Genetic and genomic analyses using RAD-seq and Stacks

Instructors:

Julian Catchen <jcatchen@illinois.edu>
  Department of Animal Biology, University of Illinois at Urbana-Champaign
Konrad Paszkiewicz <K.H.Paszkiewicz@exeter.ac.uk>
  Exeter Sequencing Service, Biosciences, University of Exeter

Objectives:

The goal of this exercise is to familiarize students with the use of next generation sequence data produced from Reduced Representation Libraries (RRL) approaches such as Restriction site Associated DNA (RAD-tags). These libraries are often used for genotyping by sequencing, and can provide a dense set of single nucleotide polymorphism (SNP) markers that are spread evenly across a genome. These markers are useful for a variety of genetic and genomic analyses in model and non-model organisms. Students will gain experience with a computational pipeline called *Stacks* that was designed for the analysis of such data. Data will be analyzed *de novo* to perform a population analysis without the aid of a reference genome, and from an organism with a reference genome to identify signatures of selection. *Stacks* can be used for other analyses of RAD-seq data as well, such as constructing genetic maps and phylogeography, although those are beyond the scope of this exercise.

Students will learn how to:

1. Prepare raw RAD Illumina data for analysis by removing low quality reads and de-multiplexing a set of barcoded samples using both sheared and double-digested data.
2. Use *Stacks* to assemble RAD-tags *de novo* from several populations of samples.
3. Call SNPs, genotypes, and haplotypes of these individuals within *Stacks*.
4. Export data from *Stacks* for analysis in Structure and build a distance-based phylogenetic tree.
5. Align RAD sequences against a reference genome.
6. Use *Stacks* to calculate population genetic statistics and plot these across the genome.

By the end of this workshop you will be expected to know how to:

7. Manipulate raw RAD Illumina data for analysis using a variety of different parameters.
8. Produce *de novo* assemblies using reads from an organism without a reference genome.
9. Align RAD tags against a reference genome to identify signatures of selection.
10. Extend what was learned to more complicated 'on your own' problems.

Introduction

The advent of short-read sequencing technologies has revolutionized the study of genomic variation from complete sequences in model organisms such as nematode worms, fruit flies, zebrafish, mice and humans. In addition, the long-standing dream of biologists of having complete genomic information from numerous individuals from different populations in the lab and wild, the field of **population genomics**, is becoming a possibility for a variety of ecological and evolutionary studies.

Until just a few years ago the goal of acquiring complete genomic information from numerous individuals in many populations was out of reach for all but a small number of model organisms. For example, producing a high density genetic map for an organism required an immense investment of resources to first produce and then type the large number of genetic markers needed to adequately cover the genome. Furthermore, identifying genomic regions associated with phenotypic variation, or involved in the adaptation of organisms to novel conditions, was restricted to organisms for which re-sequencing projects produced a dense battery of genetic markers at a significant cost.

The limits to population genomic studies will gradually fade as the costs of second generation sequencing continue to drop. However, many studies using complete genome re-sequencing will not be feasible for a while because costs are still significant, high quality read lengths are still too short, and analysis remains challenging. Luckily, many population genomic studies can now efficiently be performed by using an alternative approach called genotype-by-sequencing that occurs by the sequencing of reduced representation libraries (RRL), and subsequent identification and scoring of SNPs and inference of haplotypes. Although they do not provide complete genomic information, these approaches provide a sufficient picture via data on hundreds of thousands of SNPs and haplotypes spread across a genome at a fraction of the cost of complete re-sequencing.

We developed one such approach called **R**estriction-site **A**ssociated **D**NA sequencing (RAD-seq), which has been used to identify signatures of selection, produce high density genetic maps, help assemble genomes, and be useful for studies of allelic specific transcriptional profiling. Because these data are so new, and the sample sizes of sequences often so massive, a critical related breakthrough has been the development of algorithms and software pipelines for the analysis of such data. We have produced *Stacks* for the analysis of RAD-seq data. You will learn how to analyze RAD data with and without the use of a reference genome with the goal of identifying population structure in one case and identifying signatures of selection in a second case. Through the completion of these tasks you will learn how to process RAD-seq data and use the software programs *Stacks*, *GSnap and Structure*.

