Guidelines for filtering and analyzing RAD-seq / GBS data

AUTEUR:

Céline Reisser

UMR 5175, CEFE CNRS
Novembre 2015

Contents

List of Figures

1 Quality control of raw library, demultiplexing and filtering raw reads

1.1 Integrity of the downloaded files

The first step to carry after downloading or obtaining your sequencing results is to control the integrity of the files. This can be done by controlling the MD5 label of the file. Files that contain the .md5 file extension are checksum files that are used to ensure that the data within a file is complete and has not become corrupt.To do so, the sequencing company should have generated a report of each file, containing the MD5 code of the file. In order to check for integrity after downloading the file on your platform, type md5sum and the name of the file you want to test. Then compare the code it returns with the code given to you by the sequencing company.

```
md5sum downloaded_file.fastq.gz
```

1.2 Quality control of the libraries Fastq files

The quality of the libraries and of the raw reads have to be checked, in order to detect any bias or sequencing problems that could have occurred.

FastQC aims to provide a simple way to do some quality control checks on raw sequence data coming from high throughput sequencing pipelines. It provides a modular set of analyses which you can use to give a quick impression of whether your data has any problems of which you should be aware before doing any further analysis.
The main functions of FastQC are:

- Import of data from BAM, SAM or FastQ files (any variant)
- Providing a quick overview to tell you in which areas there may be problems
- Summary graphs and tables to quickly assess your data
- Export of results to an HTML based permanent report
- Offline operation to allow automated generation of reports without running the interactive application.

The program can be downloaded at:
http://www.bioinformatics.babraham.ac.uk/projects/fastqc/

Similarly, under linux, the program can be installed by typing:

```
sudo ap-get install fastqc
```

Select the files (fastq or fastq.gz) and generate a report. The quality of the nucleotide call should be of high quality throughout the length of the read (Fig.1). It is however possible that the last 5 to 10 nucleotides show a lower level of quality (or a higher variability in quality). It is possible then to trim the end of the reads to remove the last 5 SNPs and avoid calling false SNPs. Overall quality of the reads should be consisting of a single peak, with ideally a quality score of 30 or more (Fig.1).

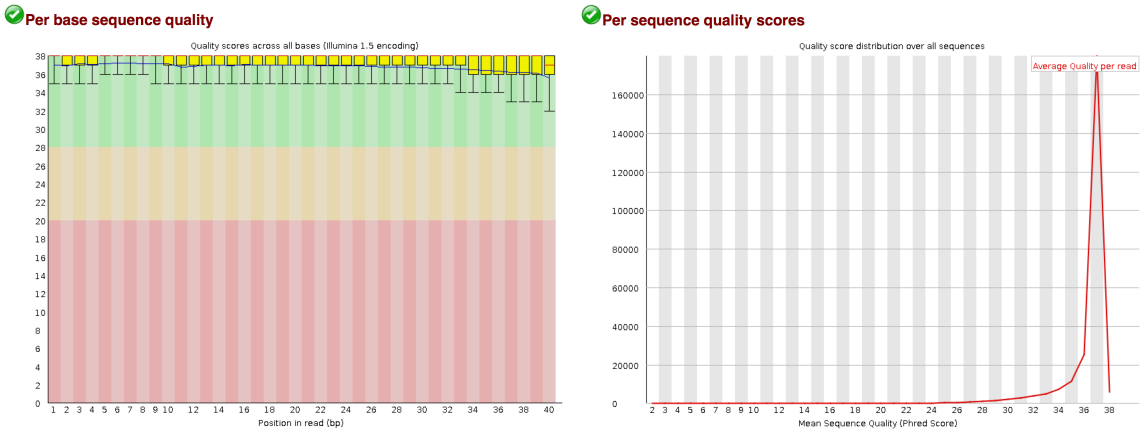


Figure 1: Mean per-nucleotide quality accross the length of the reads (left) and overall read quality (right) of the library

The nucleotidic composition per position is expected to show over-represented nucleotides in the first 6 to 7 bases. This is normal, as they consist in the enzymatic cut sites of the reads, and hence do not represent a random selection of A T C and G across the genome (Fig.1.2). Also, the GC content should fit the expectations nicely, since the technique is not supposed to lead to a representation bias (Fig.1.2).

An extra step is to control for the presence of remnant of Illumina adaptors in your library (in the adapter tabulation of the FastQC report). Usually, Illumina pipeline will remove those prior to making the results available to you. If you find a significant contamination of your library with adapters, refer to the next section for efficient filtering of your reads.

1.3 Filtering the raw reads

If the quality controls of your library do not reveal a problem, you can go ahead and skip this section, since further filtering will be available with the Stacks pipeline when demultiplexing the reads. However, if a serious quality problem was identified (such as contamination by adapter sequences, or a very low quality at the end of the reads), the library will need to be filtered accordingly. Trimmomatic can be used for trimming the length of the reads, removing bad quality reads and removing potential adapter contamination.

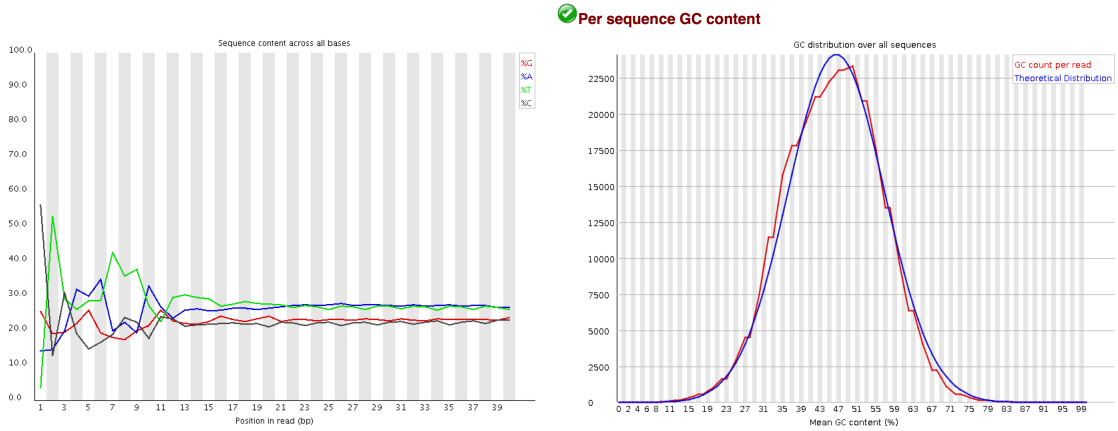


Figure 2: Nucleotide composition at each site along the reads (left) as well as the GC content of the library (right)

For Paired End data:

```
java -jar trimmomatic-0.30.jar PE -phred33 input_forward.fq.gz
    input_reverse.fq.gz output_forward_paired.fq.gz
    output_forward_unpaired.fq.gz output_reverse_paired.fq.gz
    output_reverse_unpaired.fq.gz ILLUMINACLIP:TruSeq3-PE.fa
    :2:30:10 LEADING:3 TRAILING:3 SLIDINGWINDOW:4:15
```

This will:

- Remove adapters (ILLUMINACLIP:TruSeq3-PE.fa:2:30:10)
- Remove leading low quality or N bases (below quality 3) (LEADING:3)
- Remove trailing low quality or N bases (below quality 3) (TRAILING:3)
- Scan the read with a 4-base wide sliding window, cutting when the average quality per base drops below 15 (SLIDINGWINDOW:4:15)

For Single End:

```
java -jar trimmomatic-0.30.jar SE -phred33 input.fq.gz output.
    fq.gz ILLUMINACLIP:TruSeq3-SE:2:30:10 LEADING:3 TRAILING:3
    SLIDINGWINDOW:4:15 MINLEN:36
```

This will perform the same steps, using the single-ended adapter file.

