## Analysis pipeline for 2b-RAD data

For this morning's exercises, there is a sample data set stored on the server. This dataset consists of 10 fastq files that have been truncated to 1 000 000 reads (50 base-pair, single-end reads from an Ilumina HiSeq machine). They are from a marine/brackish water isopod called "*Idotea balthica*" collected last year in Falsterbo, southwestern Sweden. The sequencing libraries were prepared with a 2b-RAD protocol and were sequenced single-end with 50 bp reads. However, the fragments are shorter than that, only 34 bases, so the last 16 bases of the reads will consist of the Illumina adapter attached to the restriction fragment. So in these bases we expect no variation at all.

In addition to this, the restriction enzyme that was used cuts out a fragment that looks like this:

(10 variable bases)CGA(6 variable bases)TGC(12 variable bases)

and also the reverse complement, which is:

(10 variable bases)GCA(6 variable bases)TCG(12 variable bases)

Thus, we also expect a reduction in variability and non-random base distributions in bases 11-13 and 20-22 of the reads, as those regions can be one of the two restriction sites only (Figure 1).

Depending on which sequencing platform one uses, this pattern can cause problems, as for example Illumina machines depend on variation in the color pattern created by the fluorescent dyes in order to distinguish between bases, and if not taken into account, one might see a sharp drop in sequence quality in the regions of low complexity (see Figure 2 below). One way to deal with this is to add a small fraction of a high complexity library of known sequence, which can then be bioinformatically remove. Illumina supplies one such called "PhiX".

"fastq" text files, are organized so that each short read takes up four lines. The first line (starting with an @) is a read identifier, the second is the DNA sequence, the third another identifier (same as line 1, but starting with a +(or sometimes only consisting of a +)) and the fourth is a Phred quality score symbol for each base in the read. The quality score is based on the ASCII character code used by computer keyboards (http://www.ascii-code.com/). Illumina's current sequencing pipeline uses an offset of 33, so that an ! (ASCII code 33) is 0, and I (ASCII code 73) is 40. The quality score for each base ranges from -5 to 40 and is defined as $Q_{phred}$ =-10 log10(p), where p is the estimated probability of a base call being wrong. So a $Q_{phred}$ of 20 corresponds to a 99 % probability of a correctly identified base.

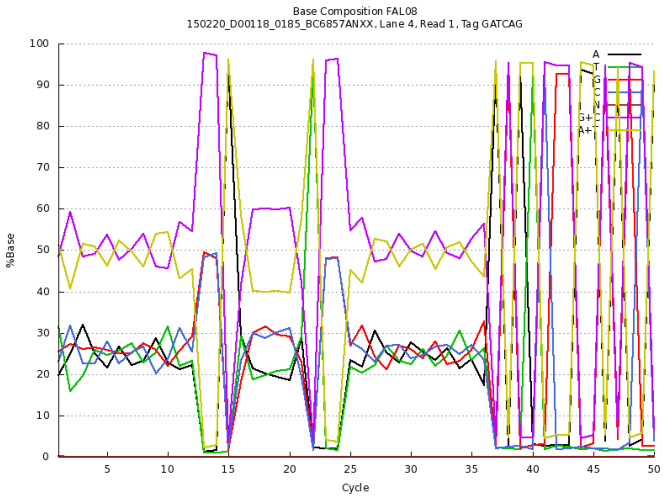


Figure 1. Percent base distributions along a 50 bp read from a 2b-RAD library.