*For more information on RAD genotyping and related methods, in particular conceptual and statistical issues, see the papers listed at the end of this document. Many of these papers can be read outside of class. However, the first set of papers (***Core readings for the lecture and workshop)** *should be read during your time in Norwich. These papers will help you better understand both the molecular biology, computational analyses, and conceptual framework for the analysis of RRL data such as RAD.*

Datasets and Software

- <u>**Data sets - All are produced using an Illumina HiSeq 2500 sequencer**</u>
  - ***Dataset 1 (DS1)*** - This data set comprises just a small proportion of a lane of single-end standard RAD data.
  - ***Dataset 2 (DS2)*** - A fragment of a lane of paired-end RAD sequences that have been double-digested with two restriction enzymes.

    You will use these first two data sets to become familiar with the structure of RAD sequences, as well as to become proficient with the pre-processing (i.e. cleaning and de-multiplexing) of data before alignment or assembly.
  - ***Dataset 3a (DS3a)*** - This dataset consists of two samples of single-end RAD data from the same set of samples, but constructed in two different libraries and sequenced independently.
  - ***Dataset 3 (DS3)*** - 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.
  - ***Dataset 4 (DS4)*** - This is a set of population genomic data from the threespine stickleback. The dataset comprises 8 individuals from each of two differentiated populations, for a total of 16 barcoded individuals. The RAD data were prepared using the restriction enzyme *SbfI*, and sequenced using an Illumina sequencer. These data are unpublished, but similar to those published in Hohenlohe, et al. 2010.

- <u>**Software - All are open source software**</u>
  - ***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.
  - ***GSnap*** **(http://research-pub.gene.com/gmap/)** - *GSnap* is a very fast and efficient software package used for aligning sequences against a reference genome. We will use *GSnap* to align RAD reads against the stickleback reference genome, and then analyze these reads within the *Stacks* pipeline. Although we will use *GSnap* for this exercise, many other algorithms and software exist for aligning against a reference genome, and these could be used in conjunction with *Stacks* as well.
  - ***Samtools*** **(http://samtools.sourceforge.net)** - A suite of software tools designed to perform a variety of common tasks with next generation sequencing data tools. The SAM and BAM were developed associated.
  - ***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 1. Data preparation, part 1

1. 10 minute mini-lecture on Phred scores and the `process_radtags` cleaning algorithm.
2. The first step in the analysis of all next generation sequencing data, including RAD-seq data, is removing low quality sequences and separating out reads from different samples that were individually barcoded. This 'de-multiplexing' serves to associate reads with the different individuals or population samples from which they were derived.
3. In each exercise you will set up a directory structure on the remote server (in this case our TGAC Virtual Machine) that will hold your data and the different steps of your analysis. We will use the directory ~/`working` on the cloud to hold these analyses. Be careful that your are reading and writing files to the appropriate directories within your hierarchy. You'll be making many directories, so stay organized!
   - Each step of your analysis goes into the hierarchy of the workspace, and each step of the analysis takes its input from one directory and places it into another directory, this is known as a '**waterfall workspace**'. We will name the directories in a way that correspond to each stage and that allow us to remember where they are. A well organized workspace makes analyses easier and prevents data from being overwritten.
   - In ~/`working`, create a directory called `clean` to contain all the data for this exercise. Inside that directory create two additional directories: `lane1` and `samples`. We will refer to the `clean` directory as the *working directory*.
   - Unarchive data set 1 (DS1):

