# From FASTQ files to Variant Calling for RNA-Seq

*Anna Quaglieri*

# Contents

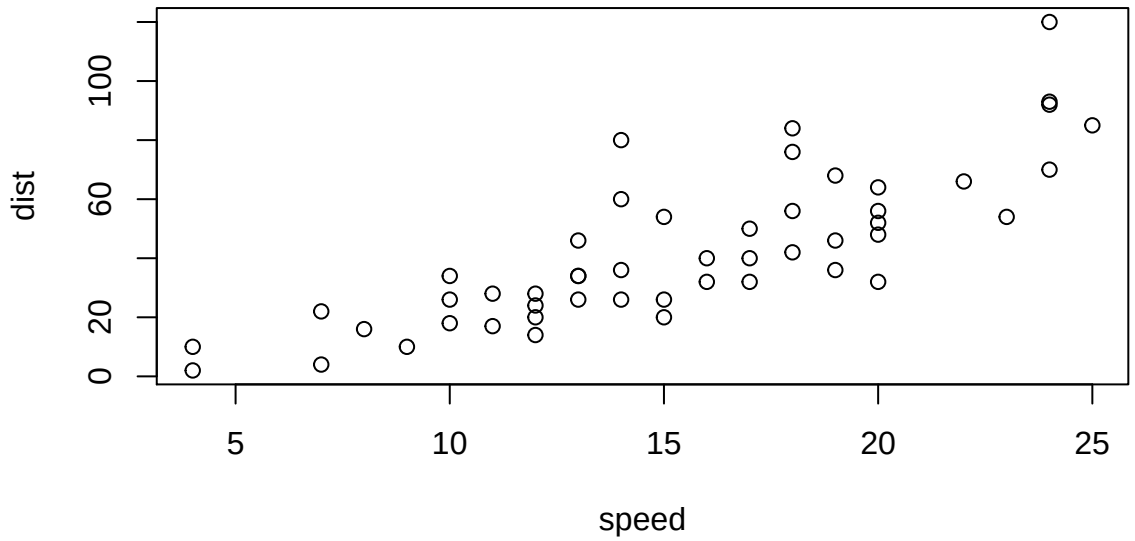
# Chapter 1

# Setup

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Cmd+Shift+Enter*.

```
plot(cars)
```




Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by pressing *Cmd+Option+I*.

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).

The preview shows you a rendered HTML copy of the contents of the editor. Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead, the output of the chunk when it was last run in the editor is displayed.

# Chapter 2

# Setup

This is an example workflow from `SRR` files to Variant calling using modular functions written in `R` and `bash`.

```
git clone git@github.com:annaquaglieri16/RNA-seq-variant-calling.git
```

All the functions used for the variant calling and downsampling pipeline are inside the `./functions` folder.

- If you already have the `FASTQ` files and you don't need to randomly downsample your samples go to Section **??**
- If you already have the `FASTQ` files and you want to randomly downsample your samples to a fix number of reads go to Section **??**
- If you already have the `BAM` files and you want to call variants go to Section **??**

## 2.1 Disclaimer

The following workflow was built in a modular way and it is not wrapped up into a pipeline manager. I aknowledge the limitations and non-user-friendliness of some steps. However, it offers a comprehensive view of several tools and steps used for variant calling in RNA-Seq as well as general tools used in any bioinformatics pipeline.

# Chapter 3

# Download RNA-Seq data from GEO

We will provide an example on how to download data from GEO using the Leucegene CBF-AML RNS-Seq data uploaded at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE49642. The first step in dowloading data from GEO is to download the `SRA` files. For that we need to get `SRX` entries which each one corresponds to a sample in the RNA-Seq cohort.

