# Advanced Bioinformatics (7BBG2016): Practical Bioinformatics Data Skills

**Student ID: m2109260**

The main script for this assessment can be found on github here (also uploaded via canvas):

[https://github.com/m2109260/bioinformatics\_course/blob/main/scripts/m2109260\_AdvancedBioinformaticsAssignment.sh](https://github.com/m2109260/bioinformatics_course/blob/c5ece05fcbcc025848edc7a6fe8cb13ef49632d7/scripts/m2109260_AdvancedBioinformaticsAssignment.sh)

Alternative script can be found here:

[https://github.com/m2109260/bioinformatics\_course/blob/main/scripts/m2109260\_AdvBio\_Alternative.sh](https://github.com/m2109260/bioinformatics_course/blob/c5ece05fcbcc025848edc7a6fe8cb13ef49632d7/scripts/m2109260_AdvBio_Alternative.sh)

The R markdown and html files can be found here:

<https://github.com/m2109260/bioinformatics_course/tree/main/RScript>

## 1. Basic Linux and the command Line (20pts – 10% of final mark, each question provides 1 point)

1.1 What does ./../.. stand for ?  
cdB. Up one directory  
C. Up two directories  
D. None of Above

C – Up 2 directories

1.2 What does cd / mean in UNIX? Please explain what the cd command does.

cd command stands for Change Directory

cd / will change your current directory to your home directory.

1.3 What command would you use to get help about the command cp? (please provide an example command)

The man command.

e.g. man cp would give you information about the cp command

1.4 What does the command pwd do?

pwd = Print Working Directory. This command tells you where in the file system you are currently located.

1.5 How do you display a listing of file details such as date, size, and access permissions in a given directory? (please provide an example command)

the ls command with additional options

ls -l will give you data, size and access permissions.

1.6 How do you print on the terminal the first 15 lines of all files ending by .txt? (please provide an example command)

you can do this using the head command with -15 parameter and specify \*.txt

e.g. head -15 \*.txt

1.7 How do you rename a file from new to old? (please provide an example command)

using the move (mv) command followed by the original filename then the new filename.

e.g. mv new old

1.8 How do you display the contents of a file myfile.txt? (please provide an example command)

less myfile.txt

1.9 How do you create a new directory called flower? (please provide an example command)

mkdir flower

1.10 How do you change the current directory to /usr/local/bin? (please provide an example command)

cd /usr/local/bin

1.11 How can you display a list of all files in the current directory, including the hidden files? (please provide an example command)

ls -a

1.12 What command do you have to use to go to the parent directory? (please provide an example command)

cd ..

1.13 Which command would you use to create a sub-directory in your home directory? (please provide an example)

mkdir /newdirname

1.14 Which command would you use to list the first lines in a text file? (please provide an example)

head samplefile.txt

1.15 Which command will display the last lines of the text file file1? (please provide an example)

tail samplefile.txt

1.16 Which command is used to extract a column from a text file? (please provide an example)

the cut command

eg the following command will extract columns 2 and 4 from myfile.txt and save these columns in newfile.txt (it will not alter the contents of the original myfile.txt)

cut -f2,4 myfile.txt > newfile.txt

1.17 How do you copy an entire directory structure? E.g., from Project to Project.backup (please provide an example)

cp -r Project Project.backup

1.18 How would you search for the string Hypertension at the end of the line in a file called diseases.txt? (please provide an example)

grep diseases.txt “Hypertension”

1.19 How do you see hidden files in your home directory? (Please provide an example)

ls -a shows all files including hidden and the tilde (~) represents the home directory.

To view only the hidden files, we then need to filter for those that start with a dot then an alphanumeric character by piping the output to grep and using the regular expression ^\\.\\w (^ means “begins with”; \\. indicates a dot; [\\w](file:///\\w) means any alphanumeric character).

The full command is:

ls -a ~ | grep "^\\.\\w"

alternatively we could use ls -d ~/.\* but this will include the file file path.

1.20 How do you run a job that will continue running even if you are logged out? (Please provide an example)

The best option is to use the nohup command (short for “no hang up”) as this will allow a job to write standard error and output to a file and continue to receive commands from the shell. It simply stops the job being terminated when the shell is closed.

example:

nohup myjob.sh &

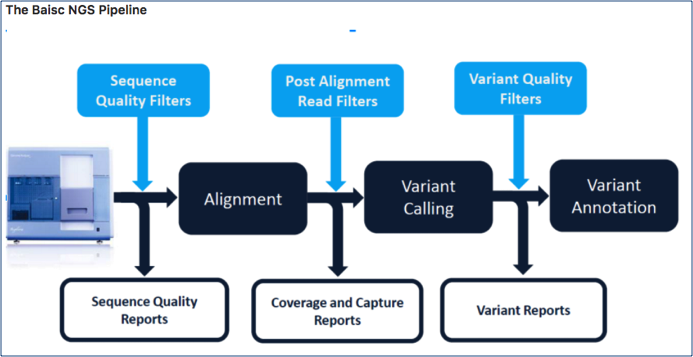
An alternative is to run a job in the background then use the disown command to remove it from the shell’s job management control. This will allow the job to continue running after you log out of the shell but has the downside that the job will no longer accept commands from the shell’s job management system.

