SVDQuartets Tutorial for Molecular Evolution Workshop 2022

David Swofford and Laura Kubatko

3 June 2022

Preliminaries

start an interactive session on LONI: srun --time 3:00:00 --ntasks=8 --nodes=1 --account=loni_molec_evo --partition=single --pty /bin/bash

In the instructions which follow, the "$" represents the Unix prompt. For example, if the instruction says:

```
$ cat filename.txt
```

you would type "cat filename.txt" and hit return. Likewise, "paup>" represents the prompt used by the PAUP* program.

First, make sure that your home directory is the current directory.

```
$ cd ~
```

If you have not already done so, install FigTree on your local machine (http://tree.bio.ed.ac.uk/software/figtree/). (FigTree is not a critical component of the tutorial, however, so don't sweat it if you run into problems.)

Download the materials for this practical and expand the archive:

tutorial.zip

```
$ wget phylosolutions.com/tutorials/wh2022-svdq-astral/svdquartets_tutor
$ unzip svdquartets_tutorial.zip
```

Alternatively, you can copy the folder from our class directory to your home directory:
cp -r /project/sackettl/MolEvol/SVDQ/ /home/yourname/

Move to the directory you just created:

```
$ cd svdquartets_tutorial
```

The programs we will use in this practical are installed on the MBL cluster, and during the workshop you should ignore the following paragraph.

After the workshop, you can run the tutorial locally on your own machines by downloading and installing PAUP* (http://phylosolutions.com/paup-test/), ASTRAL (https://github.com/smirarab/ASTRAL/blob/master/astral-tutorial.md#installation), and RAxML (https://sco.h-its.org/exelixis/web/software/raxml/). Mac and Windows users should use the command-line (console) version of PAUP* rather than the GUI version. The tutorial

assumes that PAUP*, ASTRAL, and RAxML can be started on your system using the commands **paup**, **astral**, and **raxmlHPC**, respectively. It will be easiest to set up symbolic links (symlinks) for PAUP* and RAxML; if you do not do this, you may need to modify the commands needed to start the programs from what is given in the tutorial instructions below. If you do not know how to create a symbolic link, "use the google." For ASTRAL, Linux/macOS users can create a script containing the following two lines, substituting appropriately for /path/to/astral and modifying the ASTRAL version number if necessary:

```
#!/bin/sh
java -jar /path/to/astral/astral.5.6.3.jar  $@
```

Windows users would need to create a batch file that runs the "java -jar" command above and passes all arguments from the command line analogously to the " $@" (again, google it).

Part 1: Analysis of a data set simulated from the "anomaly zone"

before launching PAUP, type: module load raxml/8.2.11/intel-19.0.5

We will begin with an example used by Liu and Edwards (2009), calling attention to problems that can arise due to gene trees for individual loci conflicting with the species tree because of "incomplete lineage sorting" (ILS). When internal branches of the species tree are very short (or effective population sizes are extremely large), the most probable gene tree can be inconsistent with the species tree, which can cause "concatenation" methods for inferring the tree to fail. See Kubatko and Degnan (2007), Liu and Edwards (2009), and Roch and Steel (2015) for details.

The data file **anomaly_zone.nex** represents one replicate of a simulation described in Fig. 2 in Liu and Edwards (2009). This file contains data for 10,000 loci, with 500 sites per locus. We will use the PAUP* program (http://paup.phylosolutions.com) to analyze this data set (PAUP* will subsequently be referred to as PAUP, without the asterisk). It is often convenient to run PAUP from a Nexus-file script containing the commands used to perform the analysis, but we will usually issue the commands interactively here for pedagogic reasons.

Start PAUP and load the data file.

/project/sackettl/MolEvol/software/paup4a168_ubuntu64 anomaly_zone…

```
$ paup anomaly_zone.nex -L anomaly.log
```

The first command argument above is the name of the data file, and the "-L" option opens a log file that will preserve all of the output of your run (you can use any name you like).