## 3.1 Get SRX sample names

```
library(GEOquery)
library(tidyverse)
library(knitr)
library(stringr)
```

Below is an example using one accession number from the Leucegene data.

```
# Get matrix files for every accession number
series_matrix_info <- function(gse){
gsed <- getGEO(gse,GSEMatrix=TRUE)
gse.mat <- pData(phenoData(gsed[[1]]))
reduced <- gse.mat[,c("title","geo_accession","relation.1")]
write.csv(reduced,file.path("../data",paste(gse,"_",nrow(gse.mat),".csv",sep="")),row.names = FALSE)
}

series_matrix_info("GSE49642") # 43 samples
```

Every row in Table 3.1 contains sample names (`title`) and `GSM` accession numbers. In order to download a particular sample we need the `SRA` terms which are the names starting with: `SRX***` in the `relation.1` column. The structure of the matrix might change across different studies but you should be able to find `SRX` entries hidden somewhere in the `GSEMatrix`!

```
matrix_file <- list.files(path = file.path("data"),pattern = "GSE",full.names = TRUE)
GSEmatrix <- read_csv(matrix_file)

kable(GSEmatrix[1:5,],caption="SRX sample names.")
```

With some string processing we can extract the `SRX` entries.

Table 3.1: SRX sample names.

| title | geo_accession | relation.1 |
|---|---|---|
| 02H053 | GSM1203305 | SRA: https://www.ncbi.nlm.nih.gov/sra?term=SRX332625 |
| 02H066 | GSM1203306 | SRA: https://www.ncbi.nlm.nih.gov/sra?term=SRX332626 |
| 03H041 | GSM1203307 | SRA: https://www.ncbi.nlm.nih.gov/sra?term=SRX332627 |
| 03H116 | GSM1203308 | SRA: https://www.ncbi.nlm.nih.gov/sra?term=SRX332628 |
| 03H119 | GSM1203309 | SRA: https://www.ncbi.nlm.nih.gov/sra?term=SRX332629 |

```
GSEmatrix$SRX <- stringr::str_extract(string = GSEmatrix$relation.1,pattern = "SRX[0-9][0-9][0-9][0-9][0
GSEmatrix$relation.1 <- NULL
kable(head(GSEmatrix))
```

| title | geo_accession | SRX |
|---|---|---|
| 02H053 | GSM1203305 | SRX332625 |
| 02H066 | GSM1203306 | SRX332626 |
| 03H041 | GSM1203307 | SRX332627 |
| 03H116 | GSM1203308 | SRX332628 |
| 03H119 | GSM1203309 | SRX332629 |
| 04H024 | GSM1203310 | SRX332630 |

## 3.2 Create NCBI query

```
search_ncbi <- paste(GSEmatrix$SRX,collapse=" OR ")
search_ncbi
```

```
## [1] "SRX332625 OR SRX332626 OR SRX332627 OR SRX332628 OR SRX332629 OR SRX332630 OR SRX332631 OR SRX33
```

Paste the search SRX332625 OR SRX332626 OR SRX332627 OR SRX332628 OR SRX332629 OR SRX332630 OR SRX332631 OR SRX332632 OR SRX332633 OR SRX332634 OR SRX332635 OR SRX332636 OR SRX332637 OR SRX332638 OR SRX332639 OR SRX332640 OR SRX332641 OR SRX332642 OR SRX332643 OR SRX332644 OR SRX332645 OR SRX332646 OR SRX332647 OR SRX332648 OR SRX332649 OR SRX332650 OR SRX332651 OR SRX332652 OR SRX332653 OR SRX332654 OR SRX332655 OR SRX332656 OR SRX332657 OR SRX332658 OR SRX332659 OR SRX332660 OR SRX332661 OR SRX332662 OR SRX332663 OR SRX332664 OR SRX332665 OR SRX332666 OR SRX332667 into NCBI https://www.ncbi.nlm.nih.gov/sra and follow the intructions in https://www.ncbi.nlm.nih.gov/sra/docs/sradownload/#download-sequence-data-files-usi **Download sequence data files using SRA Toolkit** to download all the `SRR` run names and information of the runs.

```
# Files are saved in the home directory under ncbi/public/sra
prefetch --option-file SraAccList_CBF-AML_Leucegene.txt
```

SRA files can then be converted to `fastq` files with `fastq-dump --split-files`.