2 Demultiplexing the fastq files

Now that the files containing raw sequences have passed the quality filters, the analytic pipeline can begin. For this, we will use Stacks (Catchen et al. 2013). The first step in the pipeline is to demultiplex the reads (i.e. regroup reads belonging to one individual within a single fastq file). This step is performed using the first program implemented in the Stacks analysis pipeline. The program used for demultiplexing is called process_radtags.

This program examines raw reads from an Illumina sequencing run and first, checks that the barcode and the RAD cutsite are intact, and demultiplexes the data. If there are errors in the barcode or the RAD site within a certain allowance process_radtags can correct them. Second, it slides a window down the length of the read and checks the average quality score within the window. If the score drops below 90% probability of being correct (a raw phred score of 10), the read is discarded. This allows for some sequencing errors while eliminating reads where the sequence is degrading as it is being sequenced. By default the sliding window is 15% of the length of the read, but can be specified on the command line (the threshold and window size can be adjusted).

The usual command line used for good looking library is as follow:

```
process_radtags -p Path/to/the/folder/with/fastq.gz -i gzfastq
    -o Path/to/output/folder/ -b Path/to/your/barcode/file.txt
    -e PstI -r -c -q --filter_illumina -w 0.2
```

The barcode file is a standard text file (saved in linux format, for correct line ending characters) that should look like this:

ATTCG
GGTCA
GAGAA
GTTGA

The program's options (Fig.3) allow you to set the quality thresholds to be used for filtering the reads. It also allows for automated filtering based on illumina quality scores, and rescue radtags for which one nucleotide is missing in the barcode. The barcodes have been designed to contain at least 2 nucleotides difference, to be able to attribute them to an individual even in case of one nucleotide miscall.

Once the pipeline is finished, fastq files are generated in the output folder you specified, and a log file reports the results of the filtering. This log file is important as it will be the first data you will obtain on the real quality of your library (missing individuals, excessive missing cut-sites (meaning potential contamination of your library). When opening the log file (Fig.4), you will see the first line resuming the parameters you have used in process_radtags. This line is followed by statistics on how the raw fastq.gz files were treated.

Program Options

```
process_radtags [-f in_file | -p in_dir [-P] [-I] | -1 pair_1 -2 pair_2] -b barcode_file -o out_dir -e enz

                [-c] [-q] [-r] [-t len] [-D] [-w size] [-s lim] [-h]

   f  - path to the input file if processing single-end seqeunces.
   i  - input file type, either 'bustard' for the Illumina BUSTARD format, 'bam', 'fastq' (default), or 'gzfastq' for gzipped FASTQ.
   y  - output type, either 'fastq', 'gzfastq', 'fasta', or 'gzfasta' (default is to match the input file type).
   p  - path to a directory of files.
   P  - files contained within directory specified by '-p' are paired.
   I  - specify that the paired-end reads are interleaved in single files.
   1  - first input file in a set of paired-end sequences.
   2  - second input file in a set of paired-end sequences.
   o  - path to output the processed files.
   b  - path to a file containing barcodes for this run.
   c  - clean data, remove any read with an uncalled base.
   q  - discard reads with low quality scores.
   r  - rescue barcodes and RAD-Tags.
   t  - truncate final read length to this value.
   E  - specify how quality scores are encoded, 'phred33' (Illumina 1.8+, Sanger, default) or 'phred64' (Illumina 1.3 - 1.5).
   D  - capture discarded reads to a file.
   w  - set the size of the sliding window as a fraction of the read length, between 0 and 1 (default 0.15).
   s  - set the score limit. If the average score within the sliding window drops below this value, the read is discarded (default 10).
   h  - display this help messsage.

 Barcode options:
   --inline_null: barcode is inline with sequence, occurs only on single-end read (default).
   --index_null: barcode is provded in FASTQ header, occurs only on single-end read.
   --inline_inline: barcode is inline with sequence, occurs on single and paired-end read.
   --index_index: barcode is provded in FASTQ header, occurs on single and paired-end read.
   --inline_index: barcode is inline with sequence on single-end read, occurs in FASTQ header for paired-end read.
   --index_inline: barcode occurs in FASTQ header for single-end read, is inline with sequence on paired-end read.

 Restriction enzyme options:
   -e [enz], --renz_1 [enz]: provide the restriction enzyme used (cut site occurs on single-end read)
   --renz_2 [enz]: if a double digest was used, provide the second restriction enzyme used (cut site occurs on the paired-end read).
      Currently supported enzymes include:
        'aluI', 'apeKI', 'apoI', 'bamHI', 'bglII', 'bstYI', 'claI', 'ddeI', 'dpnII', 'eaeI', 'ecoRI', 'ecoRV', 'ecoT22I', 'hindIII', 'kpnI',
        'mluCI', 'mseI', 'mspI', 'ndeI', 'nheI', 'nlaIII', 'notI', 'nsiI', 'pstI', 'rsaI', 'sacI', 'sau3AI', 'sbfI', 'sexAI', 'sgrAI',
        'speI', 'sphI', 'taqI', 'xbaI', or 'xhoI'.

 Adapter options:
   --adapter_1 [sequence]: provide adaptor sequence that may occur on the single-end read for filtering.
   --adapter_2 [sequence]: provide adaptor sequence that may occur on the paired-read for filtering.
   --adapter_mm [mismatches]: number of mismatches allowed in the adapter sequence.

 Output options:
   --merge: if no barcodes are specified, merge all input files into a single output file.

 Advanced options:
   --filter_illumina: discard reads that have been marked by Illumina's chastity/purity filter as failing.
   --disable_rad_check: disable checking if the RAD site is intact.
   --barcode_dist: provide the distace between barcodes to allow for barcode rescue (default 2).
```

Figure 3: List of the process_radtags program's options

Specifically, the table lists how many reads:

- were retained and assigned to an individual
- were filtered out due to low quality or to Illumina quality guideline
- had an ambiguous barcode (no individual assignment possible)
- had an ambiguous RAD-Tag (the enzymatic cutsite does not match with the enzyme you specified in the command line)
- were looked at in total.