## 2. The NGS Pipeline (65pts – 45% of final mark)

## 2.0 From raw data to alignment and variant calls (20pts)

The assessment is designed to:

* Test your ability to run standard NGS pipeline using the command line on a Linux system.
* Test your ability to create a Bash script that executes your NGS pipeline
* Test your basic knowledge of a standard NGS pipeline.



You have been provided with paired end fastq data and an annotation bed file from an Illumina HiSeq 2500 run. Using the assigned Openstack instance (please contact the module leaders if you have any problems with your Openstack instance), install the necessary tools and execute a standard Bioinformatics NGS pipeline to perform read alignment, variant discovery and annotation as described in the following NGS Pipeline section. **You are required to share a bash script that runs the workflow and takes the provided sequencing data as input (links provided below) with the examiner by uploading it with this report.** If uploading the script via Canvas or KEATS presents technical problems, you can also share the script by uploading it onto your github. **If you do so, please do not forget to provide the link to your github in the assignment and make sure you do not modify the file after the assignment deadline as this will show on github and will make the submission invalid.** Please make sure the bash script lines are adequately commented to provide a clear description of what it is doing. **The script will be evaluated by the examiner and up to 20pts will be given for a fully running and easy to read script.** Based on your pipeline, provide the following information and answer each question.

**Fastq Read 1 (~750MB):** <https://s3-eu-west-1.amazonaws.com/workshopdata2017/NGS0001.R1.fastq.qz>

**Fastq Read 2 (~750MB):**<https://s3-eu-west-1.amazonaws.com/workshopdata2017/NGS0001.R2.fastq.qz>

**Annotation File (10M):** <https://s3-eu-west-1.amazonaws.com/workshopdata2017/annotation.bed>

## 2.1 Install the tools and dependencies of your pipeline (using Miniconda when possible) and Download the input files (5 pts)

1. List the command lines to install all dependencies necessary to run the pipeline (3 pts)

I have downloaded a file **hg19\_chrom.bed** that this script requires from the UCSC Table Browser. The file was generated using the following steps:

1. Navigate to https://genome-euro.ucsc.edu/cgi-bin/hgTables
2. Change **group** dropdown to **All Tables**
3. Change **table** dropdown to **chromInfo**
4. Set **output filename** to **hg19\_chrom.bed**
5. Click **get output** button then download the file and move to the data/reference folder
6. run this command to remove the header line:

tail -n +2 hg19\_chrom.bed > hg19\_chrom.tmp && mv hg19\_chrom.tmp hg19\_chrom.bed

1. run the command to remove the last column out of 3:

cut -f 1,2 hg19\_chrom.bed > hg19\_chrom.tmp && mv hg19\_chrom.tmp hg19\_chrom.bed

It is stored in {REPO}/data/reference/hg19\_chrom.bed and needs to remain there for the script to run properly.

The following commands are used to install the required packages:

source ~/.bashrc

conda config --add channels defaults

conda config --add channels bioconda

conda config --add channels conda-forge

conda install samtools

conda install bwa

conda install freebayes

conda install picard

conda install bedtools

conda install trimmomatic

conda install fastqc

conda install vcflib

conda install bcftools

To install Annovar we have to request a copy from:

<https://www.openbioinformatics.org/annovar/annovar_download_form.php>

ANN\_DIR**=**"/home/ubuntu/annovar"

**$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar snp130 **$ANN\_DIR/**humandb**/**

**$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar refGene **$ANN\_DIR/**humandb**/**

**$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar knownGene **$ANN\_DIR/**humandb**/**

**$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar clinvar\_20180603 **$ANN\_DIR/**humandb**/**

**$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar exac03 **$ANN\_DIR/**humandb**/**

**#$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar dbnsfp31a\_interpro **$ANN\_DIR/**humandb**/**

then download the installation file and run tar -zxvf annovar.latest.tar.gz

We then need to download the databases that we want to use for annotation:

**wget** https**://**sourceforge.net**/**projects**/**snpeff**/**files**/**snpEff\_v4\_3t\_core.zip

**unzip** snpEff\_v4\_3t\_core.zip

To install snpEff we must execute the following commands from the home directory:

EFF\_DIR**=**"/home/ubuntu/snpEff/"

#download the ref genome for hg19/GRCh37

java **-**jar **$EFF\_DIR/**snpeff.jar download hg19

#download dbSNP database

**cd** **$DATA\_DIR/**reference

**wget** https**://**ftp.ncbi.nih.gov**/**snp**/**organisms**/**human\_9606\_b151\_GRCh37p13**/**VCF**/**00**-**common\_all.vcf.gz

**gunzip** -c 00**-**common\_all.vcf.gz **|** bgzip -c **>** dbSNP-b151-small.vcf.bgz

tabix -p vcf **$DATA\_DIR/**reference**/**dbSNP-b151-small.vcf.bgz

Next we must download the reference genome and a suitable version of dbSNP database. The dbSNP database is compressed but not with bgzip, which we need in order to pass a compressed version to snpEff, so we must run gunzip and pipe the output to bgzip with the -c flag to instruct it to read from standard input. Finally we index the file with tabix.

