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

**Student ID: 23120071**

**Github: https://github.com/TatOMICS/Advance-Bioinformatics-Assessment-23120071-**

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

**1.1 What does ./../.. stand for ?**  
A. Current directory  
B. Up one directory  
C. Up two directories  
D. None of Above

**Answer**: Answer C. Moving Up 2 Directories

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

**Answer:** . When ‘cd /’ is executed, it takes us to the root directory (the highest directory in the hierarchy) ‘cd’ stands for ‘Change Directory’ and ‘/’ represents the root directory.

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

**Answer:** The function ‘man’ (stand for ‘manual’) needs to be used. This will provide all the help available inside the documentation. **Example:** man cp

1.4 What does the command pwd do?

**Answer:** ‘pwd’ stands for ‘print working directory’ , when executed in prints/shows the current working directory in the terminal.

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)

**Command:** ‘ ls -l ’

**Answer:**  The command will list details about the files/directories in the current directory (permissions, owner, size, etc)

**Command: ‘** ls -l /path/directory ’

**Answer:**  By adding a path file we obtain the detailed information within that specific directory.

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

**Command:** find . -name "\*.txt" -exec head -n 15 {} \;

**Answer:** In order to print it from ALL files not just the current directory.

**Command:** head -n 15 \*.txt

**Answer:** Prints the matches only in the CURRENT directory

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

**Command:** mv new old

**Answer:** ‘mv’ stands for move and it can be used to rename file. The new name goes after the ‘mv’ command, and the new name goes after.

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

**Command:** less myfile.txt

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

**Command:** mkdir flower

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

**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)

**Command:** ls -a

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

**Command:** cd ..

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

**Command:** mkdir ~/my\_subdirectory

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

**Command**: head assesmentfile.txt

**Answer:** This shows by default the first 10 lines of a given file.

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

**Command:** tail assesmentfile.txt

**Answer:** Shows by default the last 10 lines.

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

**Command:** cut -f 6 assesmentfile.txt

**Answer:** The function ‘cut’ extracts the column indicated with ‘-f ‘ in this case column 6 form file assesmentfile.txt

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

**Command:** mkdir Project\_backup; cp -r Project/.\* Project/\* Project\_backup

**Answer:** We first create the ‘Project Backup’ directory and then copy everything (Both visible and hidden files) from ‘Project’ to “Project\_backup’. Note: the ‘.’ Selects the hidden files.

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)

**Command:** grep 'Hypertension$' diseases.txt

**Answer:** grep searches for Hypertension inside the diseases.txt. The symbol ‘$’ tells the system the string is at the end of the line

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

**Command:** ls -a ~

**Answer:** ‘ls’ list the directory contents , ‘-a’ shows all files Including hidden files inside the ‘~‘ home directory. To see only Hidden files we can use : ls -a ~ | grep '^\.' , that filters only the files that are hidden.

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

**Command**: nohup python my\_script.py &

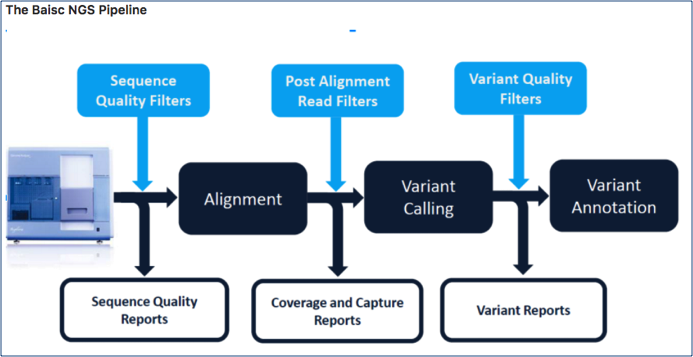
**Answer:** ‘nohup’ (stands for not hang up) will execute in this example a python script even in the terminal closes. ‘&’ ensures the command is run in the background.

## 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>

HINT: Please note that the sequencing data have an “odd” extension. You might consider renaming the files.

**In the following questions you will be asked to provide the command lines used to perform the steps of the pipeline and to comment and explain the choice of tools and all options. Please do not forget the latter as copying and pasting the command lines from the bash pipeline will not be sufficient to pass. You will need to demonstrate a clear understanding of your choices. Feel free to provide examples (even graphical/screenshots) if helpful.**

## 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)

## 1.1. Install Anaconda

cd ~/

wget https://repo.anaconda.com/archive/Anaconda3-2022.10-Linux-x86\_64.sh

chmod +x ./Anaconda3-2022.10-Linux-x86\_64.sh

bash ./Anaconda3-2022.10-Linux-x86\_64.sh

source ~/.bashrc

## 1.2 Install required packages with Anaconda

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

## I know it’s a duplication, the previous one did not worked.

conda install -c bioconda trimmomatic

## 1.3. Install remaining needed software.

sudo apt install bwa

## 1.4. Initial directory structure

cd ~

mkdir ngs\_assesment\_FINAL

mkdir ngs\_assesment\_FINAL/dnaseq

cd ngs\_assesment\_FINAL/dnaseq

mkdir data meta results logs

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

# 2. Download the files.

## 2.1 Create the untrimmed and trimmed files

cd ~/ngs\_assesment\_FINAL/dnaseq/data

mkdir untrimmed\_fastq

mkdir trimmed\_fastq

## 2.2 Download the files FastQ R1 and R2 and bed file

## Files are inside the 'data' fodler.

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

## 2.3 Move the raw fastq files to the'untrimmed\_fastq' directory.

## Leave the bed file inside the 'data' directory

mv NGS0001.R1.fastq.qz NGS0001.R2.fastq.qz ~/ngs\_assesment\_FINAL/dnaseq/data/untrimmed\_fastq

## 2.4 Convert .qz FASTQ files to .gz files

cd ~/ngs\_assesment\_FINAL/dnaseq/data/untrimmed\_fastq

mv NGS0001.R1.fastq.qz NGS0001.R1.fastq.gz

mv NGS0001.R2.fastq.qz NGS0001.R2.fastq.gz

## 2.5 Download reference genome FASTA file into a reference file (data > reference)

## Create the 'Reference' folder first.

mkdir -p ~/ngs\_assesment\_FINAL/dnaseq/data/Reference

cd ~/ngs\_assesment\_FINAL/dnaseq/data/Reference

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

## 2.6 Generate index of reference genome fasta file for alignment.

## This step takes ages 3959.920 seC.

bwa index hg19.fa.gz

## 2.7 Download & setup Annovar database for future variant anotation.

## https://www.openbioinformatics.org/annovar/annovar\_download\_form.php

## Once logged a link is provided.

## http://www.openbioinformatics.org/annovar/download/0wgxR2rIVP/annovar.latest.tar.gz

cd ~/ngs\_assesment\_FINAL

wget http://www.openbioinformatics.org/annovar/download/0wgxR2rIVP/annovar.latest.tar.gz

