Genetic and genomic analyses using RAD-seq and Stacks

de novo assembly of RAD tags without a genome for phylogenetic analysis

Instructors:

Julian Catchen <jcatchen@illinois.edu>
Department of Animal Biology, University of Illinois at Urbana-Champaign
William Cresko <wcresko@uoregon.edu>
Institute of Ecology and Evolution, University of Oregon

Datasets and Software

- **Data sets - All are produced using an Illumina HiSeq 2500 sequencer**
  - ***Dataset 6 (DS6)*** - This is a set of population genomic data from several *Danio* species. The dataset comprises 1 individual from each of 15 species, 13 *Danios* and two outgroups. These data are from McClusckey and Postlethwait, 2015.

- **Software - All are open source software**
  - ***Stacks*** **(http://catchenlab.life.illinois.edu/stacks/)** - A set of interconnected open source programs designed initially for the *de novo* assembly of RAD sequences into loci for genetic maps, and extended to be used more flexibly in studies of organisms with and without a reference genome.
  - ***RAxML*** **(**http://sco.h-its.org/exelixis/web/software/raxml/index.html**)** - A software program written by the Exelixis Lab for the construction of maximum likelihood phylogenetic trees.

Exercise IV. Building a RADseq-based Phylogenetic Tree

1. In this exercise we will be working on a set of RAD-seq data taken from a number of *Danio* species, and two outgroups to *Danio*. We want to understand how these species relate to one another phylogenetically by examining variable sites between the genomes of these 15 species. The data set includes one sample per species. Our goal is to process the raw RAD data and reconstruct loci shared across the species. We will then feed these loci to the phylogenetic software RAxML to build our tree. *For more information on the study this data originated with, see McCluskey, et al. 2015, listed at the end of this document.*

Species	
Danio rerio (AB)	*Danio feegradei*
Danio rerio (Tubigen)	*Danio kerri*
Danio rerio (Nadia)	*Danio margaritatus*
Danio rerio (WIK)	*Danio nigrofasciatus*
Danio albolineatus	*Danio tinwini*
Danio choprae	*Devario aequipinnatus*
Danio danglia	*Microdevario kubotai*
Danio erythromicron	

2. There are a number of ways that GBS data can be analyzed for the construction of trees:
   - We can look at only fixed differences between species. These are sites that are fixed within each species, but variable among species. The most common way to do this is to export a set of concatenated, fixed differences in Phylip format (http://evolution.genetics.washington.edu/phylip/doc/sequence.html).
   - We can look at fixed and variable differences — this is an export of *all* variable sites concatenated together in Phylip format (variable sites are encoded in IUPAC format, https://en.wikipedia.org/wiki/Nucleic_acid_notation).
   - We can look at the fixed and variable differences and include the surrounding invariant sequence again, all concatenated together. This is equivalent to exporting the full sequence for each RAD locus that contains one or more variable sites in Phylip format.
   - We can look at the fixed and variable differences including the surrounding invariant sequence, but we can maintain each locus independently in the Phylip output. This allows us to pass a *partition* file to the phylogenetic software so that each locus can be analyzed independently.

3. In this study, we have only one sample per species. While this setup is standard in phylogenetics, in population genetics our goal is almost always to gather many representatives from each population as possible. **What are the implications for building phylogenetic trees from one of the above types of data when we only have one sample per species?**

4. In your `./working` workspace, create a directory called `phylo` to contain all the data for this exercise.

   Unarchive data set 6 (DS6):

   ```
   ~/workshop_data/stacks/phylo/danio_phylo.tar
   ```

   into the `samples` directory.

5. Create a new MySQL database called `danio_radtags` and populate the tables by loading the table definitions from:

   ```
   /usr/local/share/stacks/sql/stacks.sql
   ```

   If you view this file, you will see all the SQL commands necessary to create tables to hold our Stacks data. To create and populate the database we can feed these commands to the MySQL server:

   [Cutting+pasting the lines below may be difficult due to line breaks and non-standard quote characters (“)]

   ```
   % mysql --defaults-file=/usr/local/share/stacks/sql/mysql.cnf
          -e "CREATE DATABASE danio_radtags"
   % mysql --defaults-file=/usr/local/share/stacks/sql/mysql.cnf
          danio_radtags < /usr/local/share/stacks/sql/stacks.sql
   ```

   To access the MySQL server, we need a username and password. When Stacks was installed, we stored the username and password in the file:

   ```
   /usr/local/share/stacks/sql/mysql.cnf
   ```

   which we are passing to the MySQL client program to create our database.

