Supporting scripts: Finding a suitable library size to call variants in RNA-Seq

Anna Quaglieri

Contents

Chapter 1

Setup

To run the workflow in the next sections, clone the repository below containing and load the R project:

```
git clone git@github.com:annaquaglieri16/Supporting-scripts-library-size-RNA-Seq.git

cd ./Supporting-scripts-library-size-RNA-Seq
```

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

- To download and QC example files from the Leucegene cohort, see Chapter 2.2
- To follow step by step and see the settings used to align, call and create a standardised output of the variants see Chapter 3. While, to run the whole workflow for the samples downloaded it is possible to directly run the script `./functions/align-call-stand.sh` after appropriately setting input files and directory and downloading the required programs.
- Chapter 5 contains information about the proportions used to downsample the TCGA-LAML samples.

Chapter 2

Download, quality control and downsample the Leucegene RNA-Seq samples

2.1 Download RNA-Seq data from GEO

The initial unaligned RNA-Seq CBF-AML samples are downloaded from GEO using the accession numbers GSE49642, GSE52656, GSE62190, GSE66917, and GSE67039. Only 46 FASTQ files of samples whose mutations are reported in Supplemental Table 3 of [1] are kept for the analysis. Sample SRX381851 is removed due to low number of available reads. The file `data/CBF_AML_samples.csv` contains the sample names of the 46 samples used in the study.

Below is an example using accession number GSE52656 where one of the CBF-AML samples is present, namely 07H099.

2.1.1 Get SRX sample names

The `GEOquery` package can be used to extract `SRX` files linked to an accession number.

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

Below the accession number **GSE52656** is used as example.

```
# 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("GSE62190")
```

Table 2.1: SRX sample names linked to the accession number GSE62190

title	geo_accession	relation.1
02H017	GSM1521543	SRA: https://www.ncbi.nlm.nih.gov/sra?term=SRX729552
06H152	GSM1521544	SRA: https://www.ncbi.nlm.nih.gov/sra?term=SRX729553
08H085	GSM1521545	SRA: https://www.ncbi.nlm.nih.gov/sra?term=SRX729554
08H129	GSM1521546	SRA: https://www.ncbi.nlm.nih.gov/sra?term=SRX729555
12H057	GSM1521547	SRA: https://www.ncbi.nlm.nih.gov/sra?term=SRX729556

Every row in Table 2.1 contains sample names (`title`) and `GSM` 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.

```
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 linked to the accession number GSE62190")
```

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

```
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
02H017	GSM1521543	SRX729552
06H152	GSM1521544	SRX729553
08H085	GSM1521545	SRX729554
08H129	GSM1521546	SRX729555
12H057	GSM1521547	SRX729556
01H001	GSM1521548	SRX729557

The next chunk extracts only samples whose variants were published in [1].

```
# CBF-AML Sample
CBF_AML_info <- read.csv("data/CBF_AML_samples.csv")
cbf.samples <- GSEmatrix[GSEmatrix$title %in% CBF_AML_info$title, ]
```

2.1.2 Create NCBI query

```
search_ncbi <- paste(cbf.samples$SRX,collapse=" OR ")
search_ncbi
```

```
## [1] "SRX729580 OR SRX729581 OR SRX729582 OR SRX729583 OR SRX729584 OR SRX729585 OR SRX729586 OR SRX7
```

The search SRX729580 OR SRX729581 OR SRX729582 OR SRX729583 OR SRX729584 OR SRX729585 OR SRX729586 OR SRX729587 OR SRX729588 OR SRX729589 OR SRX729590 OR SRX729591 OR SRX729592 OR SRX729602 OR SRX729603 OR SRX729604 OR SRX729605 OR SRX729606 OR SRX729607 OR SRX729608 OR SRX729609 OR SRX729610 OR SRX729611 OR SRX729612 OR SRX729613 OR SRX729615 OR SRX729616 OR SRX729617 OR SRX729618 OR SRX729619 OR SRX729620 OR SRX729621 OR SRX729622 OR SRX729623 OR SRX729624 OR SRX729625 OR SRX729626 OR SRX729627 OR SRX729628 OR SRX729629 OR SRX729630 OR SRX729631 OR SRX729632 OR SRX729633 can be pasted into NCBI https://www.ncbi.nlm.nih.gov/sra and by following the intructions in https://www.ncbi.nlm.nih.gov/sra/docs/sradownload/#download-sequence-data-files-usi **Download sequence data files using SRA Toolkit** one can download all the `SRR` run names and information of the runs.

Sample SRX729580 is used as example here and it was run across different lanes whose samples are listed in ./data/SraAccList.txt. The sampels are downloaded with prefetch with the code below.

```
# The sra files are downloaded in your home directory under ~/ncbi/public/sra
prefetch --option-file ./data/SraAccList.txt
```

The SRA files are then converted to FASTQ files with fastq-dump --split-files using fastqc/0.11.8.

The sra files are downloaded by default into your home directory under ~/ncbi/public/sra. In the example below I converted the sra to fastq with fastq-dump, performed quality control with fastqc and the FASTQ files to ./data/test-fastq.