# Chapter 4

# Downsampling FASTQ files

The seqtk tool can be used to downsample an exact number of reads from paired end (PE) FASTQ files. The following is an example run

```
path-to-seqtk-folder/seqtk sample -s100 test_data/SRR1608610_1.fastq.gz 10000 > test_data/sub_SRR160861
path-to-seqtk-folder/seqtk sample -s100 test_data/SRR1608610_2.fastq.gz 10000 > test_data/sub_SRR160861
```

# Chapter 5

# Define files and programs needed for the pipeline

- The reference genome **hg19** is used for this analysis.
- Below are all the programs and versions used

```
module load STAR/2.5.2
module load R/3.4.3
module load anaconda2/4.0.0
module load sambamba/0.6.6
module load picard-tools/2.9.4
module load gatk/3.7.0
module load varscan/2.3.9
module load vcftools/0.1.13
module load samtools/1.6
module load ensembl-vep/89.0
module load vcflib/1.0.0-rc1
module load vardict/1.5.1
module load freebayes/1.1.0
module load picard-tools/2.9.4
```

- The genome references and annotations used here have been downloaded from iGenome website

Below is an example of how to setup a few line of `bash` to assign directory names.

```
# Hard link to genome.fa of the reference genome
genome_fasta=path_to_hg19_genome_directory/genome.fa
# Hard link to gene.gtf where gene annotation is stored
gtf=path_to_hg19_gtf_directory/genes.gtf
```

```
# Functions directories
workdir=../functions

# STAR folders for one-pass, two-pass and merged output
star_1pass=../results/aligned_star1
star_2pass=../results/aligned_star2
star_merged=../results/star_merged_runs # Every sample comes in different SRR runs which will have to b
```

# Chapter 6

# FASTQC and adapters trimming

`fastqc` [1] can be used for QC of the FASTQ files.

```
fastqc ../data/SRR1608610_1.fastq.gz --outdir ./data/
```

## 6.1 Parallelise your `FASTQC`

This is just one example to run `fastqc` on several FASTQ files using `parallel` [5].

```
find ../data -name "*.fastq.gz" > ../data/fastq_files.txt
cat ../data/fastq_files.txt | parallel -j 2 "fastqc {} --outdir ../data"
```

## 6.2 Summarise reports with `MultiQC`

I strongly suggest to have a look at MultiQC [3] which allows you to combine together the results of multiple samples into one compact document. You can check the programs whose output is supported by `MultiQC`.

```
multiqc ../data/ --interactive -n "FASTQC_summary" -o ../data/
```

The `FASTQC` reports offer a variety of measures and one can decide about discarding some samples or doing some adapter trimming if necessary. Trimmomatic and Trim Galore! can be used for this purpose.

I suggest looking at one of my previous analyses around adapters with STAR and Subread since adapters can cause serious troubles with `STAR` default settings! Regarding this I strongly suggest to look at the fragment size distribution across samples once you have aligned your fatsq files. Unusual behaviour can help you spot problems with adapters/alignment steps, which I highlighted in this post. I normally use the CollectMultipleMetrics to extract fragment sizes from PE bamfiles. See Section @(sec:post-align) for more details.

# Chapter 7

# Alignement, Read Groups, Mark duplicates

Once the *fastq* files are ready to be processed we can align them with *STAR*. Subread/Rsubread is another widely used RNA-Seq aligner. The reason why I initially choose *STAR* over *Subread* was simply due to the fact that *STAR* can generate specific output for chimeric reads that can be directly used with STAR-Fusion to analyse gene fusions (see more in Section @(ch:fusions)). Also, *STAR* is suggested in the the GATK Best Practices to call variants in RNA-Seq.

## 7.1 Create `STAR` index

STAR requires to build an index for the reference genome that will be used in the analignment and fusion calling step.

```
# Iniaitlise Genome directory where to save STAR Index and STAR Fusion Index folders
star_genome100=path_to_genome_directory/star_index_hg19_99
mkdir -p ${star_genome100}
```

To build the *STAR index* one needs to provide the FASTA file for the reference genome used, a GTF file with information aabout the annotation and STAR also require an an extra parameter called `sjdbOverhang` which is usually set to be *(read length - 1)*. See `STAR` documentation for **Generating genome indexes** in the STAR manual - 99 is (read length - 1) relative to the samples that I was working with.