   Note: as a convention we append “`_radtags`” as a suffix onto all our RAD-tag databases. This is not necessary from a technical point of view, but the Stacks web interface will only show databases with this suffix (in case you have other, unrelated MySQL databases on your server).

6. Run the Stacks’ `denovo_map.pl` pipeline program. This program will run `ustacks`, `cstacks`, and `sstacks` on the individuals in our study.

   [Once you get `denovo_map.pl` running, it will take approximately 30 minutes.]

   - Information on `denovo_map.pl` and its parameters can be found online:
     - http://catchenlab.life.illnois.edu/stacks/comp/denovo_map.php

- We want Stacks to understand which individuals in our study belong to which population. To specify this, create a file in the working directory called *popmap*, using an editor. The file should be formatted like this:

  `<sample file prefix><tab><population ID>`

  Include all 15 samples in this file and specify and assign each individual a different population ID (an abbreviation of the species name works well). You must supply the population map to `denovo_map.pl` when you execute it.

- There are three additional important parameters that must be specified to `denovo_map.pl`, along with each of the individuals in our data set.
  - Set the `stacks` directory as the output, and set the three main parameters: *minimum stacks depth*, *distance allowed between stacks*, and *distance allowed between catalog loci*. Specify the database you just created and, finally, specify 1 as the batch number.
- You will need to specify each individual in the dataset with '`-s`'. Since there are a number of individuals it will be helpful to create the command ahead of executing it. There are several ways you could do this:
  1. Edit the command together in an external editor, say on your local laptop, then cut and paste the command into the terminal window.
  2. If you know how, create a shell script containing the command using a UNIX editor such as emacs, vi, or nano.
  3. Simply type the command out. This will take some time, but you can use tab-completion to make it easier.
- Execute the Stacks pipeline.

7. Examine the *Stacks* log and output files when execution is complete.

- From the log: how are the different programs, `ustacks`, `cstacks`, and `sstacks` executed?
- How many reads are used in each `ustacks` execution?
- Examine the output of the populations program in the log.
  - How many loci were identified?
  - How many were filtered and for what reasons?
- Familiarize yourself with the output of each Stacks' component:
  - `ustacks`: *.tags.tsv, *.snps.tsv, *.alleles.tsv
  - `cstacks`: batch_1.catalog.tags.tsv, batch_1.catalog.snps.tsv, batch_1.catalog.alleles.tsv
  - `sstacks`: *.matches.tsv
  - `populations`: *.sumstats.tsv, *.sumstats_summary.tsv

8. View the result of the Stacks analysis through the web interface:

- http://<Amazon Instance>/stacks/
- Explore the web interface
  - Why are some markers found in more samples?
  - Set the filters so that there are no fewer than 2 SNPs and no more than 5 SNPs per locus and so that there are at least 10 matching individuals per locus.
  - Select a locus that has a reasonable ratio of genotypes (depending on your parameter choices you may have slightly different loci compared with another run of the pipeline). Click on `Allele Depths` to view additional information.
  - Select a polymorphic sample, click on the alleles to see the actual stack that corresponds to the catalog locus.
    - Do any of the columns have a blue background? If so, why?
    - Why do some nucleotides in the stack have a yellow background?
    - What are the different roles played by primary and secondary reads?
- Compare the web interface data to the batch_1.haplotypes_1.tsv file.
  - Use the `cut` and `grep` commands to view all loci from a particular sample, and all samples from a particular locus.
  - Is the same data contained in both sources?

9. Now you need to run `populations` again with your population map and this time export the data for phylogenetic analysis.

You should export two Phylip data sets:

1. The first should maximize the number of taxa in the analysis — that is, you should select loci that are present in all loci in your tree.
2. The second should maximize the number of loci in the analysis — that is, you should require a minimum number of taxa that provides as many loci as possible for the analysis.

Think carefully about what is the best combination of taxa and loci and adjust the number of required populations parameter for the `populations` program (do not set this parameter too low or you might run out of memory on the Amazon Instance).

10. Run the RAxML software on each of your two data sets (the program is called `raxmlHPC` on the cluster) and choose a GTRCAT model and disable rate heterogeneity.

11. Download the resulting *best* tree in Newick format (https://en.wikipedia.org/wiki/Newick_format) from each run and visualize your trees on your local laptop.

- A good program for visualizing trees on your laptop in Newick format is FigTree (http://tree.bio.ed.ac.uk/software/figtree/).