Disclaimer

About this pipeline

This pipeline has been developed so that we could genotype individuals at the MHCIIb locus. Due to the nature of our 454 sequencing data, more user friendly pipelines, such as jMHC, worked poorly. We are confident about the results this pipeline has produced for three of our projects. However, using the pipeline is clearly not a user friendly experience. It is very likely that, when using this pipeline with your dataset, you will have to modify and tweak parts of the pipeline. We would like to stress that any user attempting to use the ngs_genotyping pipeline should have a very solid experience with Linux and programming, mostly with Python and bash scripts.

If you choose to use ngs_genotyping, we hope that the pipeline will be usefull to you. If you encounter any problem, you can contact us with precise questions and detailed information about what went wrong and how (step, script, error message...). Help is not guaranteed and is completely dependent on current workloads, the complexity of the problem, and the Linux savyness of the person requiring help.

Your first source of help should be someone near you whom you can speak to directly and who is more knowledgeable about Linux, Python and bash than you are. A problem that is not evident to you at this moment could be trivial for one of your co-workers.

Introduction

The ngs_genotyping pipeline helps you determine the allelic genotypes of individuals for a specific gene. More specifically, a gene sequence is PCR amplified in a group of individuals. The amplified DNA is then labled using MIDs that identify the individual of origin of all the sequences and then these sequences are pooled and sequenced by 454 sequencing. A minimum of 100 individuals is suggested in order to produce good results. Using 200 individuals or more is a good way to increase allele detection and improve genotyping. The reads obtained from the 454 sequencing form the raw material for the ngs_genotyping pipeline. At least a few Sanger sequences are also important to have for allele cleanup, but these do not need to represent all possible alleles in the populations.

The different steps of the pipeline work sequentially to first find all the alleles that exist in the group of individuals and then assign genotypes to all the individuals. The pipeline was originally created to genotype around 400 individuals at the MHCIIb locus, which often has more than one copy in many species. This means that individuals may have more than 2 alleles. Another characteristic of this gene is the propensity for artefactual alleles to be created in one or more of the PCR steps. These artifacts, called chimeras, are very difficult to distinguish from the true alleles. This pipeline includes multiple measures to help eliminating these. It is also highly suggested to have a certain proportion of individuals that are prepared twice in an independent manner and also sequenced twice. If you have the ressources, up to 10% of the individuals should be resequenced. These will be essential to validate your genotyping results.

The pipeline will only work under Linux. It has been tested with Ubuntu 12.04 and previous versions of Ubuntu. It should be noted again that a person without a decent knowledge of Linux and the bash command line is unlikely to be able to complete the whole process. If you have no or poor knowledge of Linux, you will definitely need assistance from someone near you that has a more solid Linux background. The pipeline was created because no other satisfactory solution was available for us to

complete our analysis and it has solved our problem perfectly. We make, however, not claim about its usefuleness to other groups or individuals.

Please note that, although great care has been taken in writing the present documentation to make the pipeline as easy as possible to use by providing enough details, the use of the pipeline itself is not user friendly and we cannot provide support for technical problems that could occur during its use, such as some of the packages used in the pipeline not working or not being supported in future versions of Ubuntu.

The pipeline is rather long and it is very important to read this documentation with attention to details. Read the whole document quickly a first time to have a first impression of the different steps. **When you are ready to proceed, read the entire text of each step before doing it!**

How to use

For each new experiment, use a freshly decompressed folder architecture from 'ngs_genotyping#.tar.gz', where '#' represents the pipeline version number. This folder architecture is required for the different scripts to work properly. **When you are going to use the scripts or terminal commands, always use the 'ngs_genotyping' directory of your experiment as the present working directory. All the commands found in this document are made to work from this directory.**