## Unpack and set up the file.

tar -zxvf annovar.latest.tar.gz

## 2.8 Download Annovar db

cd annovar

./annotate\_variation.pl -buildver hg19 -downdb -webfrom annovar knownGene humandb/

./annotate\_variation.pl -buildver hg19 -downdb -webfrom annovar refGene humandb/

./annotate\_variation.pl -buildver hg19 -downdb -webfrom annovar ensGene humandb/

./annotate\_variation.pl -buildver hg19 -downdb -webfrom annovar clinvar\_20180603 humandb/

./annotate\_variation.pl -buildver hg19 -downdb -webfrom annovar exac03 humandb/

./annotate\_variation.pl -buildver hg19 -downdb -webfrom annovar dbnsfp31a\_interpro humandb/

***Implement and run the following NGS Pipeline (please provide the command lines to run the following steps of your pipeline and comment/explain the choice of options):***

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

1. Perform quality assessment and trimming (2pt)

# 3 FastQC quality assessment on raw data

cd ~/ngs\_assesment\_FINAL/dnaseq/data/untrimmed\_fastq

## Note the extra parameter we specified for 4 threads

fastqc -t 4 \*.fastq.gz

## 3.1 Move FastQC results to a new directory

mkdir ~/ngs\_assesment\_FINAL/dnaseq/results/fastqc\_untrimmed\_reads

mv \*fastqc\* ~/ngs\_assesment\_FINAL/dnaseq/results/fastqc\_untrimmed\_reads

## 3.2 Download R 1 and R2 html using FileZilla !!!

# 4 Perform Trimmomatic (trimming) on raw sequencing data

## Trimmomatic trims adapters and drops reads below 50 bp in length, removes Illumina adapters and trims end of reads bases if below a threshold quality (25).

cd ~/ngs\_assesment\_FINAL/dnaseq/data/untrimmed\_fastq

trimmomatic PE \

-threads 4 \

-phred33 \

~/ngs\_assesment\_FINAL/dnaseq/data/untrimmed\_fastq/NGS0001.R1.fastq.gz ~/ngs\_assesment\_FINAL/dnaseq/data/untrimmed\_fastq/NGS0001.R2.fastq.gz \

-baseout ~/ngs\_assesment\_FINAL/dnaseq/data/trimmed\_fastq/NGS0001\_chr22m\_trimmed\_R \

ILLUMINACLIP:/home/ubuntu/anaconda3/pkgs/trimmomatic-0.39-hdfd78af\_2/share/trimmomatic-0.39-2/adapters/NexteraPE-PE.fa:2:30:10 \

TRAILING:25 MINLEN:50

|  |  |
| --- | --- |
|  |  |

Figure 1. This image is a screenshot of the Basic Statistics and a pre-trimming quality control report for Next-Generation Sequencing data, displaying basic statistics and a per-base sequence quality box plot.

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

# 5. Trimmed sequencing data quality assesment using FastQC.

cd ~/ngs\_assesment\_FINAL/dnaseq/data/trimmed\_fastq

fastqc -t 4 NGS0001\_chr22m\_trimmed\_R\_1P NGS0001\_chr22m\_trimmed\_R\_2P

## 5.1 Move FastQC to the trimmed results directory

mkdir ~/ngs\_assesment\_FINAL/dnaseq/results/fastqc\_trimmed\_reads

mv \*fastqc\* ~/ngs\_assesment\_FINAL/dnaseq/results/fastqc\_trimmed\_reads

|  |  |
| --- | --- |
|  |  |

Figure 2 This image is a screenshot of the Basic Statistics and a post-trimming quality control report for Next-Generation Sequencing data, displaying basic statistics and a per-base sequence quality box plot. he sequence quality across reads has been enhanced by the trimming process, evidenced by the more uniform high-quality scores throughout the entire read length in the post-trimming report, while the total number of sequences and bases remained largely consistent, indicating effective trimming of low-quality ends without significant loss of data. The improvement is clearly visible in the right hand side of the figure. 2 when compared with figure 1.

### 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)

# 6 Alignment of reads to reference genome

mkdir ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data

# 6.1 alignment of trimmed reads to hg19 reference genome, using BWA-MEM algorithm.

# 6.2 read group info obtained from the raw read FASTQ files.

bwa mem -t 4 -v 1 \

-R '@RG\tID:HWI-D0011.50.H7AP8ADXX.1.WES01\tSM:WES01\tPL:ILLUMINA\tLB:nextera-wes01-blood\tDT:2017-02-23\tPU:HWI-D00119' \

-I 250,50 \

~/ngs\_assesment\_FINAL/dnaseq/data/Reference/hg19.fa.gz \

~/ngs\_assesment\_FINAL/dnaseq/data/trimmed\_fastq/NGS0001\_chr22m\_trimmed\_R\_1P \

~/ngs\_assesment\_FINAL/dnaseq/data/trimmed\_fastq/NGS0001\_chr22m\_trimmed\_R\_2P \

> ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001.sam

# It’s taking ages !!!!

# The file is now created inside /home/ubuntu/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001.sam

# 7. Convert, process and index SAM file

cd ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data

# 7.1. Convert files from Sam to BAM format

samtools view -h -b NGS0001.sam > NGS0001.bam

# 7.2. Sort bam file

samtools sort NGS0001.bam > NGS0001\_sorted.bam

# 7.3. Generate the index

samtools index NGS0001\_sorted.bam

* Perform duplicate marking (2pts)

# 7.4 Mark duplicate reads

# Identification and marking of the duplicated reads inside the BAM file

cd ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data

picard MarkDuplicates I=NGS0001\_sorted.bam O=NGS0001\_sorted\_marked.bam M=marked\_dup\_metrics.txt

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

# 7.5 Filter BAM based on mapping quality and bitwise flags using samtools .

samtools view -F 1796 -q 20 -o NGS0001\_sorted\_filtered.bam NGS0001\_sorted\_marked.bam

# 7.6 Indexing of the files

samtools index NGS0001\_sorted\_filtered.bam

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

#.7.3 Generate alignment statistic

cd ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data

# 7.4 make a new directory for these stats

mkdir aligment\_stats

# 7.5 Generate flagstats

samtools flagstat NGS0001\_sorted\_filtered.bam > flagstats\_output.txt

mv flagstats\_output.txt ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/aligment\_stats/