Directly following are the statistics for all the fastq.gz files you have been inspecting in this round of process_radtags. Then, a list containing the barcodes you have listed in the barcode.txt file is given along with the number of reads that were assigned to

```
process_radtags -p /media/cmor/CELINE_2TB/RADSeq2013/raw_files_RADSeq/Lib_Rm_1/
-i gzfastq
-o /media/cmor/SSD_2/Lib-rm-1_cleaned/
-b /media/cmor/SSD_2/Lib-rm-1_cleaned/Rm1_barcodes.txt
-e pstI
-r -c -q -t 90
--filter_illumina
-w 0.2
-s 15

process_radtags executed 2014-04-04 18:13:49

File                                    Retained Reads  Illumina Filtered       Low Quality     Ambiguous Barcodes      Ambiguous RAD-Tag       Total
BSSE_NoIndex_L002_R1_001.fastq.gz       37861935        9858875                 1359690         425438                  494062                  50000000
BSSE_NoIndex_L002_R1_002.fastq.gz       36459665        11223444                1223401         573009                  520481                  50000000
BSSE_NoIndex_L002_R1_003.fastq.gz       37854837        9829847                 1364785         441774                  508757                  50000000
BSSE_NoIndex_L002_R1_004.fastq.gz       35973801        11533423                1265166         729528                  498082                  50000000
BSSE_NoIndex_L002_R1_005.fastq.gz       762617          899318                  29761           131393                  12623                   1835712

Total Sequences 201835712
Failed Illumina filtered reads  43344907
Ambiguous Barcodes      2301142
Low Quality     5242803
Ambiguous RAD-Tag       2034005
Retained Reads  148912855

Barcode Total           No RadTag       Low Quality     Retained
GGGGA   8318009         60069           220660          6486110
CTAGG   12478440        113056          319799  9028580
GTACA   8462607         51481           211724  6270323
CTGAA   9432958         95177           261340  7203269
AGAGT   6057664         76511           164783  4309505
TCAGA   6440111         46381           163296  4727891
GAAGC   7880982         46295           193221  5849451
CGATA   7229083         33468           201602  5268582
GTGTG   9063801         73226           243603  7228283
GAGAT   5692310         155869          139443  4354673
TCGAG   8200071         86664           225751  6296628
AGCTG   5210306         51243           132650  3834217
TGACC   6567562         52287           165824  4794101
TGGTT   6388721         63703           166511  5026290
AGGAC   4405532         42078           107295  3460967
AAAAA   10411238        126569          272711  7409269
TAATG   10292383        166728          274582  7365432
AACCC   6238687         48907           159640  4541206
AAGGG   5579372         65988           149377  4301000
CGGCG   4283615         16180           147026  3199570
TAGCA   6230417         90271           164940  4775644
AATTT   11009065        112629          298534  8390933
GCATT   3935596         38790           101058  2883054
ACACG   5184617         32418           138864  3718044
GCGCC   5142454         23615           136247  4010487
ACCAT   6131785         159972          151206  4336156
ACGTA   4663353         24071           113645  3496450
GGAAG   6203910         55221           156836  4503844
ACTGC   2399921         25138           60635   1842896
```

Figure 4: Presentation of the process_radtag log file

a specific barcode. The fastq files obtained through this analysis are the one that will be used in downstream analyses.

3 Mapping the reads to a reference genome, and post-alignment filtering

This section is for analyses for which a reference genome is available. If you do not have a reference genome and want to do the analysis De-Novo, refer to the next section.

3.1 Aligning the reads to a reference genome

There are many options and programs to choose from when mapping reads to a reference genome. The choice resides often in the downstream softwares you will be using. Bowtie2 works well with Stacks, however if you want to use GATK, the output files of Bowtie2 will not work properly. It is mostly due to a particular expectation of GATK about the indexing and organisation of the output files from an aligner.

An aligner that works well with both Stacks and GATK is BWA (Li & Durbin, 2009), using the Burrow-Wheeler Algorithm. BWA is a software package that has been ameliorated in order to deal well with short reads from NGS technologies.

The first thing to do before proceeding to mapping is to index your genome. For this, bwa have implemented an internal command called "bwa index", that takes a reference genome in FASTA format and redirect the output with a prefix of your choice. For example, the following command will index a genome using the "IS" algorithm (developped for efficiency for genomes less than 2Gb, it is the default option). If your reference genome is larger than 2Gb, then use the algorithm "bwtsw". BWA will index the genome and write the indexed file in your working directory. Make sure to be in the right place before executing the command.

```
bwa index reference_daphnia_v2-4.fasta
```

Once your genome has been successfully indexed, you are ready to start mapping your reads. As for the indexing, BWA packs up different algorithms to perform read mapping, according to the specificity of your reads and to the quality of the data. BWA-MEM and BWA-ALN share similar features such as long-read support and split alignment, but BWA-MEM, which is the latest, is generally recommended for high-quality queries (such as RAD-seq and GBS) as it is faster and more accurate. BWA-MEM also has better performance than BWA-ALN for 70-100bp Illumina reads.

Here is an example of two command lines for BWA default set-up mapping, for Single-End reads and for Paired-end reads:

```
bwa mem ref.fa reads.fq > aln-se.sam
bwa mem ref.fa read1.fq read2.fq > aln-pe.sam
```

Once the algorithm finishes, you will be left with a SAM formatted file. SAM format (and its binary version the BAM format) are the usual standard format for alignment results. A SAM formatted file reports data on the quality and type of alignment for each read of your fastq files (Fig.5). Each alignment line has 11 mandatory fields for essential alignment information such as mapping position, and variable number of optional fields for flexible or aligner specific information.

```
@SQ SN:dmagna-v-2-4 LN:15822
@PG ID:bwa  PN:bwa  VN:0.7.12-r1039 CL:./bwa mem -t 20 -T 10 -c 1 ./dmagna-v-2-4.fa /home/creisser/demultiplexed_tot/mos-01-04_AAGGG.fq

1_1208_19788_12378_1    0   scaffold00011   5264    60  95M *   0   0
TGCAGCTTGATATTGACACTTTGTTGATGTTGTCTGATTGTTTCTCTATCTTTCTATCTACTGATGAGGTGAATAAGATCTTTCTAGTATAAATA
#2=AAFHHJHIJJJJJJJJJJJJJJJEEHIIIGHJJJJIIHIIJJJIIJJJJJJJJIJIJJIJCHJJJI@DHHEIHHHHHHFFFFFFDEFFDDEE NM:i:2  MD:Z:18T56G19   AS:i:85 XS:i:0

1_1208_8720_16291_1 4   scaffold03512   14282   60  95M *   0   0
TGCAGATGTCTAAAATAAGAGATGAATAGTGTGGATAACTAATACATCAGGTATATTGTATGATTTGAAGTGACTACTTGTTTAGTTGTTGAATC
#2=AAFHHJJJJJJJJJJJJJJJJJJJJJFIFHIJJJJJJJJJJJJIJJIJJFHIIJJJHIJJJJJJJJJJIIJJIJHHHHHHFFFDFFFDEEEDD NM:i:5  MD:Z:7A3G2G8A59G11  AS:i:70 XS:i:0

1_1208_16525_26340_1    0   scaffold02569   239172  30  95M *   0   0
TGCAGATGTCTAAAATAAGAGATGAATAGTGTGGATAACTAATACATCAGGTATATTGTATGATTTGAAGTGACTACTTGTTTAGTTGTTGAATC
#24<<DHHIIIIIIIIIIEHIIIIIIGIIHHFGHIIIIIIIIIIIIIIIIIEFFHIIIIBFIIIIIIIIIIGHIIIIHGHGEHFFF@CEDCEEECC NM:i:5  MD:Z:7A3G2G8A59G11  AS:i:70 XS:i:0
```

