Advanced Bioinformatics (7BBG2016): Practical Bioinformatics Data Skills

Student ID: m2408689

GITHUB REPOSITORY FOR ALL FILES:
https://github.com/cemfl/ngs_assignment/tree/main

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

It stands for C – move up two directories. One '.' means the current directory and two '..' means the next directory up.

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

The command 'cd /' means change directory (cd) to the root.

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

You would use:

cp –help

1.4 What does the command pwd do?

The command 'pwd' shows you the current directory path from the root.

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)

To display the listing of the files and their details in the current directory you would use:

ls -lF

For another directory you would use:

Ls -lF directory_name/

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

To print the first 15 lines of all files ending in '.txt', you would use this code:

head -n 15 *.txt

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

To rename a file you can use the move command. Here is an example if you are in the folder the file is in:

mv example.txt new_example.txt

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

To display the contents of a file called myfile.txt you can simply use the command:

cat myfile.txt

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

You can use the mkdir command to make a new directory. This command would make the directory in the current folder:

mkdir flower

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

To change the current directory to /usr/local/bin you can use:

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)

To display all files in a folder in your current directory, including hidden files, you can use:

ls -la

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

To move to a parent directory you would use the cd command like so:

cd ..

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

To create a sub-directory in your home directory, you should use:

mkdir ~/new_directory

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

To list the first lines (5 in this case) in a txt file you can use:

head -n 5 textfile.txt

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

To display the last few lines (5 in this case) of a txt file you can use:

tail -n 5 textfile.txt

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

To extract a column from a text file, presuming you want to write it to another text file and extract the second column, you would use:

cut -d" " -f2 textfile.txt > textfile_column.txt

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

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)

To search for a string called "Hypertension" at the end of a tine you would use grep, the string and "$" which ensures it's at the end of the line. For example:

grep "Hypertension$" diseases.txt

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

To view hidden files hidden in your home directory you would call all files including hidden, then use grep to show only the files beginning with ".", which are the hidden files:

ls -a | grep "^\."

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

To keep a job running you would use nohup (no hangup) to start the code and end it in "&" to keep it running in the background. For example:

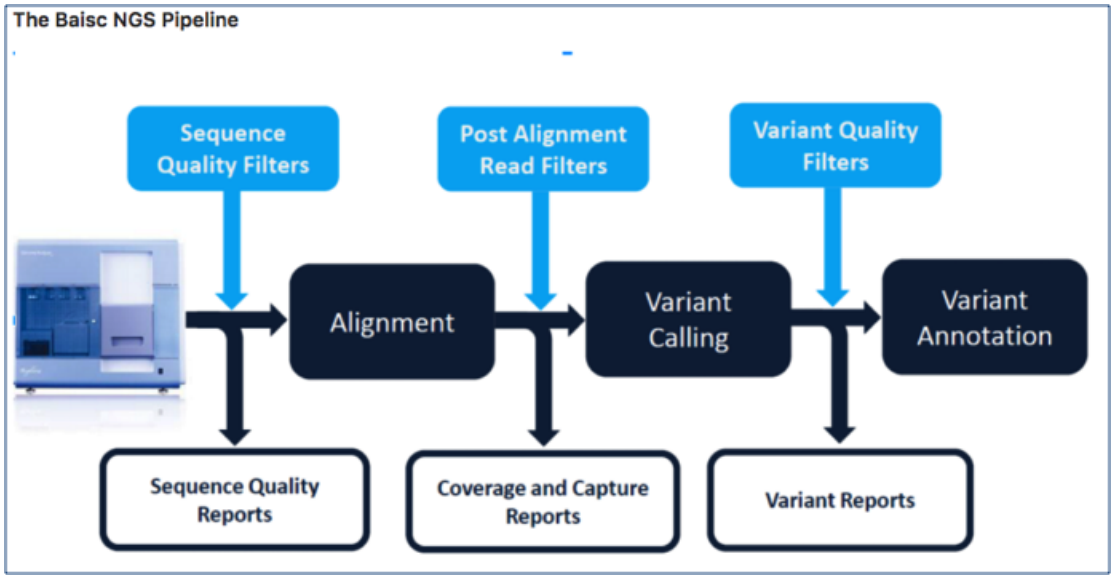
nohup bash ngs_pipeline.sh &

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)

I have the dependencies in a file called environment.yml To create the environment the code is:

conda env create -f environment.yml

This installs the following in a conda environment called 'bioinfo':

```
name: bioinfo
channels:
  - bioconda
  - conda-forge
  - defaults
dependencies:
  - mosdepth
  - prodigal
  - python=3.6.15
  - bwa=0.7.18
  - samtools=1.3.1
  - freebayes=1.1.0
  - picard=2.20.4
  - bedtools=2.31.1
  - trimmomatic=0.39
```

```
 - fastqc=0.12.1
 - vcflib=1.0.0_rc1
 - numpy=1.19.5
 - bcftools # only needed for the alternate variant caller
```