# 7.6 view the BAM file in the command line

cd ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data

samtools view -h NGS0001\_sorted\_filtered.bam | less # Perhaps i need to get rid of this for the automation

# 7.7 Generate alignment statistics per chromosome with idxstats

samtools idxstats NGS0001\_sorted\_filtered.bam > idxstats\_output.txt

mv idxstats\_output.txt ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/aligment\_stats/

# 7.8 Determine the distribution of insert sizes between read pairs with Picard tools

java -jar /home/ubuntu/anaconda3/pkgs/picard-2.18.29-0/share/picard-2.18.29-0/picard.jar CollectInsertSizeMetrics \

I=/home/ubuntu/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001\_sorted\_filtered.bam \

O=insert\_size\_metrics.txt \

H=insert\_size\_histogram.pdf

## Download from FileZilla (inside aligened\_data)

A red line graph with numbers

Description automatically generated

Graph 1. The insert size histogram for the sequencing library in NGS0001\_sorted\_filtered.bam displays a predominant peak, indicative of a uniform insert size distribution with most insert sizes clustering around the peak value. The accompanying cumulative distribution curve confirms that the majority of inserts fall within a narrow size range, highlighting the consistent and specific targeting of DNA fragment sizes during library preparation, with the 'FR' label suggesting the standard forward-reverse orientation of paired-end reads.

### 2.4. Variant Calling (4pts)

* Call Variants using Freebayes restricting the analysis to the regions in the bed file provided (2pt)

# 8. Variant calling with Freebayes

cd ~/ngs\_assesment\_FINAL/dnaseq/data

zcat ~/ngs\_assesment\_FINAL/dnaseq/data/Reference/hg19.fa.gz > ~/ngs\_assesment\_FINAL/dnaseq/data/Reference/hg19.fa

samtools faidx ~/ngs\_assesment\_FINAL/dnaseq/data/Reference/hg19.fa

# 8.1 FreeBayes to report variants in the sequencing data compared to the reference allele freebayes.

freebayes \

--bam ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001\_sorted\_filtered.bam \

--fasta-reference ~/ngs\_assesment\_FINAL/dnaseq/data/Reference/hg19.fa \

--vcf ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_chr22m.vcf

# Compression of the VCF file

bgzip ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_chr22m.vcf

# Indexing of the vcf file

tabix -p vcf ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_chr22m.vcf.gz

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

# 9.Filtering the VCF

# Remove bad quality variant calls from the VCF file.

# "QUAL > 1" removes horrible sites, "QUAL / AO > 10" additional contribution of each obs should be

# 10 log units (~ Q10 per read), "SAF > 0 & SAR > 0" reads on both strands, "RPR > 1 & RPL > 1" at

# least two reads “balanced” to each side of the site.

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

~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_chr22m.vcf.gz > \

~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_chr22m\_filtered.vcf.gz

# 9.1 use bedtools to filter VCF for regions present in the annotation bed file

bedtools intersect -header -wa -a ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_chr22m\_filtered.vcf.gz \

-b ~/ngs\_assesment\_FINAL/dnaseq/data/annotation.bed \

> ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_filtered\_in\_bedfile.vcf

# 9.2 compress filtered VCF (Not sure if required)

bgzip -f ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_filtered\_in\_bedfile.vcf

# 9.3 Indexing of the vcf.

tabix -p vcf ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_filtered\_in\_bedfile.vcf.gz

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

* Annotate variants using **ANNOVAR** (4pt) and **snpEFF** (4pt)

# 10. Variant Annotation with snpEff

# Annotate the variants using snpEff for functional effects prediction

java -Xmx4g -jar snpEff.jar hg19 \

~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_filtered\_in\_bedfile.vcf.gz \

> ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_filtered\_in\_bedfile\_snpEff.vcf

# 10.1 Compress and index the snpEff annotated VCF

bgzip -f ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_filtered\_in\_bedfile\_snpEff.vcf

tabix -p vcf ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_filtered\_in\_bedfile\_snpEff.vcf.gz

# 10.2 Convert VCF to ANNOVAR input (if still needed after snpEff annotation)

~/ngs\_assesment\_FINAL/annovar/convert2annovar.pl \

-format vcf4 \

~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_filtered\_in\_bedfile\_snpEff.vcf.gz \

> ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_chr22m\_filtered\_chr22.avinput

# 10.3 Annovar for annotation of the variants with database frequencies/functional consequences including snp138

~/ngs\_assesment\_FINAL/annovar/table\_annovar.pl ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_chr22m\_filtered\_chr22.avinput \