Figure 5: Example of a SAM formatted output file

In the SAM file, columns are organized as follow:

- Read name (from the sequencing platform)
- Read orientation on the DNA strand (mapper specific numbering)
- Chromosome / scaffold name where the read mapped
- One-based left-most mapping position of the read on the scaffold (in basepair)
- Mapping quality (mapper specific, see manual for guideline on trimming)
- Cigar string describing the mapping (only substitutions, or insertions, deletions...)
- Ref. name of the mate/next read (only for paired-end)
- Position of the mate/next read (only for paired-end)
- Observed Template Length (only for paired-end)
- Sequence of the read
- Quality of the read
- 12th to xxth columns: List of optional fields

Optional fields are generated by BWA, some of which are specific to BWA. This list can change with the aligner you are using.

Tag	Meaning
NM	Edit distance
MD	Mismatching positions/bases
AS	Alignment score
BC	Barcode sequence
X0	Number of best hits
X1	Number of suboptimal hits found by BWA
XN	Number of ambiguous bases in the referenece
XM	Number of mismatches in the alignment
XO	Number of gap opens
XG	Number of gap extentions
XT	Type: Unique/Repeat/N/Mate-sw
XA	Alternative hits; format: (chr,pos,CIGAR,NM;)*
XS	Suboptimal alignment score
XF	Support from forward/reverse alignment
XE	Number of supporting seeds

Figure 6: Optional fields in BWA. Fields starting with X are speciic to BWA.

3.2 Post-alignment filtering

Now that the alignment was performed, it is possible to filter the SAM files according to the mapping quality. Mapping quality is affected not only by the number of mismatches/gaps between the read and the reference genome, but also by the number of additional alignment that were detected for that read. Indeed, it is possible that a read maps equally wellin two different genomic parts. In that case, BWA will map the read to the position with the highest mapping score, and will "downgrade" that mapping score because of the existence of an alternative mapping position. By setting a quality threshold on the reads we will use for downstream analysis, we will thus discard those reads.
The easiest way to perform quality filtering is to use Samtools (Li et al. 2009). SAM Tools provide various utilities for manipulating alignments in the SAM format, including sorting, merging, indexing and generating alignments in a per-position format.

The filtering option is available through the "view" parameter and the -q option in Samtools. In BWA, the maximum quality score is 60. Any reads with a mapping quality below 30 can be considered "bad quality" and can be discarded. Samtools can also translate your SAM format intoa BAM format (very useful since it reduces the sizeof your files).

```
samtools view -q 30 -bh input.sam -o output.bam
```

Here,we specify that we want to remove any reads with a MAPQ less than 30, and we want the output to be in BAM format, with the header (-bh). Once all your SAMfiles have ben filtered and translated into a BAM format, you are ready to run Stacks, through the "pstacks" pipeline.

4 Using STACKS to build loci

4.1 De Novo loci building with ustacks

This section only concerns users with no reference genome. If you used a reference genome and have BAM files, refer to the next section "Reference-based loci building with pstacks".
Ustacks calls for a two-steps processus. First, all reads are compared to each other and grouped in piles of exactly matching stacks. This step allows to reconstruct all the alleles present in an individual. Second, ustacks will compare the stacks to each other in order to reconstruct the loci. This particular step calls for user input of parameters to define what is an acceptable locus in term of distance between alleles, number of alleles per locus... Also, the user will have to define the minimum depth for which a Stack can be used and merged into a locus.

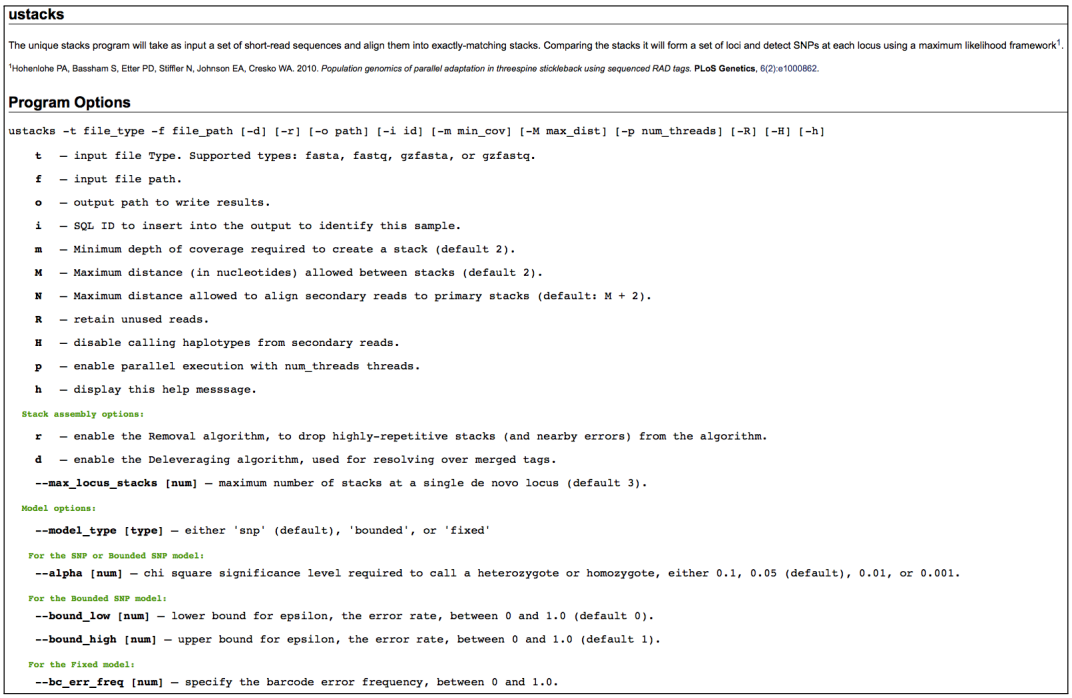
The commonly used parameters to run a De Novo analysis are:

```
ustacks -t fastq -f input/path/to/individual/files.fastq -m 4
  -M 2 -H -p 10 -o path/to/output/folder/ -r --
  max_locus_stacks 2 -i "number"
```

This command line specify that the input is a fastq format, gives the path to the fastq file to be treated, discard any stacks with a depth less than 4, refuses to merge stacks with more than 2 differences, disable calling haplotypes from secondary reads (reads that could not be grouped in a stacks and are singletons), enable the Removal algorithm to drop highly-repetitive stacks (and nearby errors) from the algorithm, and specifies that no more than 2 alleles should be found in each locus.

The specific options for ustacks are given in Figure 7.

The ustacks algorithm will generate three new files per individual: a file with the extension .tags.tsv that will describe each locus created, a file with the extension .alleles.tsv which lists all alleles and a file .snps.tsv which also lists SNPs for each locus



ustacks

The unique stacks program will take as input a set of short-read sequences and align them into exactly-matching stacks. Comparing the stacks it will form a set of loci and detect SNPs at each locus using a maximum likelihood framework[1].