     ```
     /data/clean/lane1.tar
     ```

     to the `lane1` directory.
   - You can copy the file to your working directory and use `tar` to unarchive it, or you can change to your working directory and *untar* it without moving the file (this will save you time and will dump the unarchived files into the directory you are currently in).
4. Your decompressed files has millions of reads in it, too many for you to examine in a spreadsheet or word processor. Examine the contents of the set of files in the terminal (the `head`, `more`, and `tail` commands may be of use).
   - You should see multiple different lines with different encodings.
     - How does the FASTQ file format work?
     - How are quality scores encoded? (See the link to quality scores in Appendix.)
     - How could you tell by eye which type of encoding your data are using?
   - Can you find strings of `B`s in the quality scores? What do these mean?
5. You probably noticed that not all of the data is high quality. In general, you will want to remove the lowest quality sequences from your data set before you proceed. However, the stringency of the filtering will depend on the final application. In

general, higher stringency is needed for *de novo* assemblies as compared to alignments to a reference genome. However, low quality data will almost always affect downstream analysis, producing false positives, such as errant SNP predictions.

6. We will use the Stacks's program `process_radtags` to clean and demultiplex our samples.

   - Take advantage of the manual page for `process_radtags` on the Stacks website to find information and examples:

     http://catchenlab.life.illnois.edu/stacks/comp/process_radtags.php

   - You will need to specify the set of barcodes used in the construction of the RAD library. Remember, each P1 adaptor in RAD has a particular DNA sequence (an inline barcode) that gets sequenced first, allowing data to be associated with samples such as individuals or populations.

   - Enter the following barcodes into a file called `lane1_barcodes` in your working directory (make sure you enter them in the right format):

     - `AAACGG    AACGTT    AACTGA    AAGACG`
     - `AAGCTA    AATGAG    ACAAGA    ACAGCG`
     - `ACATAC    ACCATG    ACCCCC    ACTCTT`
     - `ACTGGC    AGCCAT    AGCGCA`

   - Copy the remaining barcodes for this lane of samples from the file:

     `/data/clean/lane1_barcodes`

     and append them to your barcodes file in your working directory.

     - You can concatenate this file onto the end of your file using the cat command and the shell's append operator: `cat file1 >> file2`, or you can cut+paste.
     - Based on the barcode file, how many samples were multiplexed together in this RAD library? (The `wc` command can tell you this.)

   - You will need to specify the restriction enzyme used to construct the library (*SbfI*), the directory of input files (the `lane1` directory), the list of barcodes, the output directory (`samples`), and specify that `process_radtags` *clean*, *discard*, and *rescue* reads.

   - The `process_radtags` program will write a log file into the output directory. Examine the log and answer the following questions:

     - How many raw reads were there?
     - How many were retained?
     - Of those discarded, what were the reasons?

   - What can the list of "sequences not recorded" tell you about the data analyzed and about the design of barcodes in general?

7. Rename five of the output files in the samples directory to use more meaningful names:

```
sample_AAACGG.fq    indv_01.fq
sample_AACGTT.fq    indv_02.fq
sample_AACTGA.fq    indv_03.fq
sample_AAGACG.fq    indv_04.fq
sample_AAGCTA.fq    indv_05.fq
```

Renaming files by hand is time consuming as you can tell from just working with this subset. Renaming all the files to more meaningful names can be greatly simplified by writing a shell script in an editor, such as Emacs, and then using the search/replace function. Then, you can execute the shell script to actually rename the files.

Exercise 1. Data preparation, part 2

1. We will now work with the second data set. These data contain paired-end reads that have been double-digested and dual barcoded. Each set of paired reads contains an inline barcode on the first read, and an indexed barcode on both reads. These are known as *combinatorial barcodes* as many unique combinations can be made from pairs of barcodes.

- In ~/`working/clean`, create a directory called `lane2` to contain the raw data for this exercise and create the directory `ddsamples` to contain the cleaned output.
- Unarchive data set 2 (DS2):

```
/data/clean/lane2.tar
```

into the `lane2` directory.