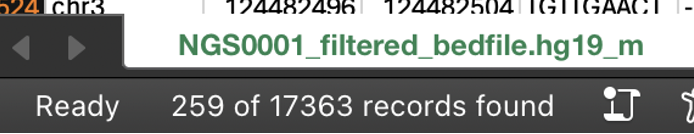
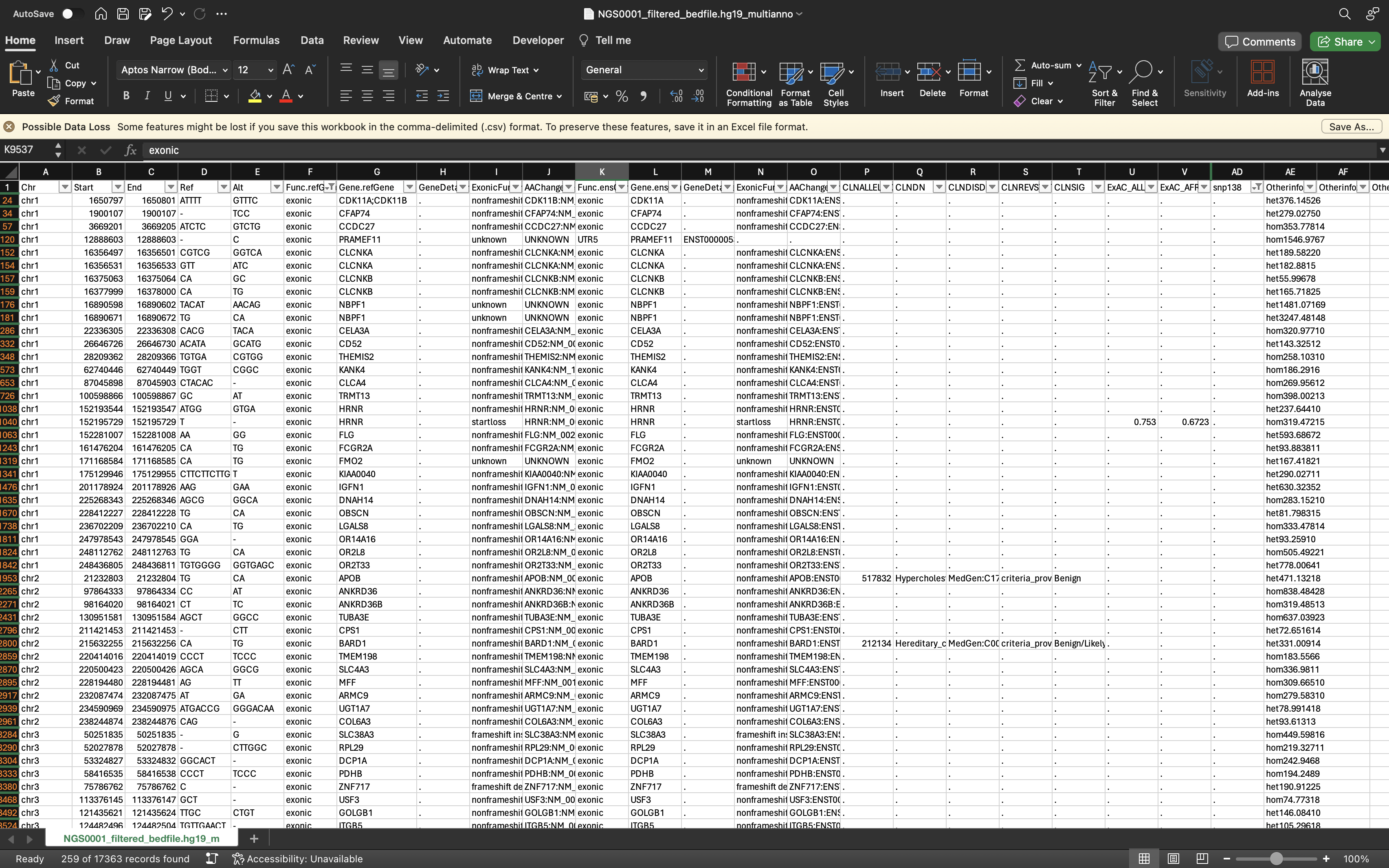
/home/ubuntu/ngs\_assesment\_FINAL/annovar/humandb/ -buildver hg19 \

-out ~/ngs\_assesment\_FINAL/dnaseq/results/NGS0001\_filtered\_bedfile -remove \

-protocol refGene,ensGene,clinvar\_20180603,exac03,dbnsfp31a\_interpro,snp138 -operation g,g,f,f,f,f \

-otherinfo -nastring . -csvout

* Perform basic variant prioritization: filter to exonic variants not seen x (2pts)



**Total: 259 prioritised variants.**

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

* Modify the pipeline by replacing either the aligner or the variant caller with an alternative tool, while leaving the rest of the pipeline unchanged. Share a new bash script with the modified pipeline with the examiners by uploading it on Canvas/KEATS with your assignment or via github (3pt)

# 6 Alignment of reads to reference genome

mkdir ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data

# 6.1 Alignment of trimmed reads to hg19 reference genome, using Bowtie 2

cd ~/ngs\_assesment\_FINAL/dnaseq/data/trimmed\_fastq

# Install Bowtie2

sudo apt update

sudo apt install bowtie2

# Index the reference genome

bowtie2-build ~/ngs\_assesment\_FINAL/dnaseq/data/Reference/hg19.fa.gz \ ~/ngs\_assesment\_FINAL/dnaseq/data/Reference/hg19\_index

# Perform Alignment with Bowtie2

bowtie2 -p 4 -x ~/ngs\_assesment\_FINAL/dnaseq/data/Reference/hg19\_index \

-1 NGS0001\_chr22m\_trimmed\_R\_1P \

-2 NGS0001\_chr22m\_trimmed\_R\_2P \

-S ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001.sam

# Convert SAM to BAM

samtools view -b -o ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001.bam ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001.sam

# Sort and Index BAM File

samtools sort -o ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001\_sorted.bam ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001.bam

samtools index ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001\_sorted.bam

# Remove unneeded data to save disk space

rm ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001.sam

rm ~/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001.bam

# The file is now created inside /home/ubuntu/ngs\_assesment\_FINAL/dnaseq/data/aligned\_data/NGS0001.sam

* 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).

Bowtie2 is a versatile and widely used bioinformatics tool designed for the alignment of sequencing reads to a reference genome. It utilizes an efficient algorithm to map millions of short DNA sequences (reads) generated by high-throughput sequencing technologies like Illumina sequencing. The primary purpose of Bowtie2 is to accurately align these reads to a reference genome, facilitating various downstream analyses such as variant calling, gene expression analysis, and genome assembly.

The choice of Bowtie2 for alignment in this pipeline is strategic for several reasons. Firstly, Bowtie2 is renowned for its speed and accuracy, making it well-suited for processing large-scale sequencing datasets efficiently. This characteristic is particularly crucial in NGS data analysis pipelines where timely processing is essential. Additionally, Bowtie2 is optimized for paired-end sequencing data, allowing it to handle the alignment of read pairs, which is common in many NGS experiments.

Furthermore, Bowtie2 offers flexibility in parameter tuning, enabling users to customize the alignment process according to the specific requirements of their analysis. Parameters such as the number of threads (`-p`) and alignment mode can be adjusted to optimize performance and sensitivity based on the computational resources available and the nature of the sequencing data being analyzed.

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

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) **✓**

result <- sum(5:55)

print(result)

## [1] 1530

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) **✓**

sumfun <- **function**(n) {

*# Calculate the sum of integers between 5 and n*

sum <- 0

**for** (i **in** 5:n) {

sum <- sum + i

}

**return**(sum)

}

*# Calculate the sum for n = 10, 20, and 100*

result\_n10 <- sumfun(10)

result\_n20 <- sumfun(20)

result\_n100 <- sumfun(100)

*# Present the results*

cat("Sum of integers between 5 and 10:", result\_n10, "\n")

cat("Sum of integers between 5 and 20:", result\_n20, "\n")

cat("Sum of integers between 5 and 100:", result\_n100, "\n")

## Sum of integers between 5 and 10: 45

## Sum of integers between 5 and 20: 200

## Sum of integers between 5 and 100: 5040

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) **✓**

*# Initialize the first two Fibonacci numbers*

fibonacci <- c(1, 1)

*# Loop to calculate the next 10 Fibonacci numbers*

**for** (i **in** 3:12) {

next\_fib <- fibonacci[i - 1] + fibonacci[i - 2]

fibonacci <- c(fibonacci, next\_fib)