[1]Hohenlohe PA, Bassham S, Etter PD, Stiffler N, Johnson EA, Cresko WA. 2010. *Population genomics of parallel adaptation in threespine stickleback using sequenced RAD tags.* **PLoS Genetics**, 6(2):e1000862.

Program Options

```
ustacks -t file_type -f file_path [-d] [-r] [-o path] [-i id] [-m min_cov] [-M max_dist] [-p num_threads] [-R] [-H] [-h]
    t  — input file Type. Supported types: fasta, fastq, gzfasta, or gzfastq.
    f  — input file path.
    o  — output path to write results.
    i  — SQL ID to insert into the output to identify this sample.
    m  — Minimum depth of coverage required to create a stack (default 2).
    M  — Maximum distance (in nucleotides) allowed between stacks (default 2).
    N  — Maximum distance allowed to align secondary reads to primary stacks (default: M + 2).
    R  — retain unused reads.
    H  — disable calling haplotypes from secondary reads.
    p  — enable parallel execution with num_threads threads.
    h  — display this help messsage.
  Stack assembly options:
    r  — enable the Removal algorithm, to drop highly-repetitive stacks (and nearby errors) from the algorithm.
    d  — enable the Deleveraging algorithm, used for resolving over merged tags.
    --max_locus_stacks [num] — maximum number of stacks at a single de novo locus (default 3).
  Model options:
    --model_type [type] — either 'snp' (default), 'bounded', or 'fixed'
  For the SNP or Bounded SNP model:
    --alpha [num] — chi square significance level required to call a heterozygote or homozygote, either 0.1, 0.05 (default), 0.01, or 0.001.
  For the Bounded SNP model:
    --bound_low [num] — lower bound for epsilon, the error rate, between 0 and 1.0 (default 0).
    --bound_high [num] — upper bound for epsilon, the error rate, between 0 and 1.0 (default 1).
  For the Fixed model:
    --bc_err_freq [num] — specify the barcode error frequency, between 0 and 1.0.
```

Figure 7: Ustacks options and line of command.

4.2 Reference-based loci building with pstacks

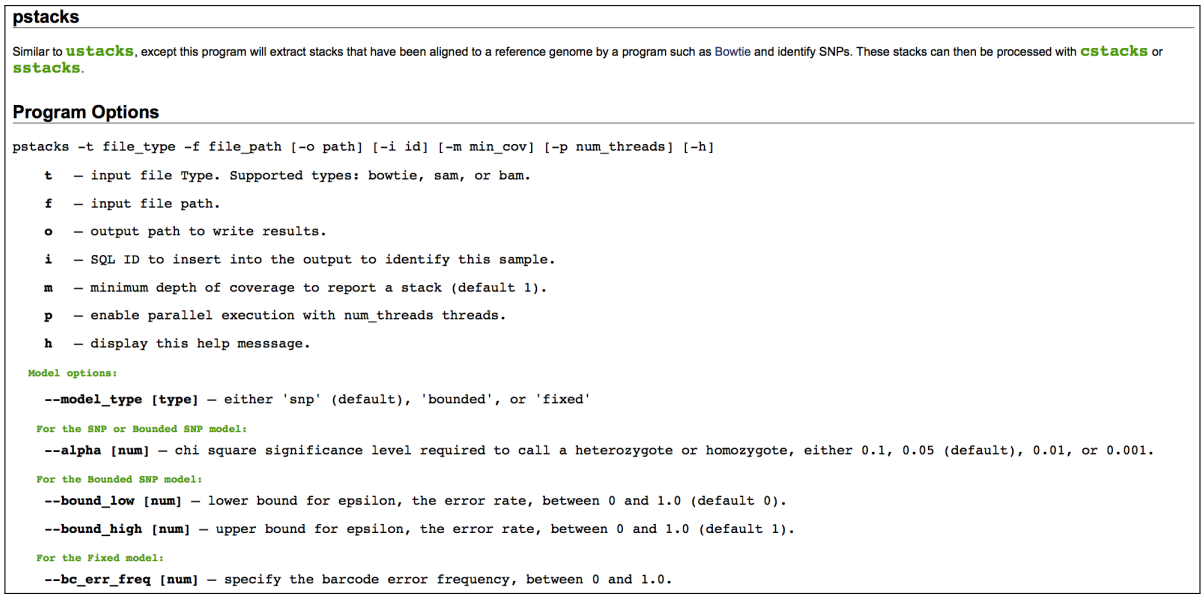
Now that your reads are mapped and you trimmed the resulting SAM/BAM file in order to remove low mapping scores, you are ready to use pstacks. The algorithm in pstacks will create stacks not by similarity, but by genomic coordinates of the left-most mapping position (beware, by opposition to Bowtie2, Stacks uses a zero-based numbering of nucleotides). Pstacks will then create the same three files per individuals that are created with ustacks.

The commonly used parameters to run pstacks are:

```
pstacks -t bam -f input/path/to/individual/files.bam -m 4 -p
    10 -o path/to/output/folder/ -i "number"
```

This line of command will execute the pstacks on BAM formatted files, removing loci for which less than 4 reads are reported. The specific options for pstacks are given in Figure 8.

The .tags.tsv file is very useful as it lists all of the loci and single-stacks found in your dataset. This means that you have access to all of what was sequenced/aligned on your genome, even the homozygous loci. Also, if your aim is to find particular SNPs and then try to blast the 100bp reads that hold them, the tags.tsv or the catalog file will be the one to go to to find the sequence.



pstacks

Similar to **ustacks**, except this program will extract stacks that have been aligned to a reference genome by a program such as Bowtie and identify SNPs. These stacks can then be processed with **cstacks** or **sstacks**.

Program Options

```
pstacks -t file_type -f file_path [-o path] [-i id] [-m min_cov] [-p num_threads] [-h]
    t  — input file Type. Supported types: bowtie, sam, or bam.
    f  — input file path.
    o  — output path to write results.
    i  — SQL ID to insert into the output to identify this sample.
    m  — minimum depth of coverage to report a stack (default 1).
    p  — enable parallel execution with num_threads threads.
    h  — display this help messsage.
  Model options:
    --model_type [type] — either 'snp' (default), 'bounded', or 'fixed'
  For the SNP or Bounded SNP model:
    --alpha [num] — chi square significance level required to call a heterozygote or homozygote, either 0.1, 0.05 (default), 0.01, or 0.001.
  For the Bounded SNP model:
    --bound_low [num] — lower bound for epsilon, the error rate, between 0 and 1.0 (default 0).
    --bound_high [num] — upper bound for epsilon, the error rate, between 0 and 1.0 (default 1).
  For the Fixed model:
    --bc_err_freq [num] — specify the barcode error frequency, between 0 and 1.0.
```

Figure 8: Pstacks options and line of command.