Below is a wrapper for `STAR` call to build an index.

```
outpud_dir=../results
../functions/build_STAR_index.sh $genome_fasta_path $gtf_path $outpud_dir "hg19" 99
```

## 7.2 STAR-1pass

If you are working with a cohort of bamfiles, STAR developer suggests to run the alignment in a two-pass mode. This consists of first aligning all the bamfiles, collecting the *splice junctions* as output of STAR and realign all the bamfiles with this new information. For more details about 1-pass, 2-pass-multi and 2-pass-single see Section 8 of the STAR documentation. In my pipeline I normally use the 2-pass multi strategy as below.

```
FQ1=../data/SRR1608907_1.fastq.gz
FQ2=../data/SRR1608907_2.fastq.gz
star_genome100=path_to_genome_directory/star_index_hg19_99

Rscript ../functions/run_STAR.R --genome_index $star_index_hg19_99 \
--fastqfiles $FQ1,$FQ2 \
--sampleName SRR1608907 \
--outdir ../results/star_1pass \
--STARmode "1Pass"
```

The `R` function above is a wrapper for the `STAR` call below:

```
# Version STAR/2.5

STAR --genomeDir path_to_star_index_hg19 \
--readFilesIn $FQ1 $FQ2 --runThreadN 27 --chimSegmentMin 10 --readFilesCommand zcat --alignSJoverhangMi
```

To see all the arguments available:

```
Rscript ../functions/run_STAR.R --help
```

After running STAR on all the fastq files available we can collect all the splice junctions from the first pass and use them for the second pass.

```
# concatenate splice junctions from all samples from ran in pass1
cat ../results/star_1pass/*SJ.out.tab > ../results/star_1pass/combined_sj.out.tab
# Dobin suggests to remove chrm cause they are usually False positives
awk '!/chrM/' ../results/star_1pass/combined_sj.out.tab > ../results/star_1pass/combined_sj_nochrM.out.
```

Again, for quality check, have a look at the amazing alignment summary enabled by MultiQC.

```
multiqc ../results/star_1pass --interactive -n "STAR_1passQC" -o ../results
```

## 7.3 STAR-2pass

The second pass alignment is exactly the same as the first one with only a few differences:

- the *sjfile* input created combining the splice junctions from the first pass
- STAR is run with the option of output chimeric reads switched on. This will allow fusion analysis.

The ouput of STAR is a bamfile already sorted by coordinate with the suffix `Aligned.sortedByCoord.out.bam`. At this stage we can also run two more steps `post_align_qc1.sh` and `post_align_qc2.sh`, discussed below.

```
FQ1=../data/SRR1608907_1.fastq.gz
FQ2=../data/SRR1608907_2.fastq.gz
star_genome100=path_to_genome_directory/star_index_hg19_99

Rscript ../functions/run_STAR.R \
--genome_index $star_index_hg19_99 \
--fastqfiles $FQ1,$FQ2 \
--sampleName SRR1608907 \
--outdir ../results/star_2pass --STARmode "2PassMulti" \
--sjfile ../results/star_1pass/combined_sj_nochrM.out.tab

# Run featurecounts and collect fragment sizes for QC
../functions/post_align_qc1.sh \
```

```
path_to_genome.fa \
path_to_genes.gtf \
../results/star_2pass/SRR1608907.Aligned.sortedByCoord.out.bam \
SRR1608907  # sample name

# Pre-process bamfile (add Read groups etc..)
../functions/post_align_qc2.sh \ ../results/star_2pass/SRR1608907.Aligned.sortedByCoord.out.bam \
SRR1608907 \
path_to_genome.fa \
SRR1608907
```

## 7.4 Details about post-alignment functions

- `post_align_qc1.sh` is optional:

1. Runs featureCounts to get gene counts and compute PCA to evaulate the concordance between bamfiles sequenced on different lanes. This allows a QC before merging the different bamfiles into a single one.
2. Runs CollectMultipleMetrics to collect the fragment distribution of the bamfiles (only possible with PE reads). This is also a good QC to check that the fragment distribution of bamfiles on different lanes is the same.