1. List all command lines necessary to download the input files (e.g. fastqs, reference genomes, etc) (2 pts)

FILE\_NAME**=**"NGS"

PROJECT\_DIR**=**"/home/ubuntu/bioinformatics\_course"

RESULTS\_DIR**=**"${PROJECT\_DIR}/results"

DATA\_DIR**=**"${PROJECT\_DIR}/data"

ALIGN\_DIR**=**"${DATA\_DIR}/aligned\_data"

STATS\_DIR**=**"${DATA\_DIR}/stats"

To make the code clearer and easier to read, modify and reuse, I have set up variables for the paths to different folders as well as a variable for the filename prefix. Note that these variables will be used in the rest of the code.

To download the input files:

**wget** https**://**s3-eu-west-1.amazonaws.com**/**workshopdata2017**/**NGS0001.R1.fastq.qz

**wget** https**://**s3-eu-west-1.amazonaws.com**/**workshopdata2017**/**NGS0001.R2.fastq.qz

**wget** https**://**s3-eu-west-1.amazonaws.com**/**workshopdata2017**/**annotation.bed

**mv** **\***fastq.qz ${DATA\_DIR}**/**untrimmed\_fastq

**mv** annotation.bed **$DATA\_DIR**

To download and index the reference genome before running the script:

**wget** http**://**hgdownload.cse.ucsc.edu**/**goldenPath**/**hg19**/**bigZips**/**hg19.fa.gz

bwa index ${DATA\_DIR}**/**reference**/**hg19.fa.gz

### 2.2. Pre-Alignment QC (4 pts)

1. Perform quality assessment and trimming (2pt)

We run fastqc to assess the quality of all the original read files, setting the -t flag to 4 to enable 4 threads to run the process and reduce time.

Then we trim the reads to remove low-quality bases that tend to appear at the end of reads and any contaminating adapter sequences. We run the trimmomatic command for paired-end reads (PE), using 4 threads in parallel (as for fastqc) and indicating that the read files use quality scores in the Phred 33 format. $R1 and $R2 are variables containing the names of the two read fastq files. The ILLUMINACLIP variable is machine-specific so must be set to the path to the nextera adapter for the machine that the script runs on.

The TRAILING parameter indicates that reads with a low score (below 25) at the “trailing” end should be trimmed.

fastqc -t 4 ${DATA\_DIR}**/**untrimmed\_fastq**/\*.**fastq.qz **\**

**>** **$STATS\_DIR/**untrimmed\_fastqc.log

nextera\_path**=**"/home/ubuntu/anaconda3/pkgs/trimmomatic-0.39-hdfd78af\_2/share/trimmomatic-0.39-2/adapters/NexteraPE-PE.fa:2:30:10"

trimmomatic PE **-**threads 4 **-**phred33 **\**

${DATA\_DIR}**/**untrimmed\_fastq**/**${R1} **\**

${DATA\_DIR}**/**untrimmed\_fastq**/**${R2} **\**

**-**baseout ${DATA\_DIR}**/**trimmed\_fastq**/**trimmed\_data **\**

ILLUMINACLIP**:$nextera\_path** TRAILING**:**25 MINLEN**:**50

MINLEN indicates that reads (after trimming) must be at least 50 base pairs, otherwise they should be discarded.

1. Perform basic quality assessment of paired trimmed sequencing data (2pt)

We now run the fastqc command (with multithreading) on the trimmed data and can manually compare the results to ensure that the trimming step fixed any issues we found.

#run FASTQC to assess quality of trimmed data

fastqc -t 4 ${DATA\_DIR}**/**trimmed\_fastq**/**trimmed\_data\_1P **\**

${DATA\_DIR}**/**trimmed\_fastq**/**trimmed\_data\_2P

### 2.3. Alignment (17pts)

* Align the paired trimmed fastq files using bwa mem and reference genome hg19 (edit your bwa mem step to include read group information in your BAM file) (9pts)

First step in the alignment is setting the Read Group information based on values in the BAM file:

ID: 11V6WR1.111.D1375ACXX.1

Sample Name: SM m2109260 (using my student ID)

Platform: PL ILLUMINA

Library Name: LB NGS-AdvBio

Date Run was Produced: DT 2017-01-01 (making an assumption here as we don't have the data)

Platform Unit: PU D1375ACXX.1

Then we call the aligner BWA-MEM, with 4 threads and a low level of warnings being reported, to align the trimmed reads to the reference genome.

read\_grp\_info**=**"@RG\tID:11V6WR1.111.D1375ACXX.1\tSM:m2109260\tPL:ILLUMINA\tLB:NGS-AdvBio\tDT:2017-01-01\tPU:D1375ACXX.1"

bwa mem -t 4 **-**v 1 -R **$read\_grp\_info** **-**I 250**,**50 ${DATA\_DIR}**/**reference**/**hg19.fa.gz **\**

${DATA\_DIR}**/**trimmed\_fastq**/**trimmed\_data\_1P **\**

${DATA\_DIR}**/**trimmed\_fastq**/**trimmed\_data\_2P **\**

**|** samtools view -h **-**Sb **-** **\**

**>** ${ALIGN\_DIR}**/**${FILE\_NAME}\_aligned.bam

The -I flags indicate and insertion and deletion “penalty” (these are set to standard values of 250 and 50). We tell the aligner the reference genome to use to map the reads against and pass in the trimmed, paired-end reads. Finally we pipe the output of bwa mem to samtools view to convert to a binary bam file. Piping allows us to avoid generating a very large uncompressed intermediary file. The -Sb flag indicates that input is SAM and the output should be binary. The ‘-‘ that follows -Sb indicates that the input should be read from standard input.