}

*# Print out the Fibonacci series*

cat("First 12 entries of the Fibonacci series:\n")

**for** (i **in** 1:12) {

cat(fibonacci[i], " ")

}

## First 12 entries of the Fibonacci series:

## 1 1 2 3 5 8 13 21 34 55 89 144

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) **✓**

*# Load the ggplot2 library*

**library**(ggplot2)

*# Load the mtcars dataset*

data(mtcars)

*# Create the boxplot using ggplot*

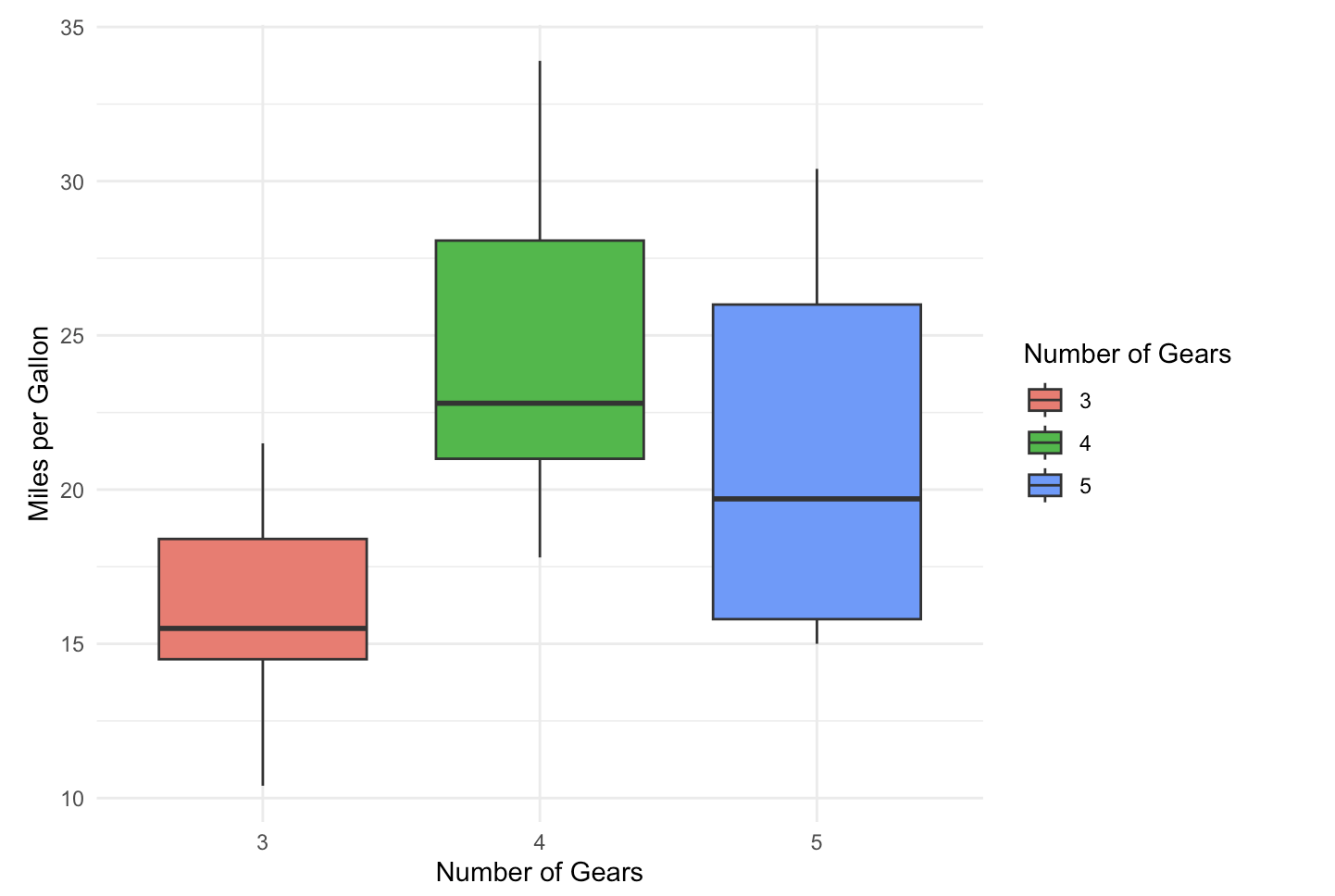
ggplot(mtcars, aes(x = as.factor(gear), y = mpg, fill = as.factor(gear))) +

geom\_boxplot() +

labs(x = "Number of Gears", y = "Miles per Gallon") +

scale\_fill\_discrete(name = "Number of Gears") +

theme\_minimal()



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) **✓**

*# Load the cars dataset*

data(cars)

*# Fit a linear model*

model <- lm(dist ~ speed, data = cars)

*# Retrieve slope, intercept, and their standard errors*

slope <- coef(model)[2]

intercept <- coef(model)[1]

slope\_se <- summary(model)$coefficients[2, 2]

intercept\_se <- summary(model)$coefficients[1, 2]

*# Print the results*

cat("Fitted slope:", slope, "\n")

## Fitted slope: 3.932409

cat("Slope standard error:", slope\_se, "\n")

## Slope standard error: 0.4155128

cat("Fitted intercept:", intercept, "\n")

## Fitted intercept: -17.57909

cat("Intercept standard error:", intercept\_se, "\n")

## Intercept standard error: 6.75844

cat("Units for speed: miles per hour (mph)\n")

## Units for speed: miles per hour (mph)

cat("Units for distance: feet (ft)\n")

## Units for distance: feet (ft)

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

*# Load the ggplot2 library*

**library**(ggplot2)

*# Fit a linear model*

model <- lm(dist ~ speed, data = cars)

*# Create a scatter plot with ggplot*

p <- ggplot(cars, aes(x = speed, y = dist)) +

geom\_point() + *# Add points for the data*

geom\_smooth(method = "lm", se = FALSE) + *# Add a linear fit line*

labs(x = "Speed (mph)", y = "Distance (ft)", title = "Linear Fit of Speed and Distance")

*# Print the plot*

print(p)

## `geom\_smooth()` using formula = 'y ~ x'

A graph with a line and dots

Description automatically generated

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) **✓**

*# Add the square of the speed to the dataset (breaking distance is proportional to the square of the speed hence we can use linear regresion.*

cars <- mutate(cars, speed\_squared = speed^2)

*# Fit the linear model*

model <- lm(dist ~ speed\_squared, data = cars)

*# Summarize the model to get coefficients and standard errors*

model\_summary <- summary(model)