2. Examine the contents of the pairs of files in the terminal again.

- How are the FASTQ headers related between pairs of files?
- Can you identify the indexed barcode in the FASTQ header?

3. We will again use the Stacks' program `process_radtags` to clean and demultiplex our samples.

- You will need to specify the set of barcode pairs used in the construction of the RAD library.
- Enter the following barcodes into a file called `lane2_barcodes` in your working directory (make sure you enter them in the right format):
  - `AACCA/ATCACG    CATAT/ATCACG    GAGAT/ATCACG`
  - `TACCG/ATCACG    AAGGA/CGATGT    CAACC/CGATGT`
  - `GACAC/CGATGT    TACGT/CGATGT`
- Copy the remaining barcodes for this lane of samples from the file:

```
/data/clean/lane2_barcodes
```

and append them to your barcodes file in your working directory.

    - You can concatenate this file onto the end of your file using the cat command and the shell's append operator: `cat file1 >> file2`, or you can cut+paste.
    - How many samples were multiplexed together in this RAD library? (The `wc` command can tell you this.)
- You will need to specify the two restriction enzymes used to construct the library (*NlaIII and MluCI*), the directory of input files (the `lane2` directory), the list of barcodes, the output directory (`ddsamples`) and specify that `process_radtags` *clean*, *discard*, and *rescue* reads.
- The `process_radtags` program will write a log file into the output directory. Examine the log and answer the following questions:
    - What is the purpose of the four different output files for each set of barcodes?
    - How many raw reads were there?
    - How many were retained?

4. For one of the sets of output files in the `ddsamples` directory rename the *.1.fq file to use a more meaningful name. Then, concatenate the other output files onto the end, so that for the set of files:

```
sample_AACCA-ATCACG.1.fq
sample_AACCA-ATCACG.2.fq
sample_AACCA-ATCACG.rem.1.fq
sample_AACCA-ATCACG.rem.2.fq
```

We are left with a single output file:

```
indv_01.fq
```

- Why are we able to concatenate all the data together from both the single and paired-end reads?
- How will Stacks interpret the single-end reads versus the paired-end reads?

On your own outside of class

1. Remind yourself of the use of shell tools and regular expressions in Unix:

- Using shell tools:
    - Identify all of the barcodes in one of the sequencing files.
    - Count the number of occurrences of each barcode.
- (`cut, grep, sort, uniq`, and the pipe "|" are the commands you need)
- Using a single shell command, extract out the restriction site from the single end reads of DS2 and print the distribution of restriction sites. Repeat this for the paired end reads. If you are experiencing lots of ambiguous RAD sites when running

`process_radtags`, this procedure can help you diagnose if your restriction enzyme site is intact.

- Determine the distribution of read lengths in your data set (you'll need to use `awk` in addition to `grep`). This is useful if you have variable length data, say from a MiSEQ machine and need to choose where to trim the data.

2. Write a shell script in Emacs to rename all the files in one execution.

- Use a shell loop to do the hard work.

3. Try using the process_radtags program with a range of parameters.

- Specify a different (incorrect) restriction enzyme
  - How many reads were retained this time?
  - Of those discarded, what were the reasons?
- Vary the sliding window score threshold, and the size of the window
  - How do these changes to the parameter affect the number or retained reads?

Exercise II. part 1: *de novo* assembly of RAD tags without a genome

1. In the first part of the second exercise we will be constructing stacks in two separate samples. These samples are from the same (anonymous) populations, however they

2. 10 minute mini-lecture on *Stacks*, primary/secondary reads, and parameters.

3. In your `~/working` workspace, create a directory called `ustacks` 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.

Copy data set 2a (DS2a):

```
        /data/denovo/lib01_samp01.fq.gz
```

and

```
        /data/denovo/lib02_samp01.fq.gz
```

into the `samples` directory and decompress them.

4. While these data are from the same biological population, they have very different characteristics when stacks are assembled from them. Your goal is to explore these two samples and understand how they are interacting with `ustacks`.