Now we sort and index the bam file for later.

samtools sort ${ALIGN\_DIR}**/**${FILE\_NAME}\_aligned.bam **>** ${ALIGN\_DIR}**/**${FILE\_NAME}\_sorted.bam

samtools index ${ALIGN\_DIR}**/**${FILE\_NAME}\_sorted.bam

* Perform duplicate marking (2pts)

picard MarkDuplicates I**=T**${ALIGN\_DIR}**/**${FILE\_NAME}\_sorted.bam \

O**=**${ALIGN\_DIR}**/**${FILE\_NAME}\_sorted\_marked.bam \

M**=$STATS\_DIR/**marked\_dup\_metrics.txt

samtools index ${ALIGN\_DIR}**/**${FILE\_NAME}\_sorted\_marked.bam

Duplicate reads can occur during sequencing for a number of reasons, such as during PCR amplification and library preparation. These are anomalies and need to be removed. To mark the duplicate molecules/reads, we use the picard MarkDuplicates tool. This tool will only mark the duplicate reads, it will not remove them.

* Quality Filter the duplicate marked BAM file (2pts)

samtools view **-**F 1796 **-**q 20 -o **$ALIGN\_DIR/**${FILE\_NAME}\_sorted\_filtered.bam **$ALIGN\_DIR/**${FILE\_NAME}\_sorted\_marked.bam

samtools index ${ALIGN\_DIR}**/**${FILE\_NAME}\_sorted\_filtered.bam

Now we filter out reads below a quality score of 20 ( -q 20 ) and set a filter flag of 1796, which is a bit flag specifying that the following should be removed:

0x4 – unmapped reads

0x100 – secondary duplicates

0x200 – QC failures

* Generate standard alignment statistics (i.e. flagstats, idxstats, depth of coverage, insert size) (4pts)

#flagstats - compare marked to filtered

#INPUT: output from picard mark duplicates then samtools view read filtering

samtools flagstat **$ALIGN\_DIR/**${FILE\_NAME}\_sorted\_marked.bam **>** **$STATS\_DIR/**marked.flagstat

samtools flagstat **$ALIGN\_DIR/**${FILE\_NAME}\_sorted\_filtered.bam **>** **$STATS\_DIR/**marked\_and\_filtered.flagstat

#Generate alignment statistics per chromosome (samtools)

samtools idxstats **$ALIGN\_DIR/**${FILE\_NAME}\_sorted\_filtered.bam **\**

**>** **$STATS\_DIR/**marked\_and\_filtered.idxstats

#Determine the distribution of insert sizes - picard

picard CollectInsertSizeMetrics I**=$ALIGN\_DIR/**${FILE\_NAME}\_sorted\_filtered.bam **\**

O**=$STATS\_DIR/**marked\_and\_filtered\_insertsize.stats **\**

H**=$STATS\_DIR/**marked\_and\_filtered\_insertsize.pdf

There are a variety of tools available to examine the results of the alignment and check on the integrity of the resulting files. Here we implement samtools flagstat, idxstats, picard Collect InsertSizeMetrics and bedtools coverage.

flagstat: produces a range of statistics on the reads including duplicates and mapped vs unmapped reads. Here we can use it to check that duplicate reads have been removed.

idxstats: produces alignment statistics per chromosome so can be used to check that we have reads across all the chromosomes we are expecting.

CollectInsertSizeMetrics: produces statistics about the insert sizes (the distance between the ends of the two paired end reads), ensuring that the sizes are in the expected range.

# sort the annotation bed file into the same order as the reference genome (and alignment file)

bedtools sort **-**i **$DATA\_DIR/**annotation.bed **-**faidx ${DATA\_DIR}**/**reference**/**hg19\_chrom.bed **\**

**>** **$DATA\_DIR/**sorted\_annotation.bed

#call coverage with -sorted to reduce processing time

bedtools coverage **-**sorted -g ${DATA\_DIR}**/**reference**/**hg19\_chrom.bed **\**

-a **$DATA\_DIR/**sorted\_annotation.bed **\**

-b **$ALIGN\_DIR/**${FILE\_NAME}\_sorted\_filtered.bam **\**

**|** **sort** **-**k5**,**5n **>** **$STATS\_DIR/**DOC\_sorted.bed

# Generate DOC Summary Statistics

**awk** 'BEGIN{count=0; sum=0; sumsq=0; min=0; max=0}

{a[count++] = $5; sum+=$5; sumsq+=$5^2; if(count==1){min=$5; max=$5}; if($5<min){min=$5}; if($5>max){max=$5}}

END{ mean=sum/count; stdev=sqrt((sumsq/count)-(mean^2));

print "Mean: " mean; print "Stdev: " stdev;

print "Median: " a[int(count/2)]; print "Min: " min; print "Max: " max}

' **$STATS\_DIR/**DOC\_sorted.bed **|** **tee** **$STATS\_DIR/**DOC\_summmary.stats

Coverage: We assess the Depth of Coverage of the sequenced reads across the regions of interest to ensure that most of the regions are covered to a deep enough level that we can be confident in the accuracy of the base calls.