Before starting, here is an overview of the steps that the pipeline will take you through:

```
STEP 0   - Making the scripts executable
STEP 1   - Preparing the sequence files and renaming the sequences
STEP 2   - Trimming sequences based on quality and length
STEP 3   - Separating the sequences by MID into different files
STEP 4   - Removing individuals with low numbers of sequences
step 4.5 - subsample individuals with too many sequences (>1000)
STEP 5   - Aligning the sequences and cleaning sequencing errors
STEP 6   - Determining alleles for the individuals with HIGGY Pop
STEP 7   - Aligning the putative allele sequences and cleaning errors
STEP 8   - Creating a dictionary of putative concensus alleles
STEP 9   - Preparing data and parameters for the genotyping step
STEP 10  - Genotyping the individuals
STEP 11  - Formating the output
```

STEP 0 - Making the scripts executable

We need to make sure that our scripts are executable before we run the steps of the pipeline. Open a terminal and move to the 'ngs_genotpyping' directory you created for your new project by uncompressing the tar.gz archive. Eg:

```
cd /home/john/projects/drosophila/ngs_genotyping1.4
```

Run the following commands:

```
chmod +x ./scripts/*.py
chmod +x ./scripts/*.sh
```

Now the scripts are executable. **Each time we use one of the scripts, we will launch it from the 'ngs_genotyping' folder.** We will have to specify the relative path of the script. For example:

```
./scripts/03_fasta_separate_mids.py
```

Would launch that specific script. To test the installation, launch that last command. You should get an output lile the following:

```
No options specified
Use -h for help
```

STEP 1 - Preparing the sequence files and renaming the sequences

In this step, we will rename the sequences within each individual fasta and quality file according to the MID of the individual. For this, the file names **must** have a format that look like the following:

```
*[2 digits].MID[1 to 6 digits]*.[extension]
```

Where * stands for any string of characters and [extension] stands for .fna and .qual. **It is crucial that both .fna and .qual be present and they need to be in the 'raw_data' folder.** For example, our individual 152 from lane 03 has the following files:

```
HAI7FSR03.MID152.fna
HAI7FSR03.MID152.qual
```

The name must contain two digits, followed by '.MID' and then one or more digits. The '.MID' part is used to find the two digit sections, that are used together to uniquely identify an individual from which the data originated. Put all the files in a folder named 'raw_data'.

To use the scripts, we will need the BioPython library, the muscle aligner, the gnuplot visualization program, which in Ubuntu we instal with:

```
sudo apt-get install python-biopython muscle gnuplot gnuplot-x11
```

In order to proceed, we will launch the the script for step 01. First, open a terminal and move to the 'ngs_genotyping' directory of your project and launch the following command:

```
./scripts/01_data_preparation.sh
```

From the distribution of lengths observed in the graph produced by the script, **determine what is a likely minimal valid length threshold for your sequences and note this value.** We are going to use it in the next step.

All the sequences have been identified to their individual or origin and their information is found in the two fllowing files:

```
/raw_data/all_identified.fna
/raw_data/all_identified.qual
```

STEP 2 - Trimming sequences based on quality and length

In this step, use your favorite software to trim the sequences and remove sequences with low quality. You can use whatever software you normally use for this purpose, as long as you can produce a fasta file with the trimmed sequences of all the individuals combined.

The two files produced in the preceding section must be used for this step:

```
/raw_data/all_identified.fna
/raw_data/all_identified.qual
```

When you are done, put all the prepared sequences together in a file named 'all_trimmed.fasta' in the folder named 'trimmed_separated_sequences'. **Do not remove the file named 'midnames.txt' from that folder.** It has been generated automatically.