To activate the environment, the code is:

```
conda activate bioinfo
```

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

This is what I used to download the files. The main code is wget with the directory the files will be downloaded to and the urls of R1 and R2. I put the directories and urls in variables to make the code easier to read and change.

```
rep1url="https://s3-eu-west-
1.amazonaws.com/workshopdata2017/NGS0001.R1.fastq.qz"
rep2url="https://s3-eu-west-
1.amazonaws.com/workshopdata2017/NGS0001.R2.fastq.qz"

wget -P “$untrimmed_dir” “$rep1url” “$rep2url”
```

I also added code to change the filenames after “*.fastq.” to fastq.gz for compatibility, as the downloaded files ended in ‘.fastq.qz

```
for f in ngs_pipeline/data/untrimmed_fastq/*.fastq.*; do
   newname="${f%%.fastq.*}.fastq.gz"
   mv "$f" "$newname"
   echo "Renamed: \"$f\" → \"$newname\""
done
```

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)

To perform quality assessment, I used fastqc, which is widely used. The main use of this is to check per-base quality scores which are useful to identify low quality sequences, and over-representative sequences (such as primers). It also outputs an html file with the data. It’s useful to help decide how to set up the trimming step, which follows, or if you need to repeat sequencing. I used the following code:

```
 fastqc --threads $threads \
  "${untrimmed_dir}"*.fastq.gz \
  -o $fastqc
```

My $threads variable was set to 4 to use four cores. For the input, the code directs to use all files ending in ‘*.fastq.gz’, from the directory the ‘*.fastq.gz’ files were downloaded to.

The files are output into a directory set out in my variables. The qc from my data showed the reads were of high quality.

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

This can be done by repeating fastqc on the trimmed files and outputting a different file (or in a different folder):

```
fastqc --threads $threads \
  "${trimmed_dir}"*.fastq.gz \
  -o "${fastqc}_trim"
```

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)

Prior to using BWA MEM I unzipped and indexed the reference genome downloaded earlier using:

```
zcat $ref_zip > $ref_unzip
bwa index $ref_unzip
```

Then I used BWA MEM, which maps the DNA sequence to a reference genome. In this case hg19 that was downloaded and indexed using BWA INDEX:

```
bwa mem \
 -t 2 \
 -v 1 \
 -R
"@RG\tID:11V6WR1.111.D1375ACXX.1\tSM:NGS0001\tPL:ILLUMINA\tLB:NGS0001-
library\tDT:2025-04-15\tPU:11V6WR1.111.D1375ACXX.1" \
 -I 250,50 \
 "$ref_unzip" \
 "$trimmed_R_1P" \
 "$trimmed_R_2P" \
 > "$samfile"
```

I specified using 2 threads as my machine has 4 and couldn't cope with using 4. For the read group I used 'zcat NGS0001.R1.fastq.gz | head -n 1' to find out the instrument ID, the run number, the flowcell id and the lane, and added these with the fastq filename. I then input the two trimmed replicate sequence files and output these into a specified directory and filename.

- Perform duplicate marking (2pts)

The output of the alignment is a SAM file. Prior to performing duplicate marking, I compressed the SAM file into a BAM file:

samtools view -@ $threads -h -b "$samfile > "$bamfile

I then sorted the BAM file, which organises the reads by the position in the genome, speeding up the further steps. It is also required for indexing and tools like variant callers:

samtools sort -@ $threads "$bamfile" > "$sorted_bamfile"

Finally, I indexed the sorted bamfile:

samtools index "$sorted_bamfile"

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

I used Picard MarkDuplicates to mark the duplicates files:

picard MarkDuplicates I=$sorted_bamfile O=$marked_bamfile
M="${mark_dup}marked_dup_metrics.txt"

After this, I indexed the make file, filtered that and indexed the filtered marked bamfile:

samtools index "$marked_bamfile"

Samtools view extracts reads based on certain criteria and filteres can be added. 1796 is a strict filter that code for several things. In a nutshell, it only selects high quality reads mapped to forward strands, with no duplicates.

samtools view -F 1796 -q 20 -o "$filtered_bamfile" "$marked_bamfile"

samtools index "$filtered_bamfile"

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

```
samtools flagstat $bamfile > "${flagstats}bam_flagstat.txt"
samtools flagstat $sorted_bamfile > "${flagstats}bam_sorted_flagstat.txt"
samtools flagstat $marked_bamfile > "${flagstats}bam_marked_flagstat.txt"
samtools flagstat $filtered_bamfile > "${flagstats}bam_filtered_flagstat.txt"

# Performs idxstats

samtools idxstats "$filtered_bamfile" >
"${idxstats}${sample_id}_sorted_filtered_idxstats.txt"

# Performs insert size metrics

picard CollectInsertSizeMetrics \
   I="$filtered_bamfile" \
   O="${insert_size}metrics.txt" \
   H="${insert_size}histogram.pdf" \
   M=0.5
```

```
# Cuts bedfile for more efficient processing

 cut -f1,2,3 “$bedfile” > “$bedfile_stripped”

# Performs Depth of Coverage with mosdepth

 mosdepth --threads $threads --by $bedfile_stripped $dep_cov_prefix $filtered_bamfile
```