```
# Convert to Fastq
fastq-dump --split-files path_to_sra_files/*

# Multiqc to check quality of files
find . -name "SRR*.fastq" > fastq_files.txt
cat fastq_files.txt | parallel -j 10 "fastqc {}"

multiqc path_to_fastq_files/* --interactive -n "files-downloaded" -o path_to_fastq_files/*

mv path_to_fastq_files/*.fastq ./data/test-fastq/
```

2.2 Downsampling FASTQ files to a fixed number of reads

Each sample was sequenced over multiple lanes and the FASTQ files for one sample were first concatenated and then downsampled. In the example below only the the FASTQ files whose samples are defined in the second argument (./data/sample-to-merge.txt) are combined. The function needs to be run in the directory where the files are saved.

```
cd data/test-fastq
python ../../functions/combined_fastqs.py ../match-SRX-SRR.txt ../sample-to-merge.txt

fastqc SRX729580_combined_1.fastq
fastqc SRX729580_combined_2.fastq
multiqc . --interactive -n "files-downloaded-merged" -o .
```

The seqtk tool was then used to downsample an exact number of reads from paired end (PE) FASTQ files. In the code below we downsample the fastq file to 30M fragments and perform QC to check the final number of reads in each file.

```
path-to-seqtk-folder/seqtk sample -s100 SRX729580_combined_1.fastq 30000000 > sub30M_SRX729580_combined
path-to-seqtk-folder/seqtk sample -s100 SRX729580_combined_2.fastq 30000000 > sub30M_SRX729580_combined

# Multiqc to check number of reads in files
fastqc sub30M_SRX729580_combined_1.fastq
fastqc sub30M_SRX729580_combined_2.fastq
multiqc . --interactive -n "files-downsampled" -o .
```

Chapter 3

Align, call and standardise

3.1 Alignement, Read Groups, Mark duplicates

The *FASTQ* files were aligned with *STAR*. The index needed for STAR is created using the functions below after downloading the UCSC hg19 fasta reference genome and GTF file from iGenomes. For the index, STAR requires an extra parameter called `sjdbOverhang` which is usually set to be *(read length - 1)* epends on the read length used for sequencing, in this case 100bp - 1.

3.1.1 STAR index

```
module load STAR/2.5
module load R/3.5.2

# Iniaitlise Genome directory where to save STAR Index and STAR Fusion Index folders
# 99 = read length - 1 as suggested in STAR manual
star_genome100=path_to_genome_directory/star_index_hg19_99
genome=reference_genome_hg19.fasta
gtf=gtf_hg19.gtf

mkdir -p ${star_genome100}

STAR \
--runMode genomeGenerate \
--genomeDir ${star_genome100} \
--genomeFastaFiles ${genome} \
--sjdbOverhang 99 \
--sjdbGTFfile ${gtf}
```

From this point onwards the script `./functions/align-call-stand.sh` can be setup and run.

3.1.2 STAR-1pass

The 45 FASTQ files were aligned using STAR in two pass mode. First all samples are aligned in the first pass using the example function below, a wrapper function that calls STAR. The example is provided for one sample.

```
module load STAR/2.5
module load R/3.5.2

cd ../../

star_genome100=path_to_genome_directory/star_index_hg19_99
# compress fastq file
pigz data/test-fastq/sub*combined*.fastq
FQ1=data/test-fastq/sub30M_SRX729580_combined_1.fastq.gz
FQ2=data/test-fastq/sub30M_SRX729580_combined_2.fastq.gz
bamout=SRX729580
mkdir data/test-fastq/star-pass1

Rscript ./functions/run_STAR.R \
--genome_index ${star_genome100} \
--fastqfiles $FQ1,$FQ2 \
--sampleName SRX729580 \
--RlibPath "path_to_R/3.4" \
--outdir data/test-fastq/star-pass1 --STARmode "1Pass"

# Check alignment stats
multiqc data/test-fastq/star-pass1 --interactive -n "star1pass" -o data/test-fastq/star-pass1
```

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

```
# Version STAR/2.5
STAR --genomeDir $star_genome100 \
--readFilesIn $FQ1 $FQ2 \
--runThreadN 27 \
--chimSegmentMin 10 \
--readFilesCommand zcat \
--alignSJoverhangMin 8 \
--outBAMcompression 10 \
--alignSJDBoverhangMin 1 \
--limitBAMsortRAM 85741557872 \
--outFilterMismatchNmax 999 \
--alignIntronMin 20 \
--alignIntronMax 200000 \
--alignMatesGapMax 20000 \
--outFileNamePrefix data/test-fastq/star-pass1/SRX729580 \
--outSAMtype BAM SortedByCoordinate \
--outFilterType BySJout \
--outFilterMultimapNmax 15
```

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

```
# concatenate splice junctions from all samples from ran in pass1
cat data/test-fastq/star-pass1/*SJ.out.tab > data/test-fastq/star-pass1/combined_sj.out.tab
# Dobin suggests to remove chrm cause they are usually False positives
awk '!/chrM/' data/test-fastq/star-pass1/combined_sj.out.tab > data/test-fastq/star-pass1/combined_sj_n
```

3.1.3 STAR-2pass

The STAR call aligns and sorts reads by coordinate.