We normally use CLC Genomics Workbench and perform the following actions:

- Create new folder for your project in CLC
- Import your sequences with 'NGS import' in the new folder
- Choose all_identified.fna AND all_identified.qual
- DO NOT Discard (uncheck boxes if necessary) read names or quality scores
- Save the new data set

- From the toolbox, go in NGS core tools and choose Trim Sequences
- A good starting quality limit to start exploring is 0.01. You can go up if you loose too many sequences
- Use '0' for the Maximum number of ambiguities
- Click both '454 Sequence Primer' for primers A and B
- Do not remove 5' or 3' nucleotides
- Discard reads below the threshold that you have determined by looking at the distribution of sequence lengths in STEP 1. Because the trimming for uncalled nucleotides ('N's) is done before the trimming for the length and some sequences may end up being shortened by that first step, **you can use a length threshold that is about 20bp below what you saw in the distribution.**
- Create a report and save the results
- Take a look at the report to make sure you are satisfied with the distribution of the sequence lengths (make sure you didn't cut off the distribution of desired reads) and the number of sequences retained. Redo the trimming to adjust if needed.

- Export the trimmed sequences as 'all_trimmed.fasta' using the Export button
- Put them in the folder named 'trimmed_separated_sequences'
- Do not remove the file named 'midnames.txt' from that folder

STEP 3 - Separating the sequences by MID into different files

In this step, we use the script named '03_fasta_separate_mids.py' to separate the data from the different individuals. First, make sure you copied the file named 'midnames.txt' that was created in the 'raw_data' folder during STEP 1 into the 'trimmed_separated_sequences'. Then, open a terminal, and launch the following command:

```
./scripts/03_fasta_separate_mids.py -i
     trimmed_separated_sequences/all_trimmed.fasta -m
     trimmed_separated_sequences/midnames.txt
```

All the prepared sequences have now been separated in different files for all the individuals.

STEP 4 - Removing individuals with low numbers of sequences

In this step, we want to get rid of any individual (MID) that has too few sequences. A good way of determining this is to look at the distribution of the number of sequences per MID, which can be done

with the following command:

```
    grep -c ">" trimmed_separated_sequences/MID*.fasta | sort -nr -t ":" -k 2 |
sed 's/:/\t/'
```

For a gene with only one copy in the genome (maximum of 2 alleles), we recommend a minimum of about 100 sequences. For genes with 2 copies, you may need 150 or more sequences per individual.

Now sort the files in the folder containing the fasta files by size in ubuntu and delete the files of the unwanted individuals from the 'trimmed_separated_sequences' folder before proceeding to the next step.

STEP 4.5 - Sub-sampling individuals with more than 1000 sequences

STEP 4.5 is not currently totally implemented. The following can be used as a guideline.

Insuring that individuals have a maximum of 1000 sequences is going to speed some of the next steps and is necessary for others. In order to randomly sub-sample 1000 sequences from problematic individuals, use the following command:

```
    grep -c ">" ./fasta/MID*.fasta | sort -nr -t ":" -k 2 | sed 's/:/\t/' | \
    perl -ane 'if ($F[1] > 1000) {print $F[0]."\n"}' | while read i; \
    do mv $i $i.too_many_sequences; done

    ### Use the files ending in 'too_many_sequences' and pull out 1000 random
    ### sequences from them. An alternative, easy solution is to only keep
    ### the first 1000 sequences.
```

STEP 5 - Aligning the sequences and cleaning sequencing errors

In order to clean sequencing errors, we produce sequence alignments from the starting sequences. We are going to use a few cycles of aligning the sequences and running a custom script to do some cleaning. **This step may take up to a few hours** since the alignment steps consume a lot of time and there are typically a lot of individuals to process. Lauch the following command:

```
    time ./scripts/05_align_clean_align.sh 0.01 0.01
```

The two decimal values represent the lowest proportion of true SNPs and indels, respectively, that are accepted and that should be left uncorrected. These are proportions, so 0.01 equals 1%. SNPs and indels present at proportions lower than these values will be corrected to the majority nucleotide for that position. Different values should be tested with a subset of the individuals as the characteristics of the data is going to vary from experiment to experiment as well as with the sequencing method used. We suggest using only a subsample of individuals to run the tests since these can be time consuming. The 'time' command at the beginning of the line will tell you how much time the whole process took.