The coverage tool is very memory-intensive so we use the -sorted option to reduce the amount of memory used. The sorted option requires both the alignment bam and the annotation region bed to be in the same sort order; in this case the bam is sorted by the hg19 coordinate system (which puts chrX between chr7 and chr8) but the bed file is sorted lexographically. We call bedtools sort to re-sort the bed file into the same order as the bam file (hg19\_chrom.bed contains chromosome order information for the reference genome, downloaded from UCSC genome browser)

We pipe the output from the coverage tool to a sort command so the data is sorted by the coverage depth at each position. This enables us to easily find the median depth.

Now we run awk code to generate summary statistics:

1. create an array of depths at their position (to find Median)
2. sum all the depths (to calculate mean) and sum the square of all the depths (for standard deviation)
3. Print and save to file (using tee) the Mean, StDev, Median, Min and Max depths

### 2.4. Variant Calling (4pts)

* Call Variants using Freebayes restricting the analysis to the (2pt)

#decompress and index the reference genome

**zcat** **$DATA\_DIR/**reference**/**hg19.fa.gz **>** **$DATA\_DIR/**reference**/**hg19.fa

samtools faidx **$DATA\_DIR/**reference**/**hg19.fa

#freebayes variant caller

freebayes **--**bam **$ALIGN\_DIR/**${FILE\_NAME}\_sorted\_filtered.bam **\**

**--**fasta-reference **$DATA\_DIR/**reference**/**hg19.fa **\**

**--**vcf **$RESULTS\_DIR/**${FILE\_NAME}**.**vcf

#compress and index the vcf to reduce storage

bgzip **$RESULTS\_DIR/**${FILE\_NAME}**.**vcf

tabix -p vcf **$RESULTS\_DIR/**${FILE\_NAME}**.**vcf.gz

We call the freebayes tool to identify and call variants where the sample sequences differ from the reference genome.

#select only the variants that exist in the annotation file

bedtools intersect **-**header **-**wa -a **$RESULTS\_DIR/**${FILE\_NAME}\_filter.tmp.vcf **\**

-b **$DATA\_DIR/**annotation.bed **>** **$RESULTS\_DIR/**${FILE\_NAME}\_filtered.vcf

We later call bedtools intersect to keep only the variants in the vcf file that overlap regions specified in the annotation.bed file. The -header flag ensures we keep a header in the resulting output vcf. The -wa flag tells bedtools to retain the full vcf record for selected variants.

* Quality Filter Variants using your choice of filters (2pt)

We will apply the following suggested freebayes hard filter for human diploid sequencing and use vcffilter to apply it:

QUAL > 1: removes corrupt sites of exceptionally low quality

QUAL / AO > 10 : AO is the Alternative Allele Observations so this term indicates that we only keep reads where the average quality for each observation is over log(10)

SAF > 0 & SAR > 0 : ensure there are reads on both strands

vcffilter -f "QUAL > 1 & QUAL / AO > 10 & SAF > 0 & SAR > 0 & RPR > 1 & RPL > 1" **\**

**$RESULTS\_DIR/**${FILE\_NAME}**.**vcf.gz **>** **$RESULTS\_DIR/**${FILE\_NAME}\_filter.tmp.vcf

RPR > 1 & RPL > 1 : at least two reads “balanced” to each side of the site

### 2.5. Variant Annotation and Prioritization (10pts)

* Annotate variants using **ANNOVAR** (4pt) and **snpEFF** (4pt)
* Perform basic variant prioritization: filter to exonic variants not seen in dbSNP (2pts)

**Annovar**

Before running annovar we must install all the databases we want to use. In this case I have selected the snp130 database for annotating with dbSNP IDs because it is an order of magnitude smaller than the latest version of dbSNP. With more resources available it would be better to use avsnp150 database instead.

ANN\_DIR**=**"/home/ubuntu/annovar"

**$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar snp130 **$ANN\_DIR/**humandb**/**

**$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar refGene **$ANN\_DIR/**humandb**/**

#there is insufficient space on the drive to download and extract all the databases

**$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar knownGene **$ANN\_DIR/**humandb**/**

**$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar clinvar\_20180603 **$ANN\_DIR/**humandb**/**

**$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar exac03 **$ANN\_DIR/**humandb**/**

**#$ANN\_DIR/**annotate\_variation.pl **-**downdb **-**buildver hg19 **-**webfrom annovar dbnsfp31a\_interpro **$ANN\_DIR/**humandb**/**

I specified refGene to enable selecting for exonic variants as well as clinvar, knownGene and exac03 for additional annotation. Ideally the dbNSFP database would also be used to annotate for in silico predictions such as SIFT; unfortunately, there was insufficient storage to download this database as well as dbSNP.

