Script for SNP calling and evaluation

January 2024

Ovidiu Paun, ovidiu.paun@univie.ac.at

To get an introduction into SNP calling we will work with data from the plant genus *Tillandsia*, from the pineapple family (Bromeliaceae). Our biological aim here is to reconstruct phylogenetic relationships between some accessions based on genomic sequences, but we will use as a reference only 1 chromosome, namely the smallest chromosome of *T. fasciculata*.

NB: For each of these aims several analytical steps are required. For one particular aim there may be several procedures (pipelines) possible.

PART 0: SETTING UP YOUR WORK SPACE

Open a Terminal with shortcut Ctrl+Alt+T or click the terminal icon. Connect to molsysbio.cica.es with SSH. First create a sudirectory called SNPcall into your home directory and copy the data (the home folder is symbolized by ~)[1]:

```
$cd ~/student_name/ (replace student_name with your name)
$mkdir SNPcall
$cd SNPcall
$cp -r ../ovidiu/2copy/* ./
```

Let's check what is in our folder:

```
$ls -lth
```

PART I: MAP READS TO A REFERENCE (ONLY EXEMPLIFIED, NOT DONE IN CLASS)

The first step after read quality assessment and filtering is to map the data to a reference genome. We will not do this step together, but just for completeness, here is an example how mapping can be done with BWA[2]. This is a software package for mapping low-divergent sequences against a large reference genome.

First the reference needs to be prepared for alignment by generating a genome index from fasta files.

```
$bwa index -a is ./reference.fasta
```

Then the reads are mapped to the reference, outputting the results as SAM. Here fasta or fastq read files can be used, but they should be quality trimmed/filtered. BWA-MEM is the latest mapping algorithm and it is generally recommended for high-quality queries as it is faster and more accurate.

```
$bwa mem -t 6 ./reference.fasta ./input.trimmed_P1.fq
 ./input.trimmed_P2.fq > ./output_mapped.sam
```

[1] The symbol $ is used to show the cursor, i.e., you should not type it in the terminal.
[2] https://doi.org/10.1093/bioinformatics/btp324

It is better for further processing to sort the file by coordinates of the reference and transform it to BAM (smaller file size) with PICARD[3]:

```
$java -jar picard.jar SortSam I=./output_mapped.sam
O=./output_mapped.bam SO=coordinate
```

PART II: VARIANT DISCOVERY WITH GATK[4]

We will exemplify SNP calling using bam files of reads already mapped to the smallest chromosome from the reference genome of *Tillandsia fasciculata*. The mapping has been already done with BWA MEM similarly as above.

The Genome Analysis Toolkit (GATK) is a standard for variant calling and filtering. This starts from mapped files (generally BAM) and produce a VCF file with SNPs and indels, which further needs to be filtered.

We will do some of the steps with a single file, and then combine with other files that are provided already prepared. Before using GATK to call variants we have to prepare a "dictionary" and an "index" from the reference with other tools, such as PICARD[3] and SAMTOOLS[5]. PICARD is a JAVA based program, and such programs are called in general like "java -jar program.jar". In addition we will use the option -Xmx6g to limit the RAM usage to 6Gb during this analysis. PICARD has different submodules, like MarkDuplicates or AddOrReplaceReadGroups.

```
$cd ~/student_name/SNPcall/data/
```

```
$java -Xmx6g -jar ../programs/picard.jar CreateSequenceDictionary
R=./Tfas_chr25.fasta O=Tfas_chr25.dict
```

```
$samtools faidx Tfas_chr25.fasta
```

We will then add read groups (i.e., internal labels in the header of the BAM file):

```
$nice -n 19 java -Xmx6G -jar ../programs/picard.jar
AddOrReplaceReadGroups I=./Tionantha_B84_map.bam
O=./Tionantha_B84.gr.bam RGID=Tionantha_B84 RGLB=Tionantha_B84
RGPL=illumina RGPU=Tionantha_B84 RGSM=Tionantha_B84
```

To see the effect of AddOrReplaceReadGroups, check the end of the header of the BAM file before and after the analyses:

```
$samtools view -H ./Tionantha_B84_map.bam | tail
```

```
$samtools view -H ./Tionantha_B84.gr.bam | tail
```

The processed file has in the header a ReadGroup field "@RG" with the subfields ID, LB, PL, SM and PU. Next we will create indexes for all .gr.bam files with a "for loop":

```
$for file in *.gr.bam; do echo $file; samtools index $file; done
```