STEP 6 - Determining alleles for the individuals with HIGGY Pop

In order to use the R script named HIGGY_Pop.r, you will need to install R and some packages on your computer. To install R on a Linux machine running Ubuntu or Debian, use:

```
    sudo apt-get install r-base-core
```

Then, to launch R, open a terminal and simply type the capital letter R:

```
R
```

We will now install the R packages required by HIGGY_Pop.r with the following command in R:

```
install.packages(c("gee","ape","seqinr"))
```

Type 'y' and select a mirror site close to your location when prompted and the packages should install automatically. To launch the HIGGY_Pop script, open a terminal and move to the 'ngs_genotyping' directory, open a terminal, launch R (type 'R') and launch the following command:

```
source('./scripts/HIGGY_Pop.r')
```

Follow the instructions for phase 1. The first time you go through this phase with a new dataset, we highly suggest that you use the option to have full control over each individual at least once. This will help you to decide on appropriate parameters to treat your dataset or to find problematic individuals that should be processed with different parameters. For each individual, an output file will be created containing all the alleles of that individual. The folder containing the fasta files is 'cleaned_aligned'.

STEP 7 - Aligning the putative allele sequences and cleaning errors

After the HIGGY_pop.r script is done, exit R with 'quit()' and concatenate all the *.fas files into one with the following command:

```
cat cleaned_aligned/*.fas > individual_alleles_higgy_phase_1.fas
```

We now need to trim uninformative portions on both extremities and remove gaps that are clearly caused by sequencing errors in your sequences.

Here is how we normally do it, but you can use your favorite sequence editor:

Add sanger sequences for the gene of interest at the beginning of the file manually (use gedit) and load the sequence file in Mega or Geneious. Align the the sequences using the muscle algorithm. Trim uninformative portions on both extremities and remove gaps that are clearly caused by sequencing errors in your sequences. Beware of indels that add up to multiples of three as these may represent valid codon deletions. Remove the sanger sequences from the alignment then export and export the new alignment as 'input_phase2.fas'.

Open 'input_phase2.fas' in the text editor gedit and use 'File --> Save as...' to modify the format from Windows to Linux if needed.

STEP 8 - Creating a dictionary of putative concensus alleles

To launch the HIGGY_Pop script, open a terminal, launch R (type 'R') and type the following command:

```
source('./scripts/HIGGY_Pop.r')
```

Follow the instructions for phase 2. Output will be written in a file named 'allele_database.fasta'. We are going to modify that file, but first make a copy of it in case you need to go back to it:

```
cp allele_database.fasta allele_database_original.fasta
```

STEP 9 - Preparing data and parameters for the genotyping step

Copy all your individual fasta files into 'ngs_genotyping/fasta' with the following command.

```
cp trimmed_separated_sequences/MID*.fasta fasta/
```

Now, we need to install the blastplus program from NCBI. Go to:

```
ftp://ftp.ncbi.nlm.nih.gov/blast/executables/blast+/LATEST/
```

and grab the version that suits your computer. It should look like this:

```
ncbi-blast-2.2.28+-x64-linux.tar.gz
```

or:

```
ncbi-blast-2.2.28+-ia32-linux.tar.gz
```

Put the downloaded file on your Desktop (home/yourname/Desktop), open a termial, make the Desktop the current working directory, and type the following (you may have to replace the version number in the file names):

```
tar xvfz ncbi-blast-2.2.28+-x64-linux.tar.gz
cd ncbi-blast-2.2.28+/bin
sudo cp * /usr/local/bin
```

Steps 9.1 to 9.6 are crucial to determining which alleles are present in your samples and to eliminate artefactual alleles. We will run the same command 3 times and do some quality control between the steps. Two versions of that command are available, '09_blast_script.sh' and '09_blast_script_proportions.sh'. Use the first one first and then try to use the second one. They use a different metric to determine thresholds to genotype the alleles. After each use of the script in steps 9.1, 9.3 and 9.6, we will ajust some parameters and make decisions about the validity of the alleles, as outlined in steps 9.2, 9.4, and 9.5 :

9.1 - Run the following script:

```
./scripts/09_blast_script.sh
```

```
or, for proportions:
```

```
./scripts/09_blast_script_proportions.sh
```

9.2 - A lot of figures should pop up on your desktop, some with points (plus signs) and some with lines. Look at the graphs showing the distribution of evalues for each allele (those with lines) and determine a minimum global evalue to be used for the blasts. Since all the alleles have the same length, the blast values should all fall in the same range. You must decide where to cut to eliminate evalues that are dubious because they are too low. These evalues could be caused by reads with a lot of errors or chimeras in your 454 data.