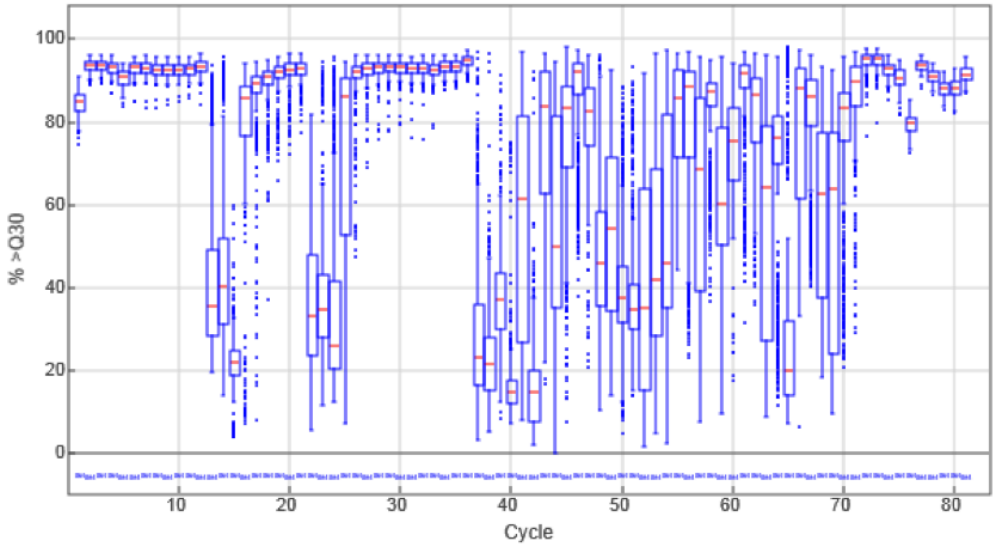


Figure 2. Percent bases with Q>30 along a 75 bp read from a 2b-RAD fragment not spiked with PhiX. Notice the drop in quality around the restriction sites and the adapter.

In this exercise, we will use bash scripts to process multiple files at once; before executing them we will need to open them and make sure that input and output filenames as well as adapter sequences are correct. The objectives are to A. Quality control and prepare the input data, B. Cluster the fragments into loci and genotype our individuals, C. Filter the genotype data to keep only high quality loci and genotypes, and D. Output the genotype data along with statistics.

## A. QUALITY CONTROL AND DATA PREPARATION

1) Login to Albiorix using *ssh*, then login to the high_mem computation node using *qlogin* and move to your working folder called "data". This is where you will be working for this entire exercise.

2) Cutting out our restriction fragments from the reads.

a. We will start by creating a bash script called `trims.sh`, using a perl script called `2bRAD_trim_launch.pl`.

```
../scripts/2bRAD_trim_launch.pl fastq sampleID=1 >
../scripts/trims.sh
```

b. Now, open the bash script in the text editor "nano" with

```
nano ../scripts/trims.sh
```

To close the text editor, just hit Control-X

As you can see, the bash script is just a list of commands that we could also type in one by one into the terminal. In our case it iterates the same command over all fastq files contained in the directory.

c. In order to make the bash script executable, we need to let the computer know that this is a bash script. We do this by writing: `# !/bin/bash` on the first line of the file, then closing nano and re-saving the file.

d. Execute the bash script by typing:

```
sh ../scripts/trims.sh
```

while in the folder containing your data. This will now cut out the fragments and save them in files with the extensions ".tr0"

Note: You might need to change file permissions to make the bash script executable. You do this with `chmod 755 ../scripts/trims.sh`

3) Filter out low quality read ends using `fastq_quality_filter` (part of the fastx toolkit)

a. We will start by creating a bash script called `filt.sh`, using a bash script called `2bRAD_filt_launch.sh`.

```
../scripts/2bRAD_filt_launch.sh
```

b. Now, open the bash script in the text editor "nano" with

To close the text editor, just hit Control-X

```
nano ../scripts/filt.sh
```

Same as in the previous step, the bash script is a list of commands. This one excutes the `fastq_quality_filter` program over all `.tr0` files contained in the directory.

c. As before, we need to let the computer know that this is a bash script, by writing:
`# !/bin/bash` on the first line of the file, then closing nano and re-saving the file.

d. Execute the bash script by typing:

```
sh ../scripts/filt.sh
```

while in the folder containing your data. This will now trim fragments and save them in files with the extensions ".trim"