4.3 Creation of a catalog of loci with cstacks

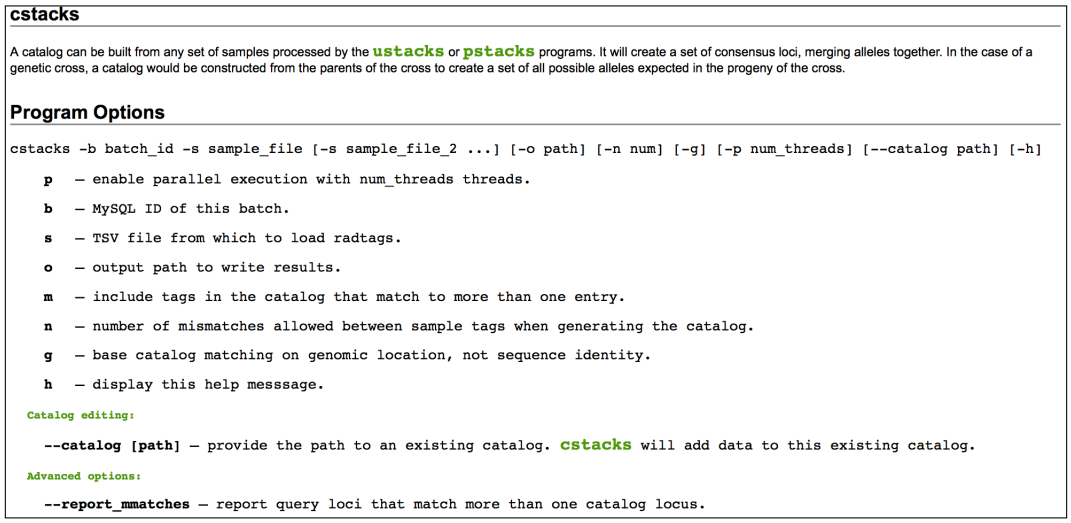
Now, for each individual, we have a list of reconstructed loci. We do know if they are polymorphic (through the alleles and snps files). Now we need to know if the loci found in one individual are also found in the other individuals. For this we need to create a catalog that will attribute the same "common" name to the same locus in each individual.

This step is relatively straightforward for loci coming out of the pstacks algorithm, since they are qualified and called as their mapping position. Hence, the task is easy for data obtained from pstacks, and the algorithm runs fast. You still have the different options to filter out particular patterns you do not want to see if your final dataset, such as multi-entry loci. When running cstacks after pstacks, you also ALWAYS need to use the parameter "-g" to tell cstacks you want to create the catalog based on genomic location.

For data obtained from ustacks, the process of assembling similar loci present in different individual calls again for a similarity approach. Loci from two different individuals will be considered to be the same locus if they do not differ by more than n nucleotides (the n parameter).

The commonly used parameters to run cstacks are:

```
cstacks -b 1 -s sample1 -s sample2 -s sample3 -m 2 -p 10 (and
  -g if coming from the pstacks algorithm)
```



cstacks

A catalog can be built from any set of samples processed by the **ustacks** or **pstacks** programs. It will create a set of consensus loci, merging alleles together. In the case of a genetic cross, a catalog would be constructed from the parents of the cross to create a set of all possible alleles expected in the progeny of the cross.

Program Options

```
cstacks -b batch_id -s sample_file [-s sample_file_2 ...] [-o path] [-n num] [-g] [-p num_threads] [--catalog path] [-h]
    p   — enable parallel execution with num_threads threads.
    b   — MySQL ID of this batch.
    s   — TSV file from which to load radtags.
    o   — output path to write results.
    m   — include tags in the catalog that match to more than one entry.
    n   — number of mismatches allowed between sample tags when generating the catalog.
    g   — base catalog matching on genomic location, not sequence identity.
    h   — display this help messsage.
  Catalog editing:
    --catalog [path] — provide the path to an existing catalog. cstacks will add data to this existing catalog.
  Advanced options:
    --report_mmatches — report query loci that match more than one catalog locus.
```

Figure 9: Pstacks options and line of command.

Note that apart from the "genotype" and the "population" algorithm, cstacks is the only algorithm for which all the samples are used in the command line. It is also important to notice that you do not have to create a new catalog for every library you will be sequencing. Indeed, let's imagine you already sequenced the library of a given population, did the analyses and obtained a catalog of loci. Now you sequenced another population but you are interested in comparing those two populations. You can then use the catalog already generated before and compare your new loci to those present in the catalog. This can be done through the option "–catalog" in which you can provide the path to your ready-to-go catalog file. The specific options for cstacks are given in Figure 9.

The cstacks algorithm will create three new files with the prefix "batch_", and the suffixe ".tags.tsv", ".snps.tsv" and ".alleles.tsv". When looking at the ".tags.tsv".

4.4 Attribution of the new catalog names to each individual with sstacks

Sstacks is a really fast algorithm that will allow to rename all loci within each individual according to its attributed catalog name.
For each locus in each individual, the sstacks algorithm will search the corresponding entry in the catalog generated by cstacks, and attribute its new name. In this algorithm again, the "-g" option is to be used for data obtained through reference-based mapping.

The commonly used parameters to run sstacks are:

```
sstacks -b 1 -c path/to/catalog -g -s sample/file/to/be/
   renamed -o output/path
```

The sstacks algorithm will generate one new file per individual, a file with the suffixe ".matches.tsv".

Here are the specific options for sstacks:

sstacks

Sets of stacks constructed by the **ustacks** or **pstacks** programs can be searched against a catalog produced by **cstacks**. In the case of a genetic map, stacks from the progeny would be matched against the catalog to determine which progeny contain which parental alleles.

Program Options

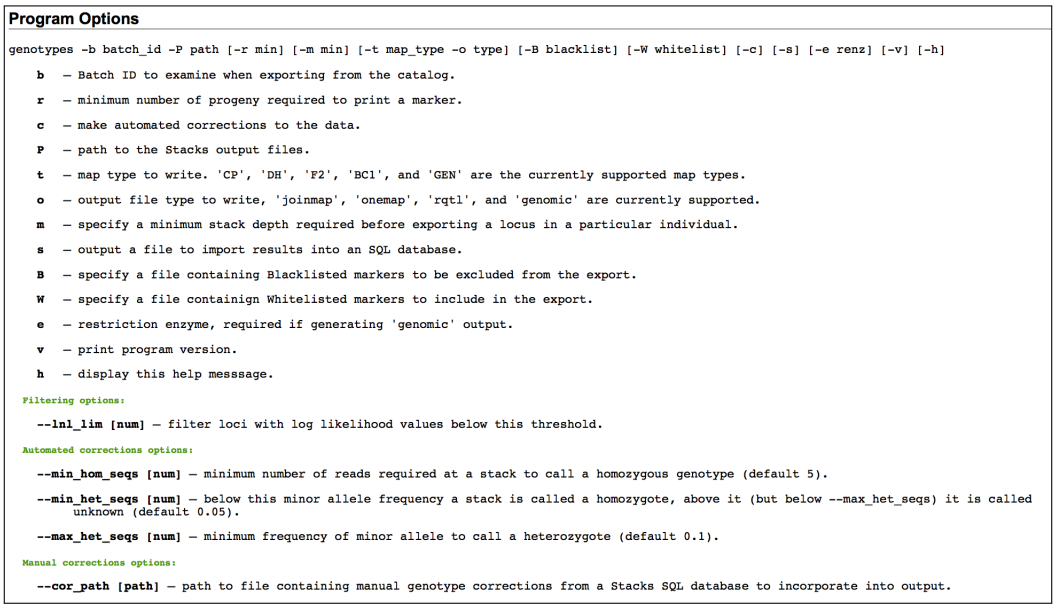
```
sstacks -b batch_id -c catalog_file -s sample_file [-s sample_file_2 ...] [-o path] [-p num_threads] [-g] [-x] [-v] [-h]

    p  — enable parallel execution with num_threads threads.
    b  — MySQL ID of this batch.
    c  — TSV file from which to load the catalog loci.
    s  — TSV file from which to load sample loci.
    o  — output path to write results.
    g  — base matching on genomic location, not sequence identity.
    x  — don't verify haplotype of matching locus.
    v  — print program version.
    h  — display this help messsage.
```