We'll begin with a "concatenated" maximum-likelihood analysis to demonstrate the problem. In order to perform ML in PAUP, you first need to estimate a substitution model, which specifies the rate of substitution from each nucleotide to each other nucleotide, the equilibrium base frequencies, and the distribution of rate variation across sites. PAUP provides a tool to automatically compare models using the AIC or BIC model selection criteria. A reasonable starting tree is needed (but the details of the tree topology have little effect on the chosen model). We'll just do a quick neighbor-joining analysis using LogDet

distances (which are a good general purpose distance when you don't know what else to use):

```
paup> dset distance=logdet;
paup> nj;
paup> automodel;
```

Using the default options, *automodel* will evaluate a set of 56 models and choose the one with that minimizes the AIC. On completion, the model parameters will be set to their maximum-likelihood estimates, and we can use this as the starting point for a maximum-likelihood search:

```
paup> set criterion=likelihood;
paup> hsearch;
```

(Note that you can abbreviate PAUP commands as long as they do not become ambiguous. For instance, instead of the above, we could have instead used `set crit=like; hs;`)

One way to see the tree found by the ML search is to use the *describetrees* command, with branch lengths drawn proportionally to the estimated number of substitutions/site:

```
paup> describe/plot=p;
```

Note the relationships among the taxa implied by the topology of this tree. Now we'll do an *svdquartets* analysis (Chifman and Kubatko, 2014, 2015) for the same data. For any command in PAUP, you can see the available options by typing "*command-name* ?;"

Before we start our next analysis, look at the options available for the *svdquartets* command:

```
paup> svdquartets ?;
```

For example, you can see that it is possible to evaluate quartets exhaustively ("evalquartets=all") or sample quartets randomly ("evalquartets=random nquartets=*n*"). Typically, you will leave most options set at their default settings, and enter values only for those you wish to override. One exception is that the current default setting for *evalquartets* is *random* (this may change in a future version), so we will explicitly request evaluation of all possible quartets. We'll also do a bootstrap analysis to assess confidence in the result:

```
paup> svdq evalq=all bootstrap;
```

The analysis finishes quickly and outputs the estimated tree, which is the one compatible with all 5 of the inferred quartet relationships. In fact, this is the correct tree that generated the data (see Liu and Edwards, 2009). The concatenated ML method estimated the tree incorrectly despite the large number of sites (500,000). Write down the bootstrap proportions for the two internal nodes for discussion later.

ASTRAL (Mirarab et al., 2014, 2015) is a "summary" method for estimating species trees from a set of gene trees estimated for each locus. Although it is very fast once the gene trees have been estimated, the preliminary phase of gene tree estimation (outside of ASTRAL) can take a long time. We'll use the RAxML program (https://sco.h-its.org/exelixis/web/software/raxml/index.html) to estimate these trees (PAUP could also be used to estimate the trees, but if you have a data set with many tips, RAxML is much faster, and this example will show you how to do it.)

PAUP now provides a wrapper for RAxML that simplifies the process of obtaining the gene trees. I wrote a small Python script to generate the PAUP commands; take a quick look at it:

```
paup> !cat  astral_prep.py
```

Commands typed at the command line beginning with "!" are passed to a Unix shell, so this command just shows the contents of the file on your terminal. The Python program just loops through the loci and outputs commands that analyze the data for each locus.

You will remain in the shell until you hit the return key at the "!" prompt, so do this now. Now run the python script from the "paup>" prompt to generate a Nexus command file, and hit return to go back to PAUP.

```
paup> !python2 astral_prep.py > run_astral.nex
```

The created Nexus file simply defines a "character partition" specifying the site numbers for each locus and runs a set of three commands for each of the 10,000 loci, e.g.:

```
include loci.locus_130/only;
raxml;
savetree file=az_tree.tre format=newick append;
```

For each locus, all of the sites except those belonging to that locus are excluded, RAxML is executed using PAUP's current model settings, and the resulting tree is appended to a growing treefile that will ultimately contain one tree for each locus.

Now run the Nexus command file generated by the Python script (you will be warned that you are going to reset the active data file; this is fine). Running the full analysis would take several hours, so read ahead rather than waiting for the command to complete.

```
paup> execute run_astral.nex;
```

Abort the command by typing **ctrl-C** after a few trees have been generated. Scroll back in the output and notice how different the trees are from one locus to the next. This variation comes from two sources: (1) the true variation in gene trees coming from the multispecies coalescent process, and (2) error in estimating the gene trees using a relatively small amount of sequence information per locus.

We will pretend that the analysis actually completed, and then quit the program.

```
paup> quit;
```

As in all good cooking shows, I have pre-baked the result so that we can pull it out of the oven immediately. It is stored in the file **az_10000_trees.tre**, which contains one tree for each of the 10,000 trees, written in the "Newick" (nested parentheses) format. It looks like this:

(A,(B,(D,E)),C);
(A,((B,D),E),C);
(A,((B,D),C),E);
(A,(B,(D,E)),C);
(A,((B,E),D),C);
.
.
.