To speed up we will skip some (optional) steps like marking PCR duplicates (not really needed as PCR-free library prep) and realignment around indels (in general only a marginal effect on output). The next optional step would be base recalibration, which can make use of a previously known set of variants to validate base calibration. As we work with a non-model, no previous acurate information/database is available with known variants, so we will skip this step as well. We are hence ready to call variants. We will use the module UnifiedGenotyper

[3] https://broadinstitute.github.io/picard/
[4] https://doi.org/10.1101/gr.107524.110
[5] https://doi.org/10.1093/gigascience/giab008

from GATK (not the latest, but one of the fastest so appropriate for such short course). This step takes ca 30 min:

```
$nice -n 19 java -Xmx6G -jar ../programs/GenomeAnalysisTK.jar -T
UnifiedGenotyper -R ./Tfas_chr25.fasta -I Talbida_B26.gr.bam -I
Tfasciculata_B25.gr.bam -I Tionantha_B84.gr.bam -I
Tionantha_B43.gr.bam -I Tleoboldiana_24.gr.bam -I Tmima_20.gr.bam
-o ./1chr.vcf
```

Day2 SNPcalling

If needed: Copy the raw vcf file from me, and continue the rest of the practical part:

`$cd ~/student_name/SNPcall` (replace `student_name` with your name)

```
$cp ../ovidiu/2copy/1vcf* ./
```

Having a raw file of variants we can attempt to filter these variants with hard filters that are recommended by GATK: Fisher Strand values (FS > 60.0), by the Quality by Depth Values (QD < 2.0) and by the Mapping Quality (MQ < 40).

```
$java -Xmx6G -jar ../programs/GenomeAnalysisTK.jar -T
VariantFiltration -R ./Tfas_chr25.fasta -V 1chr.vcf --
filterExpression "QD < 2.0 || FS > 60.0 || MQ < 40.0" --filterName
"default" -o 1chrf.vcf
```

A VCF file is a text file (tabular) format for storing genotype data. It will include data for SNPs, but potentially also indels.

Let's open the beginning of the VCF file produced and inspect it:

```
$head -40 1chrf.vcf
```

Every vcf file has 3 parts: **i) meta-information lines** (beginning with "##"); **ii) a header line** (beginning with "#CHROM"); and **iii) data lines** containing marker and genotype data (one variant per line). A data line is called a VCF record. Each VCF record will have the same number of tab-separated fields as the header line. The symbol "." is used to indicate missing data.

Each **meta-information line** must have the form ##KEY=VALUE and cannot contain white-space. The first meta-information line must specify the VCF version number (version 4.2 in the example). Additional meta-information lines are optional, but are often included to describe terms used in the FILTER, INFO, and FORMAT fields.

The first nine columns of the header line and **data lines** describe the data:

CHROM the chromosome.

POS the genome coordinate of the first base in the variant. Within a chromosome, VCF records are sorted in order of increasing position.

ID a semicolon-separated list of marker identifiers.

REF the reference allele expressed as a sequence of one or more A/C/G/T nucleotides (e.g. "A" or "AAC")

ALT the alternate allele expressed as a sequence of one or more A/C/G/T nucleotides (e.g. "A" or "AAC"). If there is more than one alternate alleles, the field should be a comma-separated list of alternate alleles.

QUAL probability that the ALT allele is incorrectly specified, expressed on the phred scale (-10log10(probability)).

FILTER Either "PASS" or a semicolon-separated list of failed quality control filters.

INFO additional information (no white space, tabs, or semi-colons permitted).

FORMAT colon-separated list of data subfields reported for each sample. The format fields in the Example are explained below.

The **sample data** is stored after the ninth column. The remaining columns contain the sample identifier and the colon-separated data subfields for each individual. The most common format subfield is GT (genotype) data. In the sample data, genotype alleles are numeric: the REF allele is 0, the first ALT allele is 1, and so on. The second record contains a GP (genotype probability) format subfield, and the third record contains PL (phred-scaled genotype likelihood) format subfield.