Modify the evalue on line 3 of '09_blast_script.sh' or '09_blast_script_proportion.sh', depending on which one you use. For example, if you desire a threshold of 1e-105, write: evalue="105". At this point, if you identify alleles that should be removed because they have e-value profiles that are strange (eg: much lower than the other alleles), remove these alles from the file named 'allele_database.fasta'.

9.3 - Run the following script again:

```
./scripts/09_blast_script.sh
```

```
or, for proportions:
```

```
./scripts/09_blast_script_proportions.sh
```

9.4 - Look at the graphs with points (plus signs). These graphs show the decreasing number of reads matching to each allele in individuals that have reads that blast to the allele. For each allele, determine a minimum number of sequences (or proportion of sequences if you use the proportion version of the script) needed to genotype an individual as possessing that allele. This is done for each allele and you must modify the threshold value corresponding to each allele manually in in the first column of the file named 'individual_thresholds.txt'. The default value is 2 sequences but it must be modified. Thresholds from 10 to 100 or more are possible, depending on your sequencing depth, the distribution of the number of sequences per individual and how much certain alleles could have been favored in the PCR amplification prior to the sequencing. For some datasets, using '09_blast_script_proportion.sh' makes the thresholds more obvious and easier to choose.

```
grep ">" allele_database.fasta | sed -E 's/>(A_[0-9]+).+?$/\1/; s/>//' | while
read allele; do echo -e "$allele\n---"; for ind in $(grep " $allele$"
./individual_summary/*summary.proportions | perl -pe 's/:/ /' | sort -t " " -k 2
-nr | perl -pe 's/ .+//'); do echo $ind; cat $ind; done; done | less
```

Make a copy of the 'individual_thresholds.txt' file (add '.good' to it) because step 9.6 will overwrite the file:

```
cp individual_thresholds.txt individual_thresholds_good.txt
```

9.5 – From the graphs of evalues and number of sequences (or proportions), you may decide that certain alleles are artefactual. For example, they could be found only in 1 or 2 individuals, be supported by too few sequences in individuals where they are found, or have very low evalues. The last two reason could be indicative of PCR chimera molecules. If you identify alleles that should be removed, remove them from the file named 'allele_database.fasta'.

9.6 - Run the following script one last time:

```
./scripts/09_blast_script.sh
```

```
or, for proportions:
```

```
./scripts/09_blast_script_proportions.sh
```

STEP 10 - Genotyping the individuals

To create the raw genotype file, use:

```
./scripts/10_genotype_from_blast_results.py summary_file_names.txt
individual_thresholds_good.txt genotypes.txt
```

```
or, for proportions:
```

```
./scripts/10_genotype_from_blast_results_proportions.py
summary_file_names_proportions.txt individual_thresholds_good.txt genotypes.txt
```

STEP 11 - Formating the output

To format the raw genotype file into two more friendly formats, use:

```
./scripts/11_format_genotypes.py genotypes.txt
```

The results can be found in the two following files:

```
genotypes_output_list.csv
genotypes_output_table.csv
```

It is a good idea to confirm that the genotypes of the individuals that were sequenced twice are reproducible. If not, one possible reason among many is the presence of an artefactual allele in one of the replicates and not another. If you identify artifacts at this stage, you can go back to step 9 armed with this information, remove the allele from the file named 'allele_database.fasta' and relaunch the script like in point 6 of step 9.