We'll now run this file in ASTRAL. First, look at the options available:

ASTRAL runs with java, so type: module load jdk/1.8.0_222/intel-19.0.5

```
$ astral --help
```

For now, we will simply use the default settings to estimate a species tree, using the treefile we generated above as input. your command (run from your /home) will be:

java -jar /project/sackettl/MolEvol/software/Astral/astral.5.7.8.jar -i az_10000_trees.tre

```
$ astral -i az_10000_trees.tre
```

The analysis runs quickly and outputs a few lines of output. The estimated species tree is found near the end of the output, and looks something like this:

```
(A,(B,(C,(E,D)1:0.025494729020277836)1:0.033915343495063886));
```

Like SVDQuartets, ASTRAL infers the (unrooted) species tree correctly. Note the line near the top: "All output trees will be *arbitrarily* rooted at A". Both the input to and output from ASTRAL are unrooted trees; the rooting implied by the result is meaningless. In our case, the outgroup is tip E, so we would need to reroot the tree appropriately for a figure.

To view the tree, copy the output tree description the clipboard (e.g., ctrl-C), launch FigTree and paste the clipboard contents (e.g., ctrl-V). Select the terminal branch leading to tip E, and click the Reroot tool.

If you want to gain more familiarity with ASTRAL (after our practical session ends), you can run the tutorial at https://github.com/smirarab/ASTRAL/blob/master/astral-tutorial.md. On the MBL cluster, you can use the command "astral" rather than "java -jar astral"5.6.3.jar" as given in the tutorial.

Part 2: Analysis of a real data set

The data set is for members of the family Canidae (dogs, wolves, coyotes, foxes, jackals, etc.). Lindblad-Toh et al. (2005) sequenced a collection of 16 nuclear loci. The alignments used here are from a *BEAST2 tutorial written by Huw Ogilvie (https://github.com/genomescale/starbeast2/releases/download/v0.14.0/StarBEAST2-tutorial.zip). I reformatted them into the Nexus format and renamed the taxa from scientific to common names.

Start PAUP and load the canid sequence data file:

```
$ paup canids.nex -L canids.log
```

This file contains data for two individuals from each species, so we use a taxon-partition to assign each individual tip to a species. This definition is already in the file and you do not need to type it here. It looks like this:

```
    taxpartition species =
        sidestriped_jackal:  sidestriped_jackal_a sidestriped_jackal_b,
        African_golden_wolf: African_golden_wolf_a African_golden_wolf_b
        coyote:              coyote_a coyote_b,
        gray_wolf:           gray_wolf_a gray_wolf_b,
        blackbacked_jackal:  blackbacked_jackal_a blackbacked_jackal_b,
        Ethiopian_wolf:      Ethiopian_wolf_a Ethiopian_wolf_b,
        Dhole:               Dhole_a Dhole_b,
        African_wild_dog:    African_wild_dog_a African_wild_dog_b
        ;
```

NOTE: The version of PAUP* running on the XSEDE cluster has a bug that frequently causes crashes when multithreading is used with SVDQuartets. (I think this is a long-standing bug that were are just noticing on very fast CPUs.) To avoid this problem, we will specify "nthreads=1" to suppress multithreading when running SVDQuartets. This is only a temporary limitation, and I will remove this notice once the bug is fixed.

Perform an SVDQuartets analysis, referencing this taxon partition:

```
paup> svdq nthreads=1 evalq=all taxpartition=species;
```

Notice that the (North American) gray wolf (*Canis lupus*) joins with the African golden wolf (*Canis anthus*) rather than the (North American) coyote (*Canis latrans*), unlike the result of the *BEAST tutorial (see StarBEAST2-tutorial.pdf. However, this result is not well supported, as we can see by doing a multilocus nonparametric bootstrap, which resamples (with replacement) both loci and sites within loci:

```
paup> svdq nthreads=1 evalq=all taxpartition=species bootstrap=multilocu
```

paup> svdq nthreads=1 evalq=all taxpartition=species bootstrap=multilocus loci=combined;

(The "combined" above is the name of the character partition defining genes in the input file.) Examining the output bootstrap consensus tree, we see that the bootstrap support for the (gray wolf, African golden wolf) grouping is weak, and many bootstrap runs will actually find the (gray wolf, coyote) grouping that is probably correct. Interestingly, the DensiTree plot in the *BEAST tutorial also demonstrates that support for this grouping is very weak.