- `post_align_qc2.sh` contains necessary pre-prcessing steps:

1. Marks PCR duplicates (using sambamba markdup)
2. Add Read Groups to single runs before merging bamfiles (using AddOrReplaceReadGroups). Even if files do not need to be merges, `GATK` requires read groups to be added bamfiles.
3. Run ValidateSamFile to check for errors in the final bamfile.

In order, its arguments are:

1. Path to **aligned bamfile**;
2. `SampleName`. This is the name of the sample applied to the `RGID` and `RGPU` fields below.
3. `SampleName of the run`. If a sample was sequenced across different lanes you need to set lane-specific read groups to each separate bamfile (e.g. SampleName_L1, SampleName_L2). This sample name will be used for the fields `RGLB` and `RGSM` in the `AddOrReplaceReadGroups` groups below. See Chapter @(ch:merge-bamfiles) for merging bamfiles.

```
# Picard tool function to add read groups to a bamfile
AddOrReplaceReadGroups \
        I= ./star_2pass/SRR1608907.Aligned.sortedByCoord.out.bam \
        O= ./star_2pass/SRR1608907.Aligned.sortedByCoord.out.RG.bam \
        RGID=SRR1608907 \
        RGPU=SRR1608907 \
        RGLB=SRR1608907_L1 \
        RGPL="illumina" \
        RGSM=SRR1608907_L1
```

After running `post_align_qc2.sh` a file with the suffix `Aligned.reorderedDupl.rg.bam` will be created where read groups are added and PCR duplicated reads marked.

This time *MultiQC* will give us a summary output also of the fragment distributions and gene counts if the output files are stored within the `star_2pass` folder.

```
multiqc ../results/star_2pass --interactive -n "STAR_2passQC" -o ../results
```

# Chapter 8

# Merge bamfiles

In some cases the sequenced reads from one sample can be sequenced across different lanes and the aligned bamfiles need to be merged. `sambamba merge` can be used for this. I created a wrapper function for it even though it assumes that the files from the same sample have a common *SampleName*. The function will merge together all bamfiles containing *SampleName*.

```
../functions/merge_runs.sh SampleName ./star_2pass
```

# Chapter 9

# GATK pre-processing

This pipeline contains function to call variants with `MuTect2`, `Samtools + VarScan2`, `VarDict` and `Freebayes`. In order to run `MuTect2` some GATK pre-processing are needed. The function `function/gatk_process_pipe.R` will perform the following steps:

- *SplitNCigarReads* see GATK documentation
- *Base recalibration* see GATK documentation.

Below is an example call which wraps the steps above and check if files have already been created.

```
Rscript ../functions/gatk_process_pipe.R \
--reference_fasta path_to_genome.fa \
--bamfile ../results/star_2pass/SRR1608907Aligned.reorderedDupl.rg.bam \
--sampleName SRX381851 \
-knownSites path_to_GATK_Bundle_files/dbsnp_138.hg19.excluding_sites_after_129.vcf \
-knownSites path_to_GATK_Bundle_files/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf \
-knownSites path_to_GATK_Bundle_files/1000G_phase1.indels.hg19.sites.vcf
```

The function above is a wrapper for the following `GATK3` calls.

## 9.1 SplitNCigarReads

```
gatk -T SplitNCigarReads -R path_to_genome.fa \
-I ../results/star_2pass/SRR1608907Aligned.reorderedDupl.rg.bam \
-o ../results/star_2pass/SRR1608907Aligned.reorderedDupl.rg.split.bam \
--filter_mismatching_base_and_quals -U ALLOW_N_CIGAR_READS -rf ReassignOneMappingQuality -RMQF 255 -RMQ
--log_to_file ../results/star_2pass/SRR1608907_RG_DUPL_SPLIT_log
```