Figure 10: Sstacks options and line of command.

5 Using STACKS to obtain SNP data for building a genetic-map

Stacks can generate input files to be used in R/Qtl or Joinmap, in order to build a genetic map. The program "genotypes" exports a Stacks data set either as a set of observed haplotypes at each locus in the population, or with the haplotypes encoded into genotypes. Many options can be used to increase or lower the quality level of markers exported as genuine markers, for example depending on the limit set for the minimum number of individuals in which the locus has to be present (-r). The higher the limit, the larger the number of progeny in which the marker will be found, the higher the quality of the marker. Similarly, the -m option can be used to restrict exported loci to those that have a minimum depth of reads, increasing the genotype quality and hence the quality of the exported dataset.



Program Options

```
genotypes -b batch_id -P path [-r min] [-m min] [-t map_type -o type] [-B blacklist] [-W whitelist] [-c] [-s] [-e renz] [-v] [-h]

    b  — Batch ID to examine when exporting from the catalog.

    r  — minimum number of progeny required to print a marker.

    c  — make automated corrections to the data.

    P  — path to the Stacks output files.

    t  — map type to write. 'CP', 'DH', 'F2', 'BC1', and 'GEN' are the currently supported map types.

    o  — output file type to write, 'joinmap', 'onemap', 'rqtl', and 'genomic' are currently supported.

    m  — specify a minimum stack depth required before exporting a locus in a particular individual.

    s  — output a file to import results into an SQL database.

    B  — specify a file containing Blacklisted markers to be excluded from the export.

    W  — specify a file containign Whitelisted markers to include in the export.

    e  — restriction enzyme, required if generating 'genomic' output.

    v  — print program version.

    h  — display this help messsage.

  Filtering options:

    --lnl_lim [num] — filter loci with log likelihood values below this threshold.

  Automated corrections options:

    --min_hom_seqs [num] — minimum number of reads required at a stack to call a homozygous genotype (default 5).

    --min_het_seqs [num] — below this minor allele frequency a stack is called a homozygote, above it (but below --max_het_seqs) it is called
        unknown (default 0.05).

    --max_het_seqs [num] — minimum frequency of minor allele to call a heterozygote (default 0.1).

  Manual corrections options:

    --cor_path [path] — path to file containing manual genotype corrections from a Stacks SQL database to incorporate into output.
```

Figure 11: genotypes options and line of command.

Here is a list of the different options for genotypes:

Making Corrections

If enabled with the -c option, the genotypes program will make automated corrections to the data. Since loci are matched up in the population, the script can correct false-negative heterozygote alleles since it knows the existence of alleles at a particular locus in the other individuals. For example, the program will identify loci with SNPs that didn't have high enough coverage to be identified by the SNP caller. It will also check that homozygous tags have a minimum depth of coverage, since a low-coverage polymorphic locus may appear homozygous simply because the other allele wasn't sequenced.

Correction Thresholds

The thresholds for automatic corrections can be modified by changing the default values for the min_hom_seqs, min_het_seqs, and max_het_seqs parameters to genotypes. min_hom_seqs is the minimum number of reads required to consider a stack homozygous (default of 5). The min_het_seqs and max_het_seqs variables represent fractions. If the ratio of the depth of the the smaller allele to the bigger allele is greater than max_het_seqs (default of 1/10) a stack is called a het. If the ratio is less than min_het_seqs (default of 1/20) a stack is called homozygous. If the ratio is in between the two values it is is unknown and a genotype will not be assigned. Automated corrections made by the program are shown in the output file in capital letters.

6 Using STACKS to create a SNP table for population genetic study

6.1 Produce a VCF file (Variant Call Format) using the population program

Stacks can generate a file containing all the information of all the SNPs found in the dataset, following certain user specified filtering. The algorithm used for this task is called "populations". Populations was developed for population geneticists and can generate specific statistics in addition to a SNP file. It can also export the SNP data in different format, let it be Phylip, Fasta, VCF, Plink, Beagle...

The filtering options concern the minimum number of individuals required to report a locus, the minimum number of populations in which the locus has to be present to be reported, the minimum minor allele frequency in the sample set for a SNP to be reported...

In order to perform the filtering steps and the statistics, you will need to provide Stacks with a detailed description of how your samples belong to groupings/populations/species. This file is called the population map. It lists the The commonly used parameters to run populations are:

```
populations -b 1 -P path/to/all/files/generated/so/far -c path
  /to/catalog -g -s sample/file/to/be/renamed -o output/path
```

The population options and the data filtering parameters are given in Figure 11.

