.54   0.55   0.52   0.48   0.57   11
10        0.34   0.35   0.34   0.34   0.33   0.35   11
11        0.68   0.74   0.79   0.68   0.69   0.74   15
12        1.4    0.99   0.99   1.4    1.31   1.5    14
13        0.11   0.14   0.14   0.11   0.09   0.11   14
14        0.68   0.74   0.79   0.68   0.69   0.74   15
```

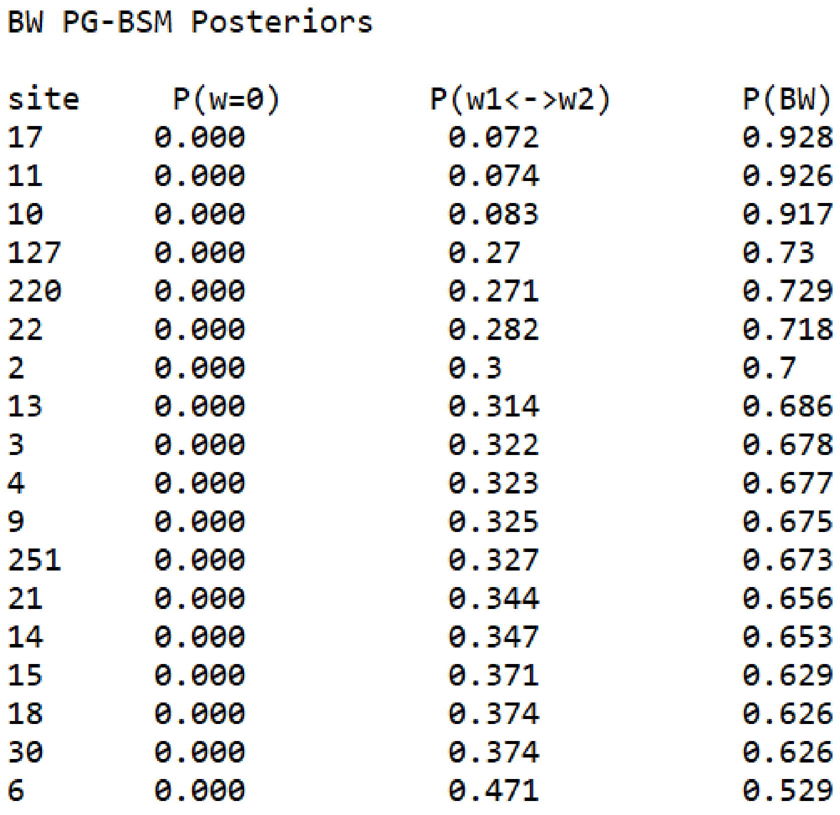
The next two sections show log-likelihoods and maximum likelihood estimates for all parameters apart from branch lengths. See Jones et al. 2017, 2019 for an explanation of all model parameters.



```
Log Likelihoods

Nul: -4034
BW: -4027
CW: -4029
rCW: -4034
nulRaMoSS: -4037
altRaMoSS: -4028
```



```
Nul PG-BSM MLEs

pi0     w1      w2      p1      delta     kappa     lambda
0.4     0.04     1.07    0.75    0.09      3.34    0.5

BW PG-BSM MLEs

pi0     w1      w2      p1      delta     kappa     lambda     piBW
0.41    0.03     1.03    0.7     0.09      3.35    0.5      0.07

CW PG-BSM MLEs

pi0     w1      w2      p1      delta     kappa     lambda     piCW
0.38    0.03     0.91    0.72    0.05      3.33    0.5      0.08

rCW PG-BSM MLEs

pi0     w1      w2      p1      delta     kappa     lambda     pirCW
0.4     0.04     1.07    0.75    0.09      3.34    0.51     0

nulRaMoSS MLEs

piCL    w1M3    w2M3    p1M3    w1CL    w2CL    p1CL    delta    kappa
0.36    0       0.65    0.77    0.08    3.26    0.95    0       3.19

nulRaMoSS MLEs

piCL    w1M3    w2M3    p1M3    w1CL    w2CL    p1CL    delta    kappa
0.56    0       0.6     0.77    0.03    2.37    0.9     0.06    3.53
```



```
BW PG-BSM Posteriors

site     P(w=0)         P(w1<->w2)       P(BW)
17      0.000            0.072           0.928
11      0.000            0.074           0.926
10      0.000            0.083           0.917
127     0.000            0.27            0.73
220     0.000            0.271           0.729
22      0.000            0.282           0.718
2       0.000            0.3             0.7
13      0.000            0.314           0.686
3       0.000            0.322           0.678
4       0.000            0.323           0.677
9       0.000            0.325           0.675
251     0.000            0.327           0.673
21      0.000            0.344           0.656
14      0.000            0.347           0.653
15      0.000            0.371           0.629
18      0.000            0.374           0.626
30      0.000            0.374           0.626
6       0.000            0.471           0.529
```

The last three sections list the posterior probabilities for the BW PG-BSM, the CW PG-BSM and the rCW PG-BSM sorted in descending order.

Here, for example, sites with the highest posterior probability P(BW) appear at the top of the list (results for the first 18 of 300 sites are shown).

P(w=0) is the posterior probability that the site evolved with the dN/dS rate ratio w = 0.

P(w1<->w2) is the posterior probability that the site evolved under the covarion-like process with random switching between w1 < w2.

P(BW) is the posterior probability that the site evolved under the BW process in association with changes in the phenotype.

Final Notes:

1. The PG-BSM has been tested under a variety of simulation scenarios and using real data (see the draft of paper to appear in Syst. Biol., Jones et al. 2019 included in the folder called documentation).

2. The most complex data used in the cite paper includes 45 taxa with as many as four discrete phenotypes (that's the cytochrome B data with phenotype equated to aquatic environment).

3. The model should perform well with similar data sets.

4. However, the model may or may not perform well when fitted to larger alignments, to alignments with many different phenotypes, or in cases where the pattern of phenotypes implies complex processes such as reversions.

If you have any difficulties running the code, please contact the code author:

cjones2@dal.ca