When you reach this point, quit PAUP (so that it will forget its current settings and we can start fresh):

```
paup> quit;
```

When you get to this point, take a break. We'll sync up here before starting the last part.

Part 3: Doing your own simulations

PAUP* now supports simulating data under the multispecies coalescent model. We will use the file **sim-6tips.nex**, and I will walk you through the components of this file:

```
#NEXUS

begin taxa;
    dimensions ntax=6;
    taxlabels A B C D E F;
end;

begin trees;
    tree 1 = [&R] (A:4.2,((B:1.0,C:1.0):2.2,(D:3.1,(E:3.0,F:3.0):0.1):0.
end;

begin paup;
    showtrees/userbrlens;
end;

begin dnasim;
    simdata multilocus=y nloci=100 nsitesperlocus=200;
    truetree source=memory treenum=1  units=2Ngen
             scalebrlen=1
             mscoal=y Ne=100000 mu=1e-8
             showtruetree=brlens showgenetrees=n storetruetrees=n
             seed=1;
    lset model=jc nst=1 basefreq=eq;    [= Jukes-Cantor model]
    beginsim nreps=100 seed=0;
        svdq nthreads=1;
        tally 'svdq';
    endsim;
end;
```

The copy of this file that you have in your directory will not run initially, because we have replaced the *nloci*=100, *nsitesperlocus*=200, and *scalebrlen*=1 with "@" symbols. Each of you will perform one (or more) simulation(s) by substituting values for the number of loci, number of sites per locus, and branch-length scaling factor. You should choose either 1 or 200 for *nsitesperlocus*. If you choose 1 site/locus, choose a value for *nloci* from the set {1000, 10000, 100000, or 1000000). If you choose 200 sites/locus, choose a value for *nloci* from the set {50, 100, 1000, or 10000}. You will now run 3 simulations, with branch-length scaling factors of 0.1 (high ILS), 1 (moderate ILS), and 10 (low ILS). *(NOTE: the 10000 loci x 200 sites/locus combinations will take a while, so randomly choose one of the 3 scaling factors for your first simulation so that if you run out of time we'll collectively have results from all 3 ILS levels.)*

Using a text editor, replace each "@" symbol with your assigned value. If you are not already comfortable using a Unix text editor, I recommend using "nano" (refer to the Unix tutorial). You can also edit using a local text editor such as BBEdit that allows editing of remote files via an SFTP connection, or you can download the file, edit it, and re-upload.

After you have modified and saved the file, execute it in PAUP:

```
$ paup sim-6tips.nex
```

As your runs complete, enter the results into the Google Sheet here: ~~SVDQuartets results~~.
https://docs.google.com/spreadsheets/d/1SQrroAGjymXB_Yi-Q0LPh9a7ze5ojbQoiSesRyqW2Ak/edit?usp=sharing
Feel free to run as many simulation conditions as you like, entering the results from each into the spreadsheet. We'll discuss the results once most people have entered their numbers.

References

Chifman J., Kubatko L.S. 2014. Quartet inference from SNP data under the coalescent model. 30:3317–3324.

Chifman J., Kubatko L.S. 2015. Identifiability of the unrooted species tree topology under the coalescent model with time-reversible substitution processes, site-specific rate variation, and invariable sites. J Theor Biol. 374:35–47.

Kubatko L.S., Degnan J.H. 2007. Inconsistency of phylogenetic estimates from concatenated data under coalescence. Syst Biol. 56:17–24.

Lindblad-Toh, K., Wade, C. M., Mikkelsen, T. S., Karlsson, E. K., Jaffe, D. B., Kamal, M., Clamp, M., Chang, J. L., Kulbokas, E. J., Zody, M. C., et al. 2005. Genome sequence, comparative analysis and haplotype structure of the domestic dog. Nature, 438(7069): 803–819.

Liu L., Edwards S.V. 2009. Phylogenetic analysis in the anomaly zone. Syst Biol. 58:452–460.

Mirarab S., Reaz R., Bayzid M.S., Zimmermann T., Swenson M.S., Warnow T.J. 2014. ASTRAL: genome-scale coalescent-based species tree estimation. Bioinformatics 30:i541–i548.

Mirarab S., Warnow T.J. 2015. ASTRAL-II: coalescent-based species tree estimation with many hundreds of taxa and thousands of genes. Bioinformatics 31:i44–i52.

Roch, S. and M. Steel. 2015. Likelihood-based tree reconstruction on a concatenation of aligned sequence data sets can be statistically inconsistent. Theoret. Pop. Biol. 100:56-62.