Genetic and genomic analyses using RAD-seq and Stacks

de novo assembly of RAD tags without a genome for a STRUCTURE 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 5 (DS5)*** - This dataset comprises a subset of RAD-seq data generated from 30 individuals from three populations of threespine stickleback, each of which has been individually barcoded. The RAD-seq data were prepared using the restriction enzyme *SbfI*, and sequenced using an Illumina sequencer. These data are a component of the data originally published in Catchen, et al. 2013.

- **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. The pipeline has a Perl wrapper allowing sets of programs to be run. However, the software is modular, allowing it to be applied to many scenarios. You will use the Perl wrapper in class and the modules on your own.
  - ***Structure*** **(http://pritch.bsd.uchicago.edu/structure.html)** - A software program originally written by Jonathan Pritchard and colleagues that uses Bayesian stochastic models of multi-locus genotype data. The package was written to estimate the distribution and abundance of genetic variation within and among populations, patterns that are now commonly called the *genetic structure* of populations.

Exercise II. part 2: *de novo* assembly of RAD tags without a genome

1. In this second exercise we will be working on a set of threespine stickleback data sampled from throughout Oregon, on the west coast of the United States. These stickleback can be found in a number of habitats from costal marine and freshwater habitats, to inland river habitats, to high mountain lakes in the interior of Oregon. We want to understand how these populations relate to one another and in this exercise, you will examine three of these populations: a coastal marine population, a costal freshwater, and an inland river population. Without using a reference genome we will assemble loci and determine population structure using the Structure program as well as generating a distance based phylogenetic tree from F_{ST} values. *For more information on the study this data originated with, see Catchen, et al. 2012, listed at the end of this document.*

2. In your `./working` workspace, create a directory called `denovo` to contain all the data for this exercise. Inside that directory, create two additional directories: `samples`, and `stacks`. To save time, we have already cleaned and demultiplexed this data and will start from the cleaned samples stage.

   Unarchive data set 5 (DS5):

   ```
   ~/workshop_data/stacks/denovo/oregon_stickleback.tar
   ```

   into the `samples` directory.

3. Create a new MySQL database called `orphy_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 orphy_radtags"
   % mysql --defaults-file=/usr/local/share/stacks/sql/mysql.cnf
          orphy_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).

4. 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 30 samples in this file and specify which individuals belong to which populations. 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.

5. 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

6. 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 3 SNPs per locus and so that there are at least 20 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?

7. Our goal now is to export a subset of loci for analysis in *Structure*, which analyzes the distribution of multi-locus genotypes within and among populations in a Bayesian framework to make predictions about the most probable population of origin for each individual. The assignment of each individual to a population is quantified in terms of Bayesian posterior probabilities, and visualized via a plot of posterior probabilities for each individual and population. A key user defined parameter is the hypothesized number of populations of origin which is represented by **k**. Sometimes the value of **k** is clear from from the biology, but more often a range of potential **k**-values must be explored and evaluated using a variety of likelihood based approaches to decide upon the ultimate **k**. In the interest of time we won't be exploring different values of **k** here, but this will be a key step in any data analysis. In addition, at the moment *Structure* can only handle a small number of loci because of the MCMC algorithms involved in the Bayesian computations, usually many fewer than are generated in a typical RAD data set. We therefore want to randomly choose a random subset of loci that are well

represented in our three populations. Nonetheless, this random subset contains more than enough information to define population structure.

- The final stage of the `denovo_map.pl` pipeline is to run the `populations` program to calculate population genetic statistics for our data. We want to execute this program by hand again, specifying filters that will give us only the most well represented loci.
    - Run `populations` again, specifying that loci must be present in at least 80% of individuals in all three populations. You will have to tell `populations` where to find the output of the *Stacks* pipeline (this should be in the `stacks` output directory).
    - One final detail: *Structure* assumes that each SNP locus is independent, so we don't want to output multiple SNPs from the same RAD locus, since they are not independent but are in linkage blocks within each RAD tag. We can achieve this behavior by specifying the `--write_single_snp` parameter to `populations`.

8. Now we want to select 1,000 loci randomly from the results and save these loci into a file. We can easily do this using the shell given a list of catalog IDs output in the previous step. The `batch_1.sumstats.tsv` file gives a list of all polymorphic loci. Use the `cat`, `grep`, `cut`, `sort`, `uniq`, `shuf`, and `head` commands to generate a list of 1000 random loci. Save this list of loci as a *whitelist*, that we can feed back into `populations`. This operation can be done in a single shell command.

9. Now execute `populations` again, this time feeding back in the whitelist you just generated. This will cause `populations` to only process the loci in the whitelist. Specify that a Structure output file be included this time and again insist that only a single SNP is output from each RAD locus. Finally, you will need to again specify the population map that you generated above to `populations` so that this information is passed into the *Structure* output file.

10. Create a new directory called `structure` and copy the *Structure* output file that Stacks generated to this directory.

- Edit the *Structure* output file to remove the comment (first line in the file, starts with "`#`").
- The parameters to run *Structure* (including a value of **k=3**) have already been prepared, you can find them here:

```
~/workshop_data/stacks/denovo/mainparams
```

and

```
~/workshop_data/stacks/denovo/extraparams
```

Copy them into your working directory as well.

11. Execute *Structure*, saving the data into this new directory:

```
structure > structure/batch_1.structure.console
```

[A common STRUCTURE error happens when your population output contains less than 1000 loci. You may need to adjust the number of loci in the `mainparams` file to match your exact Stacks output.]

12. You will need to download the `batch_1.structure.console` and `batch_1.structure.out_f` files from the cluster. You can then load them into the graphical interface for *Structure* on your local computer. Select the "File" menu and then "Load structure results…" to load the Structure output. Choose the "Barplot" menu and then "Show".

- Are the three Oregon threespine stickleback populations related to one another? How can you tell?