## 9.2 Base recalibration

Base recalibration using known sites downloaded from the GATK Bundle

```
module load gatk/3.7.0

gatk -T BaseRecalibrator -R path_to_genome.fa \
-I ../results/star_2pass/SRR1608907Aligned.reorderedDupl.rg.split.bam -nct 8 \
-knownSites path_to_GATK_Bundle_files/dbsnp_138.hg19.excluding_sites_after_129.vcf \
```

```
-knownSites path_to_GATK_Bundle_files/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf \
-knownSites path_to_GATK_Bundle_files/1000G_phase1.indels.hg19.sites.vcf \
-o ../results/star_2pass/BaseQRecal/SRR1608907/SRR1608907_recal_data.table \
--log_to_file ../results/star_2pass/BaseQRecal/SRR1608907/SRR1608907_recal_step1_log

gatk -T BaseRecalibrator -R path_hg19_reference/genome.fa \
-I ../results/star_2pass/SRR1608907Aligned.reorderedDupl.rg.split.bam -nct 8 \
-knownSites path_to_GATK_Bundle_files/dbsnp_138.hg19.excluding_sites_after_129.vcf \
-knownSites path_to_GATK_Bundle_files/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf \
-knownSites path_to_GATK_Bundle_files/1000G_phase1.indels.hg19.sites.vcf \
-BQSR ../results/star_2pass/BaseQRecal/SRR1608907/SampleName_recal_data.table \
-o ../results/star_2pass/BaseQRecal/SRR1608907/SRR1608907_post_recal_data.table \
--log_to_file ../results/star_2pass/BaseQRecal/SRR1608907/SRR1608907_recal_step2_log

gatk -T AnalyzeCovariates -R path_hg19_reference/genome.fa \
-before ../results/star_2pass/BaseQRecal/SRR1608907/SSRR1608907_recal_data.table \
-after ../results/star_2pass/BaseQRecal/SRR1608907/SRR1608907_post_recal_data.table \
-csv ../results/star_2pass/BaseQRecal/SRR1608907/SSRR1608907_recalibration_plots.csv \
-plots ../results/star_2pass/BaseQRecal/SRR1608907/SRR1608907_recalibration_plots.pdf \
--log_to_file ../results/star_2pass/BaseQRecal/SRR1608907/SRR1608907_recal_analyseCov_log

gatk -T PrintReads -R path_hg19_reference/genome.fa \
-I ../results/star_2pass/SRR1608907Aligned.reorderedDupl.rg.split.bam \
-o ../results/star_2pass/SRR1608907Recal.reorderedDupl.rg.split.bam \
-nct 8 -BQSR ../results/star_2pass/BaseQRecal/SRR1608907/SRR1608907_post_recal_data.table \
--log_to_file ../results/star_2pass/BaseQRecal/SRR1608907/SRR1608907_Log_recalibrated_bases
```

# Chapter 10

# Variant calling

You can perform variant calling on the whole genome or to a specific regions that you can specify with a `.bed` file. Here we show how to call variants on specific regions of interest. The target regions were created using the gene symbols of the mutations listed in **Supplemental Table 3** of [4] and listed in `../data/mutations_Lavallee_2016.csv`. We used the hg19 inbuilt annotation of `Rsubread` to obtain the gene ranges and added 500 bp at the end and at the beginning of each gene. The final `bed` files is `../data/target_regions.bed`.

View all the options in the `../functions/call_variants.R` function.

```
Rscript ../functions/call_variants.R --help
```

Below is an example to run `VarDict` and `MuTect2` in tumour-only mode. The directory needed for VarDict can be downloaded from the software GitHub page https://github.com/AstraZeneca-NGS/VarDict.

The function above wraps up calls for the callers included in the pipeline. Below are details about the actual caller settings that I normally use to call variants with these callers..

## 10.1 Call with MuTect2

```
module load gatk/3.7.0

gatk -T MuTect2 -R path_to_genome.fa \
-I:tumor ../results/star_2pass/SRR1608907_Aligned.reorderedDupl.rg.split.recalibrated.bam \
-L ../data/target_regions.bed \
-o ../results/mutect/regions/SampleName_germline_snvs_indels.vcf \
-log ../results/mutect/regions/SampleName_germline_snvs_indels_log
```

## 10.2 Call with Samtools + VarScan2

```
module load varscan/2.3.9
module load samtools/1.6

samtools mpileup --output-tags AD,ADF,ADR,DP,SP \
--fasta-ref -R path_to_genome.fa \
-l ../data/target_regions.bed ../results/star_2pass/SRR1608907_Aligned.reorderedDupl.rg.split.recalibra
```