5. First, using UNIX commands, determine how many reads are present in each sample and the length of those reads.

6. Run `ustacks` on each sample, choosing the appropriate parameters for the length and number of reads. Be sure to capture the output from `ustacks` in an external file. Use the "`/usr/bin/time`" (with the `-v` option) program to monitor the run time and memory usage of each sample.

7. Examine the output from `ustacks`. What characteristics are different between the two samples? What could potentially be causing these differences?

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 2 (DS2):

   ```
   /data/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 to 60 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://<TGAC Instance Address>/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:

  `/data/denovo/mainparams` and

  `/data/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
```

12. Open the `batch_1.structure.console` and `batch_1.structure.out_f` files in the graphical interface for *Structure* on the virtual machine. 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?

Exercise III. Population genomics with a reference genome

1. Population genetics is a very old field that has a rich mathematical theory and a core set of statistical approaches for inferring parameters from genetic data. These statistics are such things as nucleotide diversity (π), differentiation statistics (i.e. $\mathbf{F_{st}}$), and measures of genetic covariance such as Linkage Disequilibrium (**D** and **D′**). However, because of methodological limitations, the majority of the theoretical, statistical and empirical work in population genetics has focused on a small number of loci. With the advent of second generation sequencing, tens or hundreds of thousands of genetic markers can now be examined in dozens of individuals, allowing the field of population genomics to truly come to fruition. An exciting new activity in population genomics is the identification of signatures of selection in wild populations. Today you will process RAD data from one oceanic and one freshwater population of threespine stickleback from Middleton Island, which is located off the coast of Alaska. One set of data comes from an ancestral oceanic population, whereas the other is from a derived freshwater population that is likely less than 60 years old. We will align these data to the stickleback reference genome using *GSnap*, and then feed the alignments into *Stacks*. After *Stacks* determines the loci and associated alleles present in each population, we will export the data and calculate several population genomic statistics, including F_{ST}. Performing a study like this was nearly impossible before the advent of next generation sequencing. *For more information on population genomics, see the papers listed in that section at the end of this document.*

2. 10 minute mini-lecture on diversity and divergence parameters, kernel smoothing, and signatures.

3. Acquire and process DS3 (Middleton Island).

   - In your `~/working` workspace, create a directory called `scan` to contain all the data for this exercise. Inside that directory, create three directories: `samples`, `aligned`, and `stacks`. To save time, we have already cleaned and demultiplexed this data set and will start from the cleaned samples stage.
   - Unarchive DS3 from

     ```
     /data/scan/middleton_scan.tar
     ```

     into the `samples` directory.

4. Align the stickleback sequences against the genome with *GSnap*.

   - Run *GSnap* on the first freshwater sample: `samples/s13_fw_01.fa.gz`

     [Running GSnap could take 15-30 minutes total.]

     - Running *GSnap* with the `--help` parameter will give you a list of all options.
     - We only want to keep alignments that have a single, best alignment to the genome. With the right combination of parameters, *GSnap* can do this natively.

       Some hints for command line parameters: set *GSnap* to read gzipped input files; allow a maximum of five mismatches in the alignment; set an indel penalty of two; set the terminal threshold to a high number; allow only one path (a.k.a.

alignment) for each read and if there is more than one path, suppress that read; make sure you turn on multithreading and output SAM format. Name the output file the same as the input file, with a ".sam" suffix instead of ".fa.gz".

- The stickleback *GSnap* database is located within this directory:

  ```
  /data/gsnap_db
  ```

  the GSnap database is stored in several files, we will specify only the common prefix of those files and GSnap will know how many files to read in.

- Use Samtools to convert the SAM output file to a BAM file. Delete the SAM file.
  - What is a terminal alignment and why is it important for RAD data?
  - Why do we convert the SAM file to a BAM file, what is the advantage?
  - Why was our `middleton_scan.tar` file not also gzipped?