*# The ntercept is the distance covered during the driver's reaction time,*

*# we calculate the average reaction time by dividing it by the average speed..*

average\_speed <- mean(cars$speed)

intercept <- model\_summary$coefficients["(Intercept)", "Estimate"]

intercept\_se <- model\_summary$coefficients["(Intercept)", "Std. Error"]

*# Convert intercept to time by dividing by the average speed*

*# Convert speed from mph to fps (feet per second) by multiplying by 1.467 (since 1 mph = 1.467 fps)*

estimated\_reaction\_time <- intercept / (average\_speed \* 1.467)

standard\_error\_reaction\_time <- intercept\_se / (average\_speed \* 1.467)

*# Output the estimated reaction time and its standard error*

print(paste("Estimated average reaction time (seconds):", estimated\_reaction\_time))

## [1] "Estimated average reaction time (seconds): 0.39217984174362"

print(paste("Standard error of estimated average reaction time (seconds):", standard\_error\_reaction\_time))

## [1] "Standard error of estimated average reaction time (seconds): 0.180876642762238"

*# Plot the data points and the fitted relationship*

**library**(ggplot2)

*# Once we have a fitted model, we use it to plot*

ggplot(cars, aes(x = speed, y = dist)) +

geom\_point(color = "red") + *# This plots the data points in red*

geom\_smooth(method = "lm", formula = y ~ poly(x, 2), se = TRUE, color = "blue") + *# This adds a fitted quadratic line with a confidence interval*

labs(x = "Speed (mph)", y = "Braking Distance (feet)",

title = "Time Taken to Brake") +

theme\_minimal() *# This gives a minimalistic theme similar*

A graph with red dots and blue line

Description automatically generated

**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
* *# Set the file paths*
* count\_data\_path <- "/Users/estebantato/Desktop/LMS\_RNAseq\_short-master-2023-final/course/exercises/data/exercise1\_counts.csv"
* sample\_description\_path <- "/Users/estebantato/Desktop/LMS\_RNAseq\_short-master-2023-final/course/exercises/data/exercise1\_sample\_description.info"
* *# Read in count data*
* count\_data <- read.csv(count\_data\_path, row.names = 1, check.names = FALSE)
* *# Read in sample description*
* sample\_description <- read.csv(sample\_description\_path, header = TRUE)
* *# View the first few lines of the data*
* head(count\_data)

## control\_FFa1.bam control\_FFa2.bam control\_FFa3.bam mutant\_KOa1.bam

## 497097 16 16 0 21

## 100503874 20 0 0 64

## 100038431 0 0 2 0

## 19888 11 0 10 113

## 20671 14 16 0 40

## 27395 465 193 596 436

## mutant\_KOa2.bam mutant\_KOa3.bam mutant\_KOb1.bam mutant\_KOb2.bam

## 497097 16 27 20 0

## 100503874 0 4 5 0

## 100038431 0 8 0 0

## 19888 0 26 14 6

## 20671 8 33 33 12

## 27395 686 572 1378 1901

## mutant\_KOb3.bam

## 497097 2

## 100503874 2

## 100038431 0

## 19888 16

## 20671 24

## 27395 1553

head(sample\_description)

## filename.sample.condition.batch

## 1 control\_FFa1.bam\tFFa1\tFFa\ta

## 2 control\_FFa2.bam\tFFa2\tFFa\tb

## 3 control\_FFa3.bam\tFFa3\tFFa\tc

## 4 TMC\_mutant\_KOa1.bam\tKOa1\tKOa\ta

## 5 TMC\_mutant\_KOa2.bam\tKOa2\tKOa\tb

## 6 TMC\_mutant\_KOa3.bam\tKOa3\tKOa\tc

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

*# Check dimensions*

col\_data <- sample\_description

dim\_count\_data <- dim(count\_data)

dim\_col\_data <- dim(col\_data)

*# Print dimensions*

cat("Dimensions of count\_data: ", dim\_count\_data[1], "rows by", dim\_count\_data[2], "columns\n")

cat("Dimensions of col\_data: ", dim\_col\_data[1], "rows by", dim\_col\_data[2], "columns\n")

*# Ensure the number of columns in count\_data matches the number of rows in col\_data*

**if** (dim\_count\_data[2] == dim\_col\_data[1]) {

cat("The dimensions align correctly.\n")

} **else** {

cat("Warning: The dimensions do not align! Check your data.\n")

}

## Dimensions of count\_data: 26301 rows by 9 columns

## Dimensions of col\_data: 9 rows by 1 columns

## The dimensions align correctly.

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

*# Load DESeq2*

**library**(DESeq2)

*# Extract 'condition' from the 'filename.sample.condition.batch' column*

col\_data$condition <- sapply(strsplit(col\_data$filename.sample.condition.batch, "\_"), **function**(x) x[1])

*# Convert 'condition' to a factor and set the reference level, if there is one*

col\_data$condition <- factor(col\_data$condition)

*# Now, check the new 'condition' column to ensure it looks correct*

head(col\_data$condition)

## [1] control control control TMC TMC TMC

## Levels: control TMC

*# Now create the DESeqDataSet*

dds <- DESeqDataSetFromMatrix(countData = count\_data,

colData = col\_data,

design = ~ condition)

*# Check the constructed DESeqDataSet object*

dds

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

*# Perform the rlog transformation*

*# 'blind=TRUE' for exploratory purposes*

rlog\_dds <- rlog(dds, blind=TRUE)

*# Perform the Variance Stabilizing Transformation (VST)*

*# Similarly, 'blind=TRUE' for exploratory purposes*

vst\_dds <- vst(dds, blind=TRUE)

*# At this point, rlog\_dds and vst\_dds contain the transformed data*

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

*# Make sure the DESeq2 object has been processed through rlog or VST*

*# If not, run the transformations*

rlog\_dds <- rlog(dds, blind = TRUE)

vst\_dds <- vst(dds, blind = TRUE)

*# For the heatmap, choose which transformation to use: rlog or VST.*

*# Transformation using rlog.*

transformed\_counts <- assay(rlog\_dds)

*# Identify the top 40 genes with the highest mean expression*

top\_genes <- head(order(rowMeans(transformed\_counts), decreasing = TRUE), 40)

*# Extract the subset of the rlog-transformed data for the top genes*

top\_genes\_rlog\_data <- transformed\_counts[top\_genes,]