## 10.3 Call with VarDict

The `teststrandbias.R` and `var2vcf_valid.pl` scripts were downloaded from https://github.com/AstraZeneca-NGS/VarDict. VarDict only calls variants on a target region.

```
module load vardict/1.5.1

vardict -c 1 -S 2 -E 3 -g 4 -r 2 -t -th 10 -v -G \
-R path_to_genome.fa \
-b ../results/star_2pass/SRR1608907_Aligned.reorderedDupl.rg.bam ../data/target_regions.bed  | vardict_
```

## 10.4 Annotate variants

Below is an example using the output from VarScan but the same call is used for Mutect2 and VarDict vcf files.

```
module load ensembl-vep/89.0

vep --dir_cache dir_to_VEP_cache/.vep --offline \
-i ../results/varscan/regions/SampleName_germline_snvs_indels.vcf \
-o ../results/varscan/regions/annotated_variants/SampleName_germline_annotated.vcf \
--cache --everything --force_overwrite --assembly GRCh37 --fork 12 --vcf --port 3337
```

## 10.5 Targeted INDEL calling with `km` algorithm

The `km` algorithm [2] is one of the sofware developed for targeted INDEL calls in RNA-Seq. Calling INDELs from RNA-Seq is challenging and several tools have been published between 2017-2019 to accomplish this task. The `km` algorithm was benchmarked on the AML Leucegene and TCGA-LAML datasets.

# Chapter 11

# Standardise output of variants

We are now at the stage where variants are called. The scripts above helps with calling variants using 4 callers (VarDict, Varscan, MuTect2, km).

At this stage, depending on how one needs to use the variants, there are several options:

## 11.1 superFreq to combine time-course mutations for one patient

If you have time-course data for each patient you can use software like `superFreq` to analyse mutations over time; estimate the clonality development of your cancer samples; CNVs; somatic mutations etc...

If you decide to call variants with one of the callers above independently from the `call_variants.R` function, you still have options to standardise the `VCF` output from a caller so to explore it with the `Mutexplore` app. - No annotation is run -> apply function from `samplepower` - Annotation is run -> apply function from `samplepower` - Output is taken from superFreq -> functions from `lineplots`

# Chapter 12

# Call fusions with `STAR-Fusion` and `FusionInspector`

Here we suggest to ways to detect fusions from RNA-Seq samples which depends on whether one is doing an exploratory or confirmatory analysis for specific recurrent fusions (requiring high sensitivity). The former analysis can be performed with `STAR-Fusion` and the second analysis with `FusionInspector` which is now integrated into `STAR-Fusion` as a sub module. `STAR-Fusion` uses the chimeric reads output from the `STAR` aligner to detect fusion trascripts. The wiki page of `STAR-Fusion` describes how to prepare all the necessary files and software requirements to run the fusion callers. These steps are also summarised in the `../functions/star_fusion_prepare.sh` file and they need to be run only once. The wiki pages for both software also provide a detailed information about how to interpret the results.

# Chapter 13

# Analyse time-course mutations with superFreq

# Chapter 14

# Explore mutations with the `Mutexplore` Shiny app

# Bibliography


[1] Simon Andrews. *FastQC: a quality control tool for high throughput sequence data*. 2010.

[2] Eric Olivier Audemard, Patrick Gendron, Vincent-Philippe Lavallée, Josée Hébert, Guy Sauvageau, Sébastien Lemieux. "Targeted variant detection in leukemia using unaligned RNA-Seq reads". In: *bioRXiv* (Apr. 2018).

[3] Philip Ewels et al. "MultiQC: summarize analysis results for multiple tools and samples in a single report". en. In: *Bioinformatics* 32.19 (Oct. 2016), pp. 3047–3048.

[4] Vincent-Philippe Lavallée et al. "RNA-sequencing analysis of core binding factor AML identifies recurrent ZBTB7A mutations and defines RUNX1-CBFA2T3 fusion signature". In: *Blood* (Mar. 2016).

[5] Ole Tange. "GNU Parallel: The Command-Line Power Tool". In: ().