**$ANN\_DIR/**convert2annovar.pl **-**format vcf4 $**RESULTS\_DIR/**${FILE\_NAME}\_filtered.vcf.gz **\**

**>** **$RESULTS\_DIR/**annovar**/**${FILE\_NAME}\_filtered.avinput

First, we must convert the vcf file of filtered variants into a format that Annovar can process.

The basic approach to running Annovar involves calling table\_annovar.pl.

We set the build version to the reference genome version, specify -remove to delete temporary files after completion, and -nastring to instruct annovar the string should be used for empty values. The -csvout flag tells Annovar to generate a comma separated values file; the default is tab-separated.

**$ANN\_DIR/**table\_annovar.pl **$RESULTS\_DIR/**${FILE\_NAME}\_filtered.avinput **\**

**$ANN\_DIR/**humandb**/** **-**buildver hg19 **\**

**-**out **$RESULTS\_DIR/**annovar**/**${FILE\_NAME}\_annovar **-**remove **\**

**-**protocol refGene**,**snp130**,**clinvar\_20180603**,**exac03**,**dbnsfp31a\_interpro **\**

**-**operation g**,**f**,**f**,**f**,**f **-**otherinfo **-**nastring **.** **-**csvout

We specify the databases we want to use for annotation in the –protocol option and the operation flag for each protocol.

To filter variants to exonic that are not present in dbSNP, we have to use the annotation\_variation command.

#filter to identify variants not in dbSNP

**$ANN\_DIR/**annotate\_variation.pl **-**filter **-**dbtype snp130 **-**buildver hg19 **\**

**-**outfile **$RESULTS\_DIR/**annovar**/**${FILE\_NAME}\_annovar.dbSNP **\**

**$RESULTS\_DIR/**annovar**/**${FILE\_NAME}\_filtered.avinput **$ANN\_DIR/**humandb**/**

#then filter to only keep those with exonic function (regionanno)

**$ANN\_DIR/**annotate\_variation.pl **-**geneanno **-**buildver hg19 **-**dbtype refGene **\**

**-**outfile **$RESULTS\_DIR/**annovar**/**${FILE\_NAME}\_annovar.RG **\**

**$RESULTS\_DIR/**annovar**/**${FILE\_NAME}\_annovar.dbSNP.hg19\_snp130\_filtered **$ANN\_DIR/**humandb**/**

#remove the first 3 columns that were added by annnotate\_variation -geneanno

#necessary to be able to run table\_annovar

**cut** -f 4**-** -d$'\t' **$RESULTS\_DIR/**annovar**/**${FILE\_NAME}\_annovar.RG.exonic\_variant\_function **\**

**>** **$RESULTS\_DIR/**annovar**/**${FILE\_NAME}\_annovar.RG.new\_evf

We must first run a Filter to separate variants that have a dbSNP id from those that don’t. Next we perform Gene-level Annotation to obtain a file that only contains variants that have an exonic function.

Finally we can run the table command on the resulting variants to add additional annotations to the remaining variants. Here we are only setting a sample of the potential annotations due to limited storage for downloading databases.

#finally add annotations for remaining variants

**$ANN\_DIR/**table\_annovar.pl **$RESULTS\_DIR/**annovar**/**${FILE\_NAME}\_annovar.RG.new\_evf **\**

**$ANN\_DIR/**humandb**/** **-**buildver hg19 **\**

**-**out **$RESULTS\_DIR/**annovar**/**${FILE\_NAME}\_annovar **-**remove **\**

**-**protocol refGene**,**knownGene**,**exac03**,**clinvar\_20180603 **\**

**-**operation g**,**g**,**f**,**f **-**otherinfo **-**nastring **.** **-**csvout

**snpEff**

First we must download the reference genome for snpEff and the dbSNP database. snpEff requires the dbSNP database to be compressed and indexed with bgzip but the database can only be downloaded in standard gzip format, so we have to decompress it then recompress with bgzip and index it.

EFF\_DIR**=**"/home/ubuntu/snpEff/"

java **-**jar **$EFF\_DIR/**snpeff.jar download hg19

# these commands should only be performed once as the file is very large

**cd** **$DATA\_DIR/**reference

**wget** https**://**ftp.ncbi.nih.gov**/**snp**/**organisms**/**human\_9606\_b151\_GRCh37p13**/**VCF**/**00**-**common\_all.vcf.gz

**gunzip** -c 00**-**common\_all.vcf.gz **|** bgzip -c **>** dbSNP-b151-small.vcf.bgz

tabix -p vcf **$DATA\_DIR/**reference**/**dbSNP-b151-small.vcf.bgz

Now we run SnpSift annotate to add dbSNP IDs to the variants. We use the -exists option to add a flag “DBSNP” to the info field for variants that have an ID in the dbSNP database. This can be used later for filtering.