4) Now that we have removed poor quality data, we should remove redundant data before the clustering step, otherwise it will take too long. We do this by removing duplicates and only keeping unique reads using the perl script called "`uniquerOne.pl`"

a. We will start by creating a bash script called `unii.sh`, using a bash script called `2bRAD_uni_launch.sh`.

```
../scripts/2bRAD_uni_launch.sh
```

b. Now, open the bash script in the text editor "nano" with

```
nano ../scripts/unii.sh
```

```
To close the text editor, just hit Control-X
```

Same as in the previous step, the bash script is a list of commands. This one excutes the uniquerOne.pl script over all `.trim` files contained in the directory.

c. As before, we need to let the computer know that this is a bash script, by writing:
`# !/bin/bash` on the first line of the file, then closing nano and re-saving the file.

d. Execute the bash script by typing:

```
sh ../scripts/unii.sh
```

while in the folder containing your data. This will now unique your files (remove duplicates), and save the new files with the extension ".trim.uni".

e. Finally, you can merge all of your uniqued files into one that we will use for clustering. This can be accomplished with the script `mergeUniq.pl`, by typing:

```
../scripts/mergeUniq.pl uni >Merged.uniq
```

This will create both a file called Merged.uniq, where count information is stored about how many reads each unique sequence represents for each individual (we'll use this one for genotyping later), and also one file called "mergedUniqTags.fasta", which only contains the sequences (this is the file that we will use for clustering).

## B. CLUSTERING AND GENOTYPING

1) For clustering, we will you the software "cd-hit-est", which is quick and optimized for short reads. We are using "mergedUniqTags.fasta" as input file and the output file will be called "cdh_alltags.fas".
There are many different parameters that can be changed for this software (for a full list, you can type: `../scripts/cd-hit-est –help`). For our purposes, the most important one is –c, which is the identity threshold. We set it to 91 %, which translates to about 3 nucleotide substitutions allowed.

   a. We run the program by typing (while in the folder containing our data):

```
../scripts/cd-hit-est -i mergedUniqTags.fasta -o
cdh_alltags.fas -aL 1 -aS 1 -g 1 -c 0.91 -M 0 -T 0
```

   b. Now, we want to merge our clusters with the counts data for each individual (stored in Merged.uniq), in order to be able to genotype. For this, we use a perl script called `uniq2loci.pl`. The output file is called "`alltags.ul`".

```
../scripts/uniq2loci.pl Merged.uniq
cdh_alltags.fas.clstr cdh_alltags.fas >alltags.ul
```

   c. We can examine what `alltags.ul` looks like using head: `head alltags.ul.`
   As you can see, this file contains information about loci, tags (alleles), the sequences, and counts for all individuals.

d. Finally, we want to extract genotype data from the alltags.ul file. We use the haplocall_denovo.pl script for this.
Now we are at a step where we really need to consider biology. We need to specify: Minimum depth for calling a homozygote (5), how many individuals should have the allele to call it (2), and what fraction heterozygotes should we allow (0.8)?

```
../scripts/haplocall_denovo.pl alltags.ul mindp=5
ind=2 hetero=0.8
```

e. This will output information about the number of loci and alleles on the screen, and will save genotype data both for SNPs and haplotypes in two separate files in the .vcf file format (variant call format). We will talk more about this format later, but you can find information about it here: http://samtools.github.io/hts-specs/VCFv4.2.pdf

**C. FILTERING**

1) From now on, we will work with the SNP data, but the instructions below would work for the haplotype data as well. We will start by using out technical replicate to see if there are any potential contaminants or PCR errors, and then use this information in order to optimize our dataset using a machine-learning algorithm. As we will see, this only works if you have a large amount of data and several technical replicates - as we are working with truncated data files, our filtering steps will not be efficient.