2.4. Variant Calling (4pts)

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

I used this code to use FreeBayes to call variants. The bed file was used to restrict analysis to specific regions:

```
freebayes \
 --bam "$filtered_bamfile" \
 --fasta-reference "$ref_unzip" \
 --targets "$bedfile" \
 --vcf "$vcfFile_unzip"
```

This code prepares the data for use in filtering:

```
bgzip "$vcfFile_unzip"
tabix -p vcf "$vcfFile_zip"
```

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

```
vcffilter -f "QUAL > 30 & QUAL / AO > 20 & SAF > 1 & SAR > 1 & RPR > 2 & RPL > 2" \
  "$vcfFile_zip" > "$vcfFile_filtered"

bgzip "$vcfFile_filtered"
tabix -p vcf "$vcfFile_filtered_zip"
```

2.5. Variant Annotation and Prioritization (10pts)

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

I used ANNOVAR for variant calling. The code I used downloads annotation databases for hg19 and saves them. The final section of code then uses this to annotate the variants produced using FreeBayes. This is the code to install ANNOVAR from the file in the same directory the script is run in. It must be manually downloaded prior to using the script:

```
 tar -zxvf annovar.latest.tar.gz
```

The next few lines:

```
annovar/annotate_variation.pl -buildver hg19 -downdb -webfrom annovar knownGene
humandb/
annovar/annotate_variation.pl -buildver hg19 -downdb -webfrom annovar refGene
humandb/
```

```
annovar/annotate_variation.pl -buildver hg19 -downdb -webfrom annovar ensGene
humandb/
annovar/annotate_variation.pl -buildver hg19 -downdb -webfrom annovar
clinvar_20180603 humandb/
annovar/annotate_variation.pl -buildver hg19 -downdb -webfrom annovar exac03
humandb/
annovar/annotate_variation.pl -buildver hg19 -downdb -webfrom annovar
dbnsfp31a_interpro humandb/

annovar/convert2annovar.pl -format vcf4 "$vcfFile_annotated_zip" > "$avinput"

annovar/table_annovar.pl $avinput humandb/ \
 -buildver hg19 \
 -out "${results_dir}${sample_id}_filtered_annotation" \
 -remove \
 -protocol refGene,ensGene,clinvar_20180603,exac03,dbnsfp31a_interpro \
 -operation g,g,f,f,f \
 -otherinfo \
 -nastring . \
 -csvout
```

SNPEFF is another variant annotator which uses a pre-built database, in this case along with hg19, to map variants and predict effecrs . The code I used for snpEff is:

```
 snpEff -v hg19 "ngs_pipeline/results/NGS0001_filtered.vcf.gz" \
 > "ngs_pipeline/results/NGS0001_snpeff.vcf"
```

- Perform basic variant prioritization: filter to exonic variants not seen in dbSNP (2pts)

I used BCFTOOL VIEW for basic variant prioritization. The code I used is here:

```
 bcftools view -i 'INFO/ANN ~ "exonic"' \
 ngs_pipeline/results/NGS0001_snpeff.vcf \
 -Oz -o ngs_pipeline/results/prioritized.vcf.gz
```

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)

I replaced Freebayes, the variant caller, with BCFTOOLS MPILEUP. This is shared as 'ngs_pipeline_alternate.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).

```
 bcftools mpileup \
  -f $ref_unzip \
  -a AD,DP \
```

```
  --threads $threads \
  -Ou $filtered_bamfile |
 bcftools call \
  -mv \
  --threads $threads \
  -Oz \
  -o $vcfFile_zip

tabix -f -p vcf $vcfFile_zip #index the compressed vcf

bcftools filter -i "QUAL > 10 && FORMAT/DP > 20 && FORMAT/AD[0:1] > 8 &&
FORMAT/AD[0:1]/FORMAT/DP > 0.3" \
 -o $vcfFile_filtered_zip $vcfFile_zip

tabix -f -p vcf $vcfFile_filtered_zip
```

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 (25 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. (3pt)

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. (3pt)

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. (3pt)

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. (3pt)

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? (3pt)

3.6. Use *ggplot* to plot the data points from Task 3.5 and the linear fit. (3pt)

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. (7pt)

RNA-seq assessment (13 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.

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. **(2 pts)**
3.12. Draw a heatmap of count matrix based on the top 40 highly expressed genes using rlog and VST data. **(2 pts)**
3.13. Generate a SDM to see the clustering of count data. **(2 pts)**
3.14. Perform the Principal Component Analysis using rlog method and find out the % significance values of first two principal components. **(2 pts)**
3.15. Repeat the PCA, this time using VST method and compare the plots with the ones obtained using rlog method. **(2 pts)**

ChIP-seq assessment (7 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.

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. **(2 pts)**
3.18. Extract the sequences underneath the file and write them to FASTA file in you working directory. Inspect the file in notepad. **(2 pts)**
3.19. Upload the sequences to Meme-ChIP and report the results when complete. **(2 pts)**