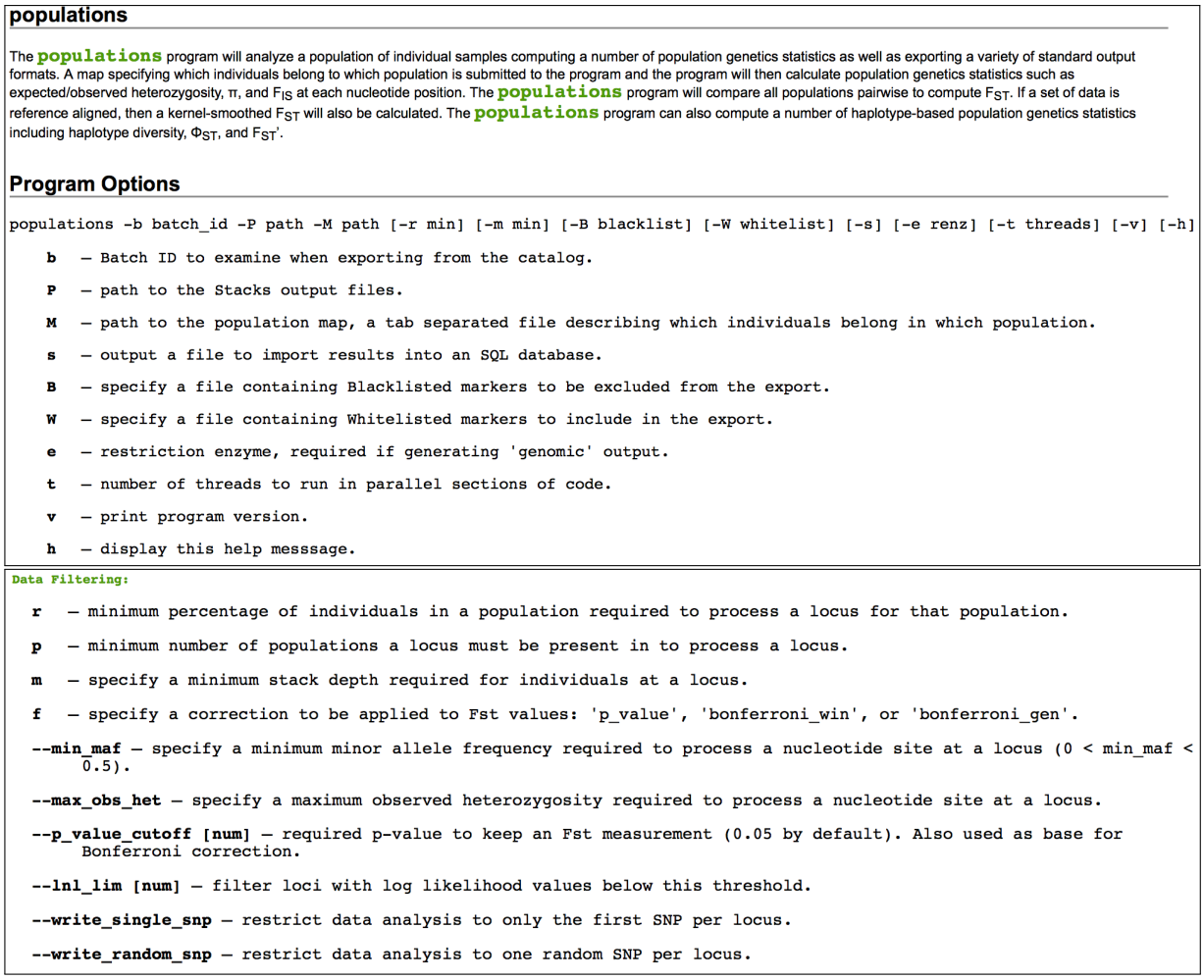
As mentioned, there is a quantity of format your file can be exported to. The most common one is VCF. Population genomics programs and packages such as ADEGENET, APE, and more and more are beginning to include the VCF format as an input format.

Here is a list of the input formats available in populations:

6.2 Extract the genotypes from the VCF file

If you try to open the VCF file in an excel spreadsheet, you will notice that the genotypes are coded in a binary manner, with 0/0 being homozygote for the reference (here the most common in the dataset) allele, 0/1 being heterozygote, and finally 1/1 being homozygote for the alternative (here less common) allele. These genotypes are hidden in the cells starting at column

J. Catchen, P. Hohenlohe, S. Bassham, A. Amores, and W. Cresko. Stacks: an analysis tool set for population genomics. Molecular Ecology. 2013



populations

The **populations** program will analyze a population of individual samples computing a number of population genetics statistics as well as exporting a variety of standard output formats. A map specifying which individuals belong to which population is submitted to the program and the program will then calculate population genetics statistics such as expected/observed heterozygosity, π, and F_{IS} at each nucleotide position. The **populations** program will compare all populations pairwise to compute F_{ST}. If a set of data is reference aligned, then a kernel-smoothed F_{ST} will also be calculated. The **populations** program can also compute a number of haplotype-based population genetics statistics including haplotype diversity, Φ_{ST}, and F_{ST}'.

Program Options

```
populations -b batch_id -P path -M path [-r min] [-m min] [-B blacklist] [-W whitelist] [-s] [-e renz] [-t threads] [-v] [-h]

    b   – Batch ID to examine when exporting from the catalog.
    P   – path to the Stacks output files.
    M   – path to the population map, a tab separated file describing which individuals belong in which population.
    s   – output a file to import results into an SQL database.
    B   – specify a file containing Blacklisted markers to be excluded from the export.
    W   – specify a file containing Whitelisted markers to include in the export.
    e   – restriction enzyme, required if generating 'genomic' output.
    t   – number of threads to run in parallel sections of code.
    v   – print program version.
    h   – display this help messsage.

Data Filtering:

  r   – minimum percentage of individuals in a population required to process a locus for that population.
  p   – minimum number of populations a locus must be present in to process a locus.
  m   – specify a minimum stack depth required for individuals at a locus.
  f   – specify a correction to be applied to Fst values: 'p_value', 'bonferroni_win', or 'bonferroni_gen'.
  --min_maf – specify a minimum minor allele frequency required to process a nucleotide site at a locus (0 < min_maf <
       0.5).
  --max_obs_het – specify a maximum observed heterozygosity required to process a nucleotide site at a locus.
  --p_value_cutoff [num] – required p-value to keep an Fst measurement (0.05 by default). Also used as base for
       Bonferroni correction.
  --lnl_lim [num] – filter loci with log likelihood values below this threshold.
  --write_single_snp – restrict data analysis to only the first SNP per locus.
  --write_random_snp – restrict data analysis to one random SNP per locus.
```

Figure 12: Populations options and line of command.

Li H. and Durbin R. (2009) Fast and accurate short read alignment with Burrows-Wheeler Transform. Bioinformatics, 25:1754-60.

Li H., Handsaker B., Wysoker A., Fennell T., Ruan J., Homer N., Marth G., Abecasis G., Durbin R. and 1000 Genome Project Data Processing Subgroup (2009) The Sequence alignment/map (SAM) format and SAMtools. Bioinformatics, 25, 2078-9.

```
File ouput options:

  --ordered_export – if data is reference aligned, exports will be ordered; only a single representative of each
       overlapping site.

  --genomic – output each nucleotide position (fixed or polymorphic) in all population members to a file.

  --fasta – output full sequence for each allele, from each sample locus in FASTA format.

  --fasta_strict – output full sequence for each haplotype, from each sample locus in FASTA format, only for
       biologically plausible loci.

  --vcf – output results in Variant Call Format (VCF).

  --vcf_haplotypes – output haplotypes in Variant Call Format (VCF).

  --genepop – output results in GenePop format.

  --structure – output results in Structure format.

  --phase – output genotypes in PHASE format.

  --fastphase – output genotypes in fastPHASE format.

  --beagle – output genotypes in Beagle format.

  --beagle_phased – output haplotypes in Beagle format.

  --plink – output genotypes in PLINK format.

  --phylip – output nucleotides that are fixed-within, and variant among populations in Phylip format for phylogenetic
       tree construction.

  --phylip_var – include variable sites in the phylip output.
```

Figure 13: Populations output format options.

7 Tutorial for De Novo SNP identification from populations data

7.1 Quality control

In the folder "data", you should find a file named "raw_reads.fastq.gz". This file is a fastq file that is zipped (using tar command). To unzip it, type in the terminal window:

```
cd /Path/to/data/folder/
tar zxvf raw_reads.fastq.gz
```

However, for quality control using FastQC, you can use the zipped version. Open fastQC on your computer