- Run *GSnap* again with the first oceanic sample: `samples/s13_an_01.fa.gz` and convert it again to a BAM file using Samtools.
- To save time, the remaining 14 alignments can be found here:

  ```
  /data/scan/s13_gsnap.tar
  ```

  `Untar` these remaining *GSnap* alignments into the `aligned` directory.

5. Create a new MySQL database called `middleton_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. We need to create the database and then feed these commands to the MySQL server:

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

% mysql --defaults-file=/usr/local/share/stacks/sql/mysql.cnf
        middleton_radtags < /usr/local/share/stacks/sql/stacks.sql
```

6. We next want to run *Stacks* on the freshwater and anadromous population.

- Run the *Stacks* `ref_map.pl` pipeline program. This program will run `pstacks`, `cstacks`, and `sstacks` on the members of the population, accounting for the alignments of each read.
  - Information on `ref_map.pl` and its parameters can be found online:
    - http://catchenlab.life.illnois.edu/stacks/comp/ref_map.php
  - Create a file in the working directory called `popmap` that is formatted like this:

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

    Include all 16 samples in this file and specify which individuals belong to which populations and specify it to the `ref_map.pl` program.

- Specify each *GSnap*-aligned individual as a "sample" to `ref_map.pl`. Specify the `stacks` directory as the output location.

- Once *Stacks* has completed running, investigate the output files. Notice that each locus now has a chromosome/base pair specified in each of the `*tags.tsv` files and in the catalog files.
- Examine the Stacks output through the web interface:
    - http://<TGAC Instance Address>/stacks/

  Expand the details for one or more loci (click on a locus). Turn on the allele depths and click on some alleles to see the actual stack that corresponds.

7. The program `populations` calculates population genetic statistics for each SNP in the two populations for one level of population subdivision, as we have here. So, it will calculate expected and observed heterozygosity, π, F_{IS}, and it includes F_{ST} as a measure of genetic differentiation between populations. It uses the same method for calculating F_{ST} as was used in the human HapMap project.

- Now look at the output in the file `batch_1.sumstats.tsv`. There are a large number of statistics calculated at each SNP, such as the frequency of the major allele (*P*), and the observed and expected heterozygosity, and F_{IS}. Use UNIX commands like `head`, `cat`, `cut`, `more`, `column`, and `sort` to focus on the minimum and maximum heterozygosity and F_{IS} statistics. Are these statistics in the (roughly) same locations within the genome between the two populations? How are these summary statistics related to Hardy-Weinberg equilibrium?
- What are the population mean and standard error for π in the two populations (check the `batch_1.sumstats_summary.tsv` file).
- Examine catalog **locus 65** in the web interface (place the locus number in the catalog filter at the top of the web page). Use `grep` to extract **locus 65** from the `batch_1.sumstats.tsv` file. The sumstats file provides a measure of *P*, the most frequent allele present in the population. Verify that for the two SNPs in **locus 65** that *P* properly corresponds to the number of alleles present in the web interface.

8. Because RAD produces so many genetic markers, and because we have a reference genome sequence, we can examine population genetic statistics like F_{ST} as continuous distributions along the genome. The populations program does this using a kernel-smoothing sliding window approach.

  - Run the `populations` program again, this time turning on kernel smoothing for F_{ST}. Also, turn on filters so we only include loci available in both populations that are found in 75% of individuals of each population, and use a p-value correction to exclude insignificant F_{ST} measures. Be sure to again specify the population map you created previously and turn on multithreading.

- The output file `batch_1.fst_summary` contains the mean F_{ST} measure between populations. What is the mean F_{ST} between the marine and freshwater populations?

- The output file `batch_1.fst_1-2.tsv` contains F_{ST}, a measure of genetic differentiation between the two populations. What is the maximum value of F_{ST} at any SNP? How many SNPs reach this F_{ST} value?
- Look at the genomic distribution of F_{ST} in the file `batch_1.fst_1-2.tsv`. Use UNIX commands like `cut` and `sort` to find the genomic regions that show the highest levels of population differentiation.
  - What does the p-value generated by Fisher's exact test tell you about a particular F_{ST} score? How about the LOD score?
- Now plot F_{ST} over a single linkage group. First use `grep` to produce a new F_{ST} file with only data for Linkage Group IV (labeled **groupIV**), call it `batch_1.fst_1-2_lg4.tsv`. Now plot this file using gnuplot.
- Copy the Gnuplot script to the `stacks` directory:

```
/data/scan/fst_groupIV.gnuplot
```

- Cat the file to see what it does.
- Execute Gnuplot:

```
% gnuplot < fst_groupIV.gnuplot
```

- Download the resulting PDF file and open it. The red crosses represent the raw F_{ST} measures while the green line is the kernel-smoothed average value.

Citations and Readings

Core readings in preparation for the lecture and workshop

Amores, A., et al. 2011. Genome evolution and meiotic maps by massively parallel DNA sequencing. **Genetics** 188:799-808.

Arnold, B. et al. 2013. RADseq underestimates diversity and introduces genealogical biases due to nonrandom haplotype sampling. **Molecular Ecology** 22: 3179–3190.

Catchen, J. et al. 2011. *Stacks*: building and genotyping loci de novo from short-read sequences. **G3: Genes, Genomes and Genetics** 1:171-182.

Catchen, J. et al. 2013a. The population structure and recent colonization history of Oregon threespine stickleback determined using restriction-site associated DNA-sequencing. **Molecular Ecology**, 22:2864–2883.

Catchen, J. et al. 2013b. Stacks: an analysis tool set for population genomics. **Molecular Ecology** 22:3124–3140.

Davey, J. W., et al. 2011. Genome-wide genetic marker discovery and genotyping using next-generation sequencing. **Nature Reviews Genetics** 12:499-510.

Davey, J. W. et al. Special features of RAD Sequencing data: implications for genotyping. **Molecular Ecology** 22: 3151–3164.

Etter, P. D., et al. 2011. SNP Discovery and Genotyping for Evolutionary Genetics using RAD sequencing. *in* Molecular Methods in Evolutionary Genetics, Rockman, M., and Orgonogozo, V., eds.

Ekblom, R., and J. Galindo. 2010. Applications of next generation sequencing in molecular ecology of non-model organisms. **Heredity** 107:1-15.

Hohenlohe, P. A. et al. 2010. Population genomics of parallel adaptation in threespine stickleback using sequenced RAD tags. **PLoS Genetics** 6: 1-23.

Population genomics background, concepts and statistical considerations

Cariou, M. et al. 2013. Is RAD-seq suitable for phylogenetic inference? An *in silico* assessment and optimization. **Ecol Evol** 3, 846–852.

Gompert, Z., and C. A. Buerkle. 2011a. A hierarchical Bayesian model for next-generation population genomics. **Genetics** 187:903-917.

Hohenlohe, P. A., et al. 2010. Using population genomics to detect selection in natural populations: Key concepts and methodological considerations. **International Journal of Plant Sciences** 171:1059-1071.

Luikart, G.,et al. 2003. The power and promise of population genomics: from genotyping to genome typing. **Nature Reviews Genetics** 4:981-994.

Lynch, M. 2009. Estimation of allele frequencies from high-coverage genome-sequencing projects. **Genetics** 182:295-301.

Nielsen, R., et al. 2005. Genomic scans for selective sweeps using SNP data. **Genome Research** 15:1566-1575.

Nielsen, R., et al. 2011. Genotype and SNP calling from next-generation sequencing data. **Nature Reviews Genetics** 12:443-451.

Rubin, B. E. R. et al. 2012. Inferring Phylogenies from RAD Sequence Data. **PLoS ONE** 7, e33394–e33394.

Stapley, J., et al. 2010. Adaptation genomics: the next generation. **Trends in Ecology and Evolution** 25:705-712.

Empirical studies using RRL and RAD sequencing

Altshuler, D., et al. 2000. An SNP map of the human genome generated by reduced representation shotgun sequencing. **Nature** 407:513-516.