##1) snpsift annotate to add dbsnp ids

java **-**jar **$EFF\_DIR/**SnpSift.jar annotate **-**id **-**exists "DBSNP" **$DATA\_DIR/**reference**/**dbSNP-b151-small.vcf.bgz **\**

**$RESULTS\_DIR/**${FILE\_NAME}\_filtered.vcf.gz **\**

**>** **$RESULTS\_DIR/**snpEff**/**${FILE\_NAME}\_snpEff1\_dbsnpids.vcf

##2) snpeff to annotate with functions against hg19/grch37 genome

java **-**Xmx8g **-**jar **$EFF\_DIR/**snpEff.jar hg19 **$RESULTS\_DIR/**snpEff**/**${FILE\_NAME}\_snpEff1\_dbsnpids.vcf **\**

**>** **$RESULTS\_DIR/**snpEff**/**${FILE\_NAME}\_snpEff2\_annotated.vcf

Next we use the snpEff command to annotate the variants from the previous step. We use the -Xmx8g parameter to increase the memory available to the java virtual machine to 8G.

##3) snpsift filter to remove non-exonic and any that exist in dbsnp

java **-**jar **$EFF\_DIR/**SnpSift.jar filter **\**

"ANN[\*].EFFECT has 'missense\_variant' & !exists DBSNP" **\**

**$RESULTS\_DIR/**snpEff**/**${FILE\_NAME}\_snpEff2\_annotated.vcf **\**

**>** **$RESULTS\_DIR/**snpEff**/**${FILE\_NAME}\_snpEff3\_prioritised.vcf

Finally we use SnpSift filter to select only the variants where one of the ANN EFFECT fields (in the INFO column) has ‘missense\_variant’ (this is our criteria for exonic), and where it does not have a DBSNP flag in the INFO field (that we set earlier). This way we are able to select exonic variants that are not present in dbSNP.

**2.6 Using an alternative tool (5pts)**

* Modify the pipeline by replacing either the aligner or the variant caller with an alternative tool. Share a new bash script with the modified pipeline with the examiners (3pt)

Alternative script can be found on github here (and on Canvas):

[https://github.com/m2109260/bioinformatics\_course/blob/main/scripts/m2109260\_AdvBio\_Alternative.sh](https://github.com/m2109260/bioinformatics_course/blob/adb13991e1bee43292213d252097a0aabd1f352e/scripts/m2109260_AdvBio_Alternative.sh)

* Provide below the new commands used to run the alternative tool and comment on your choice of options and how and if using this tool would affect the results (2pt).

I have replaced the Freebayes variant called with bcftools mpileup and call commands. The bcftools approach is a popular method of variant calling and will likely generate some different variant calls to the Freebayes approach.

First, we generate appropriate input pileup data then we pipe the output, in uncompressed binary format, to the call command to actually call the variants.

* The -Ou parameter in mpileup outputs uncompressed binary format - this is the fastest approach when piping to the call command.
* The -mv parameter of the call command specifies both multiallelic (m) AND variant-only calling (v).
* The -Ov flag causes the output to be in vcf format as required for vcffilter.

bcftools mpileup **-**Ou -f **$DATA\_DIR/**reference**/**hg19.fa.bgz **\**

**$ALIGN\_DIR/**${FILE\_NAME}\_sorted\_filtered.bam **\**

**|** bcftools call **-**mv **-**Ov -o **$RESULTS\_DIR/**${FILE\_NAME}**.**vcf

bgzip **$RESULTS\_DIR/**${FILE\_NAME}**.**vcf

tabix -p vcf **$RESULTS\_DIR/**${FILE\_NAME}**.**vcf

We compress and index the vcf with bgzip and tabix to reduce the file size.

The downstream impact of changing the variant caller is that the fields in the INFO field of the vcf are different to those created by Freebayes, so it is necessary to update the vcffilter command to account for this.

The AC (Alternative Allele Count) field in bcftools is the equivalent AO (Alternative Allele Observations) field in Freebayes so this is a simple replacement.

There are no direct replacements for the other Freebayes fields used in the filter so we have to select different fields to check for Strand Bias (FORMAT/SP) and Position Bias (RPB). Significant bias occurs at values less than 0.05 so for Read Position Bias (pvalue) we want to discard any such reads. The SP field is in Phred33 format so we have to take the -10log of the pvalue – a commonly used filter for Strand Bias is to exclude anything with SP greater than 60 so we apply this filter here.

vcffilter -f "QUAL > 1 & QUAL / AC > 10 & RPB < 0.05" **\**

-g "SP > 60" **$RESULTS\_DIR/**${FILE\_NAME}**.**vcf.gz **\**

**>** **$RESULTS\_DIR/**${FILE\_NAME}\_filter.tmp.vcf

## R/RStudio assessment (45pts – 45% of final mark)

The Rmd and html for this part of the assessment can be found on github here:

<https://github.com/m2109260/bioinformatics_course/tree/main/RScript>

This R assignment is split into 3 parts. The first part is about the general use of R/Rstudio, the second part about RNAseq and the third about ChIP-Seq. In these parts you will be asked to perform a number of tasks in R/RStudio and report them in your own markdown document.