*# Draw the heatmap using pheatmap*

pheatmap(top\_genes\_rlog\_data,

cluster\_rows = TRUE,

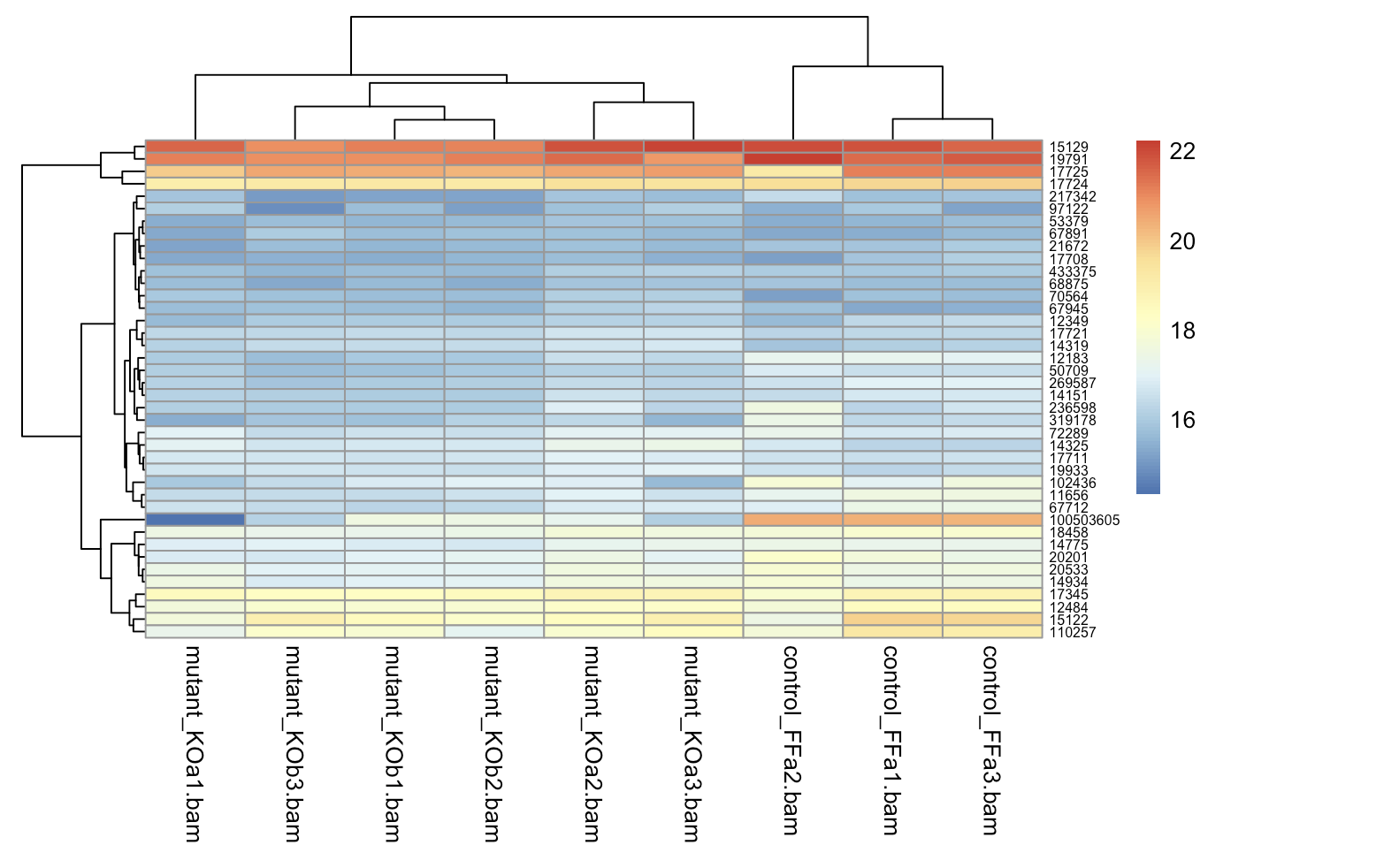
cluster\_cols = TRUE,

show\_rownames = TRUE,

show\_colnames = TRUE,

fontsize\_row = 6,

fontsize\_col = 10)

****

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

*# Calculate sample distance matrix*

sample\_dist <- dist(t(count\_data))

*# Perform hierarchical clustering*

sample\_hclust <- hclust(sample\_dist, method = "complete")

*# Plot the dendrogram*

A diagram of a clustering gene

Description automatically generatedplot(sample\_hclust, main = "Sample Clustering Dendrogram")

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

*# Perform PCA using rlog-transformed data*

ls()

## [1] "average\_speed" "cars"

## [3] "col\_data" "count\_data"

## [5] "count\_data\_path" "dds"

## [7] "dim\_col\_data" "dim\_count\_data"

## [9] "estimated\_reaction\_time" "fibonacci"

## [11] "i" "intercept"

## [13] "intercept\_se" "model"

## [15] "model\_summary" "mtcars"

## [17] "next\_fib" "p"

## [19] "result" "result\_n10"

## [21] "result\_n100" "result\_n20"

## [23] "rlog\_dds" "sample\_description"

## [25] "sample\_description\_path" "sample\_dist"

## [27] "sample\_hclust" "slope"

## [29] "slope\_se" "standard\_error\_reaction\_time"

## [31] "sumfun" "top\_genes"

## [33] "top\_genes\_rlog\_data" "transformed\_counts"

## [35] "vst\_dds"

pca\_rlog <- prcomp(t(assay(rlog\_dds)))

*# Extract the proportion of variance explained by each principal component*

variance\_explained <- pca\_rlog$sdev^2 / sum(pca\_rlog$sdev^2)

*# Calculate the percentage significance values of the first two principal components*

percentage\_significance <- variance\_explained[1:2] \* 100

*# Print the percentage significance values*

percentage\_significance

## [1] 34.98241 31.37770

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

*# Perform PCA using VST-transformed data*

pca\_vst <- prcomp(t(assay(vst\_dds)))

*# Extract the proportion of variance explained by each principal component for VST-transformed data*

variance\_explained\_vst <- pca\_vst$sdev^2 / sum(pca\_vst$sdev^2)

*# Calculate the percentage significance values of the first two principal components for VST-transformed data*

percentage\_significance\_vst <- variance\_explained\_vst[1:2] \* 100

*# Plot the PCA for comparison with rlog-transformed data*

par(mfrow = c(1, 2)) *# Set up a 1x2 plotting grid*

plot(pca\_rlog$x[,1], pca\_rlog$x[,2], xlab = "PC1", ylab = "PC2", main = "PCA (rlog)")

plot(pca\_vst$x[,1], pca\_vst$x[,2], xlab = "PC1", ylab = "PC2", main = "PCA (VST)")