We could count quickly how many variants are in the file:

```
$cat 1chrf.vcf | cut -f 7 | sort | uniq -c
```

We should try to evaluate our results, to get a better grasp of the data. We can use for example vcftools[6] and bcftools[7] to produce some basic statistics. First we will select with BCFTOOLS just SNPs that passed the filter in VariantFiltration/GATK.

```
$bcftools view -f PASS 1chrf.vcf > 1chrf_pass.vcf
```

We can next calculate with vcftools the missingness on a per-individual basis:

```
$vcftools --vcf 1chrf_pass.vcf --missing-indv --out 1chrf_pass
```

The output will carry a suffix .imiss. We can print the result on the screen vertically aligned:

```
$column -t -s $'\t' 1chrf_pass.imiss
```

Q: *Discuss the following question:*
What are the factors that can influence the level of missing data per individual in a SNP dataset?

You could also try the option `--missing-site` in VCFTOOLS, which will generate a report on a per site basis.

Let's remove the sites with missing data. Use VCFTOOLS with the option `--max-missing` which can take values from 0 to 1, where 1 means no missing data allowed.

```
$vcftools --vcf ./1chrf_pass.vcf --max-missing 1 --recode --out
final
```

Let's rename the output as the name is too long:

```
$mv final.recode.vcf final.vcf
```

Let's check the Ts/Tv ratio. This will be printed on screen.

```
$vcftools --vcf ./final.vcf --TsTv-summary
```

Q: *Discuss the following question:*
Why is TsTv ratio important and how can we improve it in a SNP dataset?

Let's look at the coverage of the data. To calculate the average coverage (depth) per individual:

```
$vcftools --vcf ./final.vcf --depth --out final

$column -t -s $'\t' final.idepth
```

We can calculate also the coverage per site:

[6] https://doi.org/10.1093/bioinformatics/btr330
[7] https://doi.org/10.1093/bioinformatics/btr509

```
$vcftools --vcf ./final.vcf --site-mean-depth --out final
```

Let's look at the min coverage per site:

```
$cut -f3 final.ldepth.mean | sort -n | uniq | head
```

Q: *Discuss the following question:*
What is the minimum and what is the maximum coverage per individual? Can you explain the coverage difference?
What is the min and max coverage per locus? Can you find explanations of the coverage range?

Let's calculate a measure of heterozygosity on a per-indivdiual basis. An inbreeding coefficient F will also be estimated for each accession. The output will have the suffix .het:

```
$vcftools --vcf ./final.vcf --het --out final
```

Q: *Discuss the following question:*
How can we explain the difference in heterozygosity between individuals?

For the analyses we want to use today we will focus only on loci that have the minor allele present in at least 2 individuals. We can filter the data to retain only those loci:

```
$vcftools --vcf ./final.vcf --maf 0.3 --recode --out finalF
```

Finally, we can check what is the Ts/Tv for the final dataset:

```
$vcftools --vcf ./finalF.recode.vcf --TsTv-summary --out finalF
```

Q: *Discuss the following question:*
Given the observed Ts/Tv ratio, how good is the data?

PART III: CONVERT FILE TO PHYLIP FORMAT.

We will convert the vcf file to a phylip format using VCF2PHYLIP[8]:

```
$../programs/vcf2phylip.py -i finalF.recode.vcf -m 2 -r --output-prefix final
```

PART IV: BUILD A PHYLOGENETIC TREE WITH RAXML[9].

RAxML is a standard tool for Maximum-likelihood based phylogenetic inference, particularly useful when dealing with data formed as concatenated SNPs. RAxML may complain that at least 2 sequences are identical in the file, but chose not to exclude them from the analysis. We will run ML plus rapid bootstrapping of 100 replicates, by taking a GTRGAMMA model (a general time reversible model including rate heterogeneity). This is just a quick way of getting a phylogenetic tree, but additional considerations should be observed if a solid phylogenetic tree needs to be produced.

```
$nice -n 19 raxmlHPC-PTHREADS-AVX -T 8 -f a -m GTRGAMMA -p 12345 -x 12345 -# 100 -s ./final.min2.phy -n tree
```

When the RAxML inference finishes, we visualize the tree at https://itol.embl.de/upload.cgi. Use cat to print in the terminal the output RAxML_bipartitions.tree on screen, copy it and paste it in the field "Tree text" of the itol website. Upload.

Reroot the tree with Tmima by clicking on the respective branch with the left mouse button, and selecting "Tree structure/Re-root the tree here". Change Label options in the Menu on the

[8] https://doi.org/10.5281/zenodo.2540861
[9] https://doi.org/10.1093/bioinformatics/btu033

right hand to Position “At tips”. To show bootstrap supports, go in the Menu on the right hand side to the Advanced tab, and select for Bootstrap/metadata “Display”, as text.