Baxter, S. W., et al. 2011. Linkage mapping and comparative genomics using next-generation RAD sequencing of a non-model organism. **PLoS ONE** 6:e19315.

Chutimanitsakun, Y., et al. 2011. Construction and application for QTL analysis of a Restriction Site Associated DNA (RAD) linkage map in barley. **BMC Genomics** 12: 1-13.

Emerson, K. J., et al. 2010. Resolving postglacial phylogeography using high-throughput sequencing. **Proceedings of the National Academy of Sciences** 107:16196-16200.

Gore, M. A., et al. 2009. A first-generation haplotype map of maize. **Science** 326:1115-1117.

Richards, P. M. et al. 2013. RAD-Seq derived markers flank the shell colour and banding loci of the *Cepaea nemoralis* supergene. **Molecular Ecology** 22, 3077–3089.

RAD-seq genotyping methodology

Baird, N. A., et al. 2008. Rapid SNP discovery and genetic mapping using sequenced RAD markers. **PLoS ONE** 3:e3376.

Etter, P. D., et al. 2011. Local De Novo Assembly of RAD Paired-End Contigs Using Short Sequencing Reads. **PLoS ONE** 6:e18561

Hohenlohe, P. A., et al. 2011. Next-generation RAD sequencing identifies thousands of SNPs for assessing hybridization between rainbow and westslope cutthroat trout. **Molecular Ecology Resources** 11 Suppl 1:117-122.

Miller, M. R., et al. 2007. Rapid and cost-effective polymorphism identification and genotyping using restriction site associated DNA (RAD) markers. **Genome Research** 17:240-248.

Willing, E. M., et al. 2011. Paired-end RAD-seq for de novo assembly and marker design without available reference. **Bioinformatics** 27:2187-2193.

Related reduced representation library (RRL) methodologies

Andolfatto, P., et al. 2011. Multiplexed shotgun genotyping for rapid and efficient genetic mapping. **Genome Research** 21:610-617.

Elshire, R. J., et al. 2011. A Robust, Simple Genotyping-by-Sequencing (GBS) Approach for High Diversity Species. **PLoS ONE** 6:e19379.

Peterson, B. K. et al. 2012. Double digest RADseq: an inexpensive method for de novo SNP discovery and genotyping in model and non-model species. **PLoS ONE** 7, e37135.

Rigola, D., et al. 2009. High-Throughput Detection of Induced Mutations and Natural Variation Using KeyPoint™ Technology. **PLoS ONE** 4:e4761.

van Orsouw, N. J., et al. 2007. Complexity reduction of polymorphic sequences (CRoPS): a novel approach for large-scale polymorphism discovery in complex genomes. **PLoS ONE** 2:e1172.

van Tassell, C. P., et al. 2008. SNP discovery and allele frequency estimation by deep sequencing of reduced representation libraries. **Nature Methods** 5:247-252

Wang, S. et al. 2012. 2b-RAD: a simple and flexible method for genome-wide genotyping. **Nature Methods** 9, 808–810.

Useful links

1. Quality scores
    1. http://en.wikipedia.org/wiki/FASTQ_format
    2. http://en.wikipedia.org/wiki/Phred_quality_score
    3. http://www.phrap.com/phred/
    4. http://www.illumina.com/truseq/quality_101/quality_scores.ilmn

2. Basic Unix and PERL commands
    1. http://mally.stanford.edu/~sr/computing/basic-unix.html
    2. http://korflab.ucdavis.edu/Unix_and_Perl/

3. *Stacks* download and tutorials
    1. http://catchenlab.life.illnois.edu/stacks/

4. Great site for information on next gen sequencing
    1. http://seqanswers.com/