**A comparison of a graph

Description automatically generated with medium confidence**

**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**✓**
* *# Load the GenomicRanges Library*
* suppressPackageStartupMessages(
* **library**(GenomicRanges)
* )
* *# Set the path to the peak files*
* melPeak\_Rep1 <- read.delim("/Users/estebantato/Desktop/LMS\_ChIPseq\_short-master-2023-final/mycmelrep1\_peaks.xls",sep="\t",comment.char = "#")
* melPeak\_Rep2 <- read.delim("/Users/estebantato/Desktop/LMS\_ChIPseq\_short-master-2023-final/mycmelrep2\_peaks.xls",sep="\t",comment.char = "#")
* *# Create a GRanges object for the first replicate.*
* melRep1\_GR <- GRanges(
* seqnames=melPeak\_Rep1[,"chr"],
* IRanges(melPeak\_Rep1[,"start"],
* melPeak\_Rep1[,"end"]
* )
* )
* *# Add metadata columns to the GRanges object.*
* mcols(melRep1\_GR) <- melPeak\_Rep1[,c("abs\_summit", "fold\_enrichment")]
* *# Print the GRanges object to view its contents.*
* melRep1\_GR

## GRanges object with 52933 ranges and 2 metadata columns:

## seqnames ranges strand | abs\_summit fold\_enrichment

## <Rle> <IRanges> <Rle> | <integer> <numeric>

## [1] 1 3049670-3049833 \* | 3049799 4.61726

## [2] 1 3435991-3436154 \* | 3436060 5.56316

## [3] 1 4774935-4775285 \* | 4775250 4.18060

## [4] 1 4775337-4775959 \* | 4775616 9.52390

## [5] 1 4847544-4847931 \* | 4847795 7.45675

## ... ... ... ... . ... ...

## [52929] Y 1913013-1913289 \* | 1913171 6.92589

## [52930] Y 1934470-1934640 \* | 1934521 4.61726

## [52931] Y 1964602-1964765 \* | 1964756 3.65200

## [52932] Y 2555745-2555908 \* | 2555815 5.72846

## [52933] Y 2890951-2891338 \* | 2891207 2.83564

## -------

## seqinfo: 22 sequences from an unspecified genome; no seqlengths

*# Create a GRanges object for the second replicate.*

melRep2\_GR <- GRanges(

seqnames=melPeak\_Rep2[,"chr"],

IRanges(melPeak\_Rep2[,"start"],

melPeak\_Rep2[,"end"]

)

)

*# Add metadata columns to the GRanges object*

mcols(melRep2\_GR) <- melPeak\_Rep2[,c("abs\_summit", "fold\_enrichment")]

*# Print the GRanges object to view its contents.*

melRep2\_GR

## GRanges object with 41443 ranges and 2 metadata columns:

## seqnames ranges strand | abs\_summit fold\_enrichment

## <Rle> <IRanges> <Rle> | <integer> <numeric>

## [1] 1 4506720-4506910 \* | 4506852 5.79027

## [2] 1 4775356-4776102 \* | 4775578 11.70799

## [3] 1 4797708-4797962 \* | 4797822 3.75350

## [4] 1 4847097-4848049 \* | 4847802 5.81891

## [5] 1 4848406-4848603 \* | 4848479 4.18488

## ... ... ... ... . ... ...

## [41439] Y 581955-582358 \* | 582206 8.34254

## [41440] Y 622263-622682 \* | 622529 4.57042

## [41441] Y 622802-623337 \* | 623093 5.85560

## [41442] Y 1903412-1903639 \* | 1903501 6.03422

## [41443] Y 1905483-1905649 \* | 1905522 5.12232

## -------

## seqinfo: 22 sequences from an unspecified genome; no seqlengths

length(melRep1\_GR)

## [1] 52933

length(melRep2\_GR)

## [1] 41443

*# Now we can use our GRanges to identify the peak in common or unique to replicates*

*# There are many options, one of them is Using table*

table(melRep1\_GR %over% melRep2\_GR)

## FALSE TRUE

## 22700 30233

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) ✓**

*# Extract sequences for the selected peaks*

*# Sort peaks by fold enrichment*

sorted\_peaks <- melRep1\_GR[order(mcols(melRep1\_GR)$fold\_enrichment, decreasing = TRUE)]

*# Select top 500 peaks*

top\_500\_peaks <- sorted\_peaks[1:500]

*# Resize peaks to 200bp around center*

resized\_peaks <- resize(top\_500\_peaks, width = 200, fix = "center")

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) ✓**

*# Set up required libraries*

*# Libraries were installed at the begining so they dont show the result within the html.*

genome <- BSgenome.Mmusculus.UCSC.mm9

seqlevelsStyle(resized\_peaks) <- "UCSC"

*# Extract sequences underneath the resized peaks*

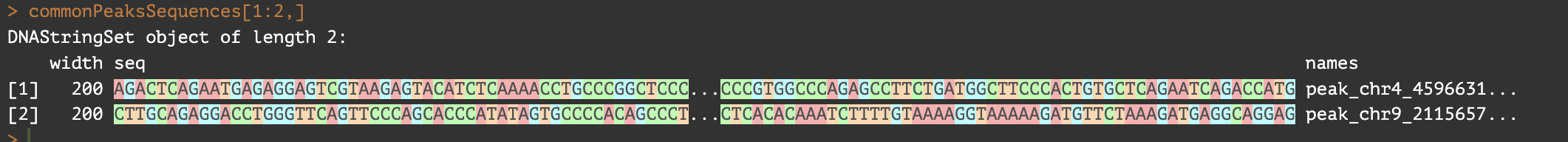
commonPeaksSequences <- getSeq(genome, resized\_peaks)

names(commonPeaksSequences) <- paste0("peak\_", seqnames(resized\_peaks), "\_",

start(resized\_peaks), "-",

end(resized\_peaks))

commonPeaksSequences[1:2,]

****

**A black screen with white text

Description automatically generated**

*# Fasta file*

writeXStringSet(commonPeaksSequences,file="consensusPeaks.fa")

*# Download the FASTA file to the specified directory*

*# Define the source and destination file paths*

source\_file <- "consensusPeaks.fa"

destination\_dir <- "/Users/estebantato/Desktop/LMS\_ChIPseq\_short-master-2023-final/"

*# Copy the file to the destination directory*

file.copy(source\_file, destination\_dir)

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

**A blue text on a white background

Description automatically generated**

**MEME-ChIP HTML output**

**A screenshot of a computer

Description automatically generated**

**MEME-ChIP TSV output**

**A black screen with white text

Description automatically generated**

**COMPLETE RESULTS LINK:**

<https://meme-suite.org/meme/info/status?service=MEMECHIP&id=appMEMECHIP_5.5.517132784467271555761413>