Initial task: Create a new markdown document in *RStudio*, set the title to "Advanced Bioinformatics 2023 assessment", and insert an "author:" tag below the title, followed by your student id. Share your markdown document and html via your github account.

In the following, for each task, create a new heading called "Task X" for task X, and insert a new R code chunk that holds any code required. Make sure to evaluate the expression before saving to include the output in the html file. If you have multiple lines that produce outputs, you can split them into separate code chunks for increase clarity (but it is not necessary to pass the assessment). Please also explain your steps.

**General R/Rstudio assessment (33 pts)**

3.1. Using the *sum*() function and : operator, write an expression in the code snippet to evaluate the sum of all integers between 5 and 55. (4pt)

3.2. Write a function called *sumfun* with one input parameter, called *n*, that calculates the sum of all integers between 5 and *n*. Use the function to do the calculation for *n* = 10, *n* = 20, and *n* = 100 and present the results. (4pt)

3.3. The famous Fibonacci series is calculated as the sum of the two preceding members of the sequence, where the first two steps in the sequence are 1, 1. Write an R script using a for loop to calculate and print out the first 12 entries of the Fibonacci series. (4pt)

3.4. With the *mtcars* dataset bundled with R, use *ggplot* to generate a box of miles per gallon (in the variable *mpg*) as a function of the number of gears (in the variable *gear*). Use the fill aesthetic to colour bars by number of gears. (4pt)

3.5. Using the *cars* dataset and the function *lm*, fit a linear relationship between *speed* and breaking distance in the variable *distance*. What are the fitted slope and intercept of the line, and their standard errors? What are the units used for the variables in the dataset? (4pt)

3.6. Use *ggplot* to plot the data points from Task 6 and the linear fit. (4pt)

3.7. Again using the cars dataset, now use linear regression (*lm*) to estimate the average reaction time for the driver to start breaking (in seconds). To simplify matters you may assume that once breaking commences, breaking distance is proportional to the square of the speed. Explain the steps in your analysis. Do you get reasonable results? Finally, use *ggplot* to plot the data points and the fitted relationship. (9pt)

**RNA-seq assessment (8 pts)**

In this part, we will analyse the RNASeq data used in the RNA-seq tutorial to:

1. create a DESeq2 object,
2. normalize RNA-seq data with DESeq2,
3. perform differential Expression analysis with DESeq2,
4. visualize RNA-seq data using SDM and PCA methods.

You may access to the data that we used during tutorial from [here](https://emckclac-my.sharepoint.com/:u:/g/personal/k2037526_kcl_ac_uk/EYabNsg1JVZHrYzuMKqlHFEB_9WI3aHeNLzvk7eGqX-0yQ?e=GOB5VC).

3.8. Read in count data and sample description. **(1pts)**

* LMS\_RNAseq\_short-master-2023-final/course/exercises/data/exercise1\_counts.csv
* LMS\_RNAseq\_short-master-2023-final/course/exercises/data /exercise1\_sample\_description.info

3.9. Create col\_data and check dimensions. **(1 pts)**

3.10 Construct DESeqDataSet object using count data and sample description. **(1 pts)**

3.11. Perform rlog and VST transformation on the data. **(1 pts)**

3.12. Draw a heatmap of count matrix based on the top 40 highly expressed genes using rlog and VST data. **(1 pts)**

3.13. Generate a SDM to see the clustering of count data. **(1 pts)**

3.14. Perform the Principal Component Analysis using rlog method and find out the % significance values of first two principal components. **(1 pts)**

3.15. Repeat the PCA, this time using VST method and compare the plots with the ones obtained using rlog method. **(1 pts)**

**ChIP-seq assessment (4 pts)**

In this assessment, we will read in two replicate sets of CHIP-seq peaks from the Myc Encode dataset and extract sequences underneath subsets of peaks. We will write these sequences out to a FASTA file and upload the FASTA file to Meme-ChIP to detect motifs underneath of these peaks.

You may access to the data that we used during tutorial from [here](https://emckclac-my.sharepoint.com/:u:/g/personal/k2037526_kcl_ac_uk/EfGg2LEp4xhIrWQMO6wBvbgB60MJOE3322BORVR1IANLkw?e=WHpZTd).

3.16. Read in the two Myc Mel peakset replicates and create the common peakset as we did for our previous exercise. **(1 pts)** The files you need are here:

* + LMS\_ChIPseq\_short-master-2023-final/course/data /MacsPeaks/mycmelrep1\_peaks.xls
  + LMS\_ChIPseq\_short-master-2023-final/course/data /MacsPeaks/mycmelrep2\_peaks.xls

3.17. Now we can rank them by their fold enrichment, select the top 500 peaks and resize these peaks to 200bp around centre. **(1 pts)**

3.18. Extract the sequences underneath the file and write them to FASTA file in you working directory. Inspect the file in notepad. **(1 pts)**

3.19. Upload the sequences to Meme-ChIP and report the results when complete. **(1 pts)**