Each of the following filtering steps creates a vcf output file that is a modified and renamed version of the input file. In a real case you would want to rename them to something that makes sense to you, but in this case, we will just use the names in the exercise. The final file after this step will be called `SNPs_FINAL.recode.vcf`, which is the file we will then extract our genotype data from in the next step.

a. First, we need to tell the software which of our samples are the replicates. To do this, we create a file called "clonepairs.tab". It is just a tab-delimited textfile which lists out replicates, one on each row. In our case we only have one replicate pair, which is FAL25A and FAL25B. So, we can type in: `nano clonepairs.tab`
And then write: FAL25A[TAB]FAL25B
Then we close with CTRL-X and save.

b. We use the script called "replicatesMatch.pl" to encorporate the replicate information into a new vcf file called "`vqsr.denovo.vcf`"

```
../scripts/replicatesMatch.pl
vcf=alltags.ul_Variants_count5_ab10_sb10_clip0.vcf
replicates=clonepairs.tab polyonly=1 > vqsr.denovo.vcf
```

c. Recalibrate the SNP quality scores based on the above information, using recalibrateSNPs.pl. This scripts also outputs a table on the screen that indicates which quality cutoff will produce the best gain in amount of output data. We will use this setting in the next step (this will produce nonsense in our case as we do not have enough data, so we will have to use a cutoff of 1 in the next step).

```
../scripts/recalibrateSNPs.pl
vcf=alltags.ul_Variants_count5_ab10_sb10_clip0.vcf
true=vqsr.denovo.vcf -nosb -notp >denovo.recal.vcf
```

d. Now, filter the vcf file using the quality cutoff with the maximum step from the previous step (unfortunately 1, as we lack enough data to produce an accurate model). This step we conduct with the software package called "vcftools".

```
../scripts/vcftools --vcf denovo.recal.vcf --minQ 1 --
recode --out denovo.filt
```

e. Evaluating the accuracy of the replicates

```
../scripts/repMatchStats.pl vcf=denovo.filt.recode.vcf
replicates=clonepairs.tab
```

f. Checking samples for heterozygosity values

```
../scripts/vcftools --vcf denovo.filt.recode.vcf --het
--out denovo.out
```

g. Then open the file called denovo.out.het with nano and make note of the observed vs expected heterozygosity. Does anything look strange?

h. Finally, we want to use vcftools to filter out low-quality SNPs (Q<20) and genotypes (Q<15), individuals with more than 2 alleles and loci with a lot of missing data (present in 80 % of samples). This will produce our final filtered call set, in a file called SNPs_FINAL.recode.vcf.

```
../scripts/vcftools --vcf denovo.filt.recode.vcf --
minQ 20 --minGQ 15 --min-alleles 2 --max-alleles 2 --
max-missing 0.8 --recode --out SNPs_FINAL
```

## D. OUTPUT AND STATISTICS

1) We will now use vcftools to extract statistics and genotype data from out SNPs_FINAL.recode.vcf file. Vcftools has a plethora of different options for extracting data from vcf files. For a detailed list of all the options, the manual can be found here: https://vcftools.github.io/man_latest.html#OUTPUT%20OPTIONS

   Today, we will focus on extraction of genotype data as a 012-matrix, and to extract statistics about Hardy-Weinberg equilibrium. The command for these two options is the same, except for the last flag.

   a. Extract a genotype matrix, by typing

```
../scripts/vcftools --vcf SNPs_FINAL.recode.vcf --out
SNPs_FINAL --012
```

      This will produce an output file called `SNPs_FINAL.012`, which you can copy to your local computer and open in for example Excel. It also outputs files called `SNPs_FINAL.012.indv`, where information about the sample names is located and `SNPs_FINAL.012.pos`, where information about the loci is stored.

   b. Open the .012 file in Excel and examine it. -1 codes for missing data, 0 and 2 are homozygotes and 1 is a heterozygote. What's the fraction of missing data (remember, we got rid of loci present in less than 80 % of samples)? Do you see any immediate pattern?

   c. Extract Hardy-Weinberg statistics, by typing:

```
../scripts/vcftools --vcf SNPs_FINAL.recode.vcf --out
SNPs_FINAL --hardy
```

      This will produce a text file called SNPs_FINAL.hwe, where the Hardy-Weinberg statistics for each locus are stored.

   d. Copy the .hwe file to your local computer and open it in Excel. The two last columns are the p-values for heterozygote deficiency and heterozygote excess, respectively. Calculate what fraction of loci exhibit heterozygote deficiency at a p-value of 0.05. We expect to see about 2.5 % by chance alone, what do we see?