```
module load STAR/2.5
module load R/3.5.2

star_genome100=path_to_genome_directory/star_index_hg19_99
FQ1=data/test-fastq/sub30M_SRX729580_combined_1.fastq.gz
FQ2=data/test-fastq/sub30M_SRX729580_combined_2.fastq.gz
mkdir data/test-fastq/star-pass2

Rscript ./functions/run_STAR.R \
--genome_index $star_genome100 \
--fastqfiles $FQ1,$FQ2 \
--sampleName SRX729580 \
--RlibPath "path_to_R/3.4" \
--outdir data/test-fastq/star-pass2 --STARmode "2PassMulti" \
--sjfile data/test-fastq/star-pass1/combined_sj_nochrM.out.tab

# Check alignment stats
multiqc data/test-fastq/star-pass1 --interactive -n "star2pass" -o data/test-fastq/star-pass2
```

The function above is a wrapper for the STAR call used in the first pass with the additinal settings below:

```
--outFilterScoreMinOverLread 0.3 \
--chimSegmentReadGapMax 6 \
--alignSJstitchMismatchNmax 5 -1 5 5 \
--chimOutType WithinBAM \
--chimJunctionOverhangMin 2 \
--limitSjdbInsertNsj 2273673 \
--sjdbFileChrStartEnd data/test-fastq/star-pass1/combined_sj_nochrM.out.tab
```

3.1.4 Processing post alingment

These include:

1. Mark PCR duplicates, sambamba markdup from `sambamba/0.6.6` was used.
2. Add Read Groups using AddOrReplaceReadGroups (from `picard-tools/2.9.4`). Read groups are required by `GATK`.

If a sample was sequenced across different lanes lane-specific read groups are required for each separate bamfile (e.g. SampleName_L1, SampleName_L2). This sample name will be used for the fields `RGLB` and `RGSM` in the `AddOrReplaceReadGroups` example below.

```
module load picard-tools/2.9.4
# Picard tool function to add read groups to a bamfile
AddOrReplaceReadGroups \
I= data/test-fastq/star-pass2/SRX729580Aligned.sortedByCoord.out.bam \
O= data/test-fastq/star-pass2/SRX729580Aligned.sortedByCoord.out.RG.bam \
RGID=SRX729580 \
RGPU=SRX729580 \
RGLB=SRX729580_L1 \
RGPL="illumina" \
RGSM=SRX729580_L1
```

3. ValidateSamFile (from picard-tools/2.9.4) can be used to check for errors in the final bamfile.
4. `sambamba index` to index the final bamfile.

```
module load sambamba/0.6.6

sambamba index data/test-fastq/star-pass2/SRX729580Aligned.sortedByCoord.out.RG.bam
```

3.2 GATK pre-processing to call variants with MuTect and VarScan

The function `./functions/gatk_process_pipe.R` performs the following steps required by GATK3 to run MuTect2:

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

These steps are suggested in the GATK best practices for RNA-Seq variant calling.

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

```
module load gatk/3.7.0
module load sambamba/0.6.6
module load R/3.5.2

genome_fasta=reference_genome_hg19.fasta

Rscript ./functions/gatk_process_pipe.R \
--reference_fasta $genome_fasta \
--bamfile data/test-fastq/star-pass2/SRX729580Aligned.sortedByCoord.out.RG.bam \
--sampleName SRX729580 \
--knownSites1 path_to_GATK_Bundle_files/dbsnp_138.hg19.excluding_sites_after_129.vcf \
--knownSites2 path_to_GATK_Bundle_files/Mills_and_1000G_gold_standard.indels.hg19.sites.vcf \
--knownSites3 path_to_GATK_Bundle_files/1000G_phase1.indels.hg19.sites.vcf
```

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

3.2.1 SplitNCigarReads

```
gatk -T SplitNCigarReads -R path_to_genome.fa \
-I data/test-fastq/star-pass2/SRX729580Aligned.sortedByCoord.out.RG.bam \
-o data/test-fastq/star-pass2/SRX729580Aligned.reorderedDupl.rg.split.bam \
--filter_mismatching_base_and_quals -U ALLOW_N_CIGAR_READS -rf ReassignOneMappingQuality -RMQF 255 -RMQ
--log_to_file data/test-fastq/star-pass2/SRX729580_RG_DUPL_SPLIT_log
```

3.2.2 Base recalibration

Base recalibration using known sites downloaded from the GATK Bundle. More information about base recalibration can be found on GATK website.

```
module load gatk/3.7.0

gatk -T BaseRecalibrator -R path_to_genome.fa \
```

```
-I data/test-fastq/star-pass2/SRX729580Aligned.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 data/test-fastq/star-pass2/BaseQRecal/SRX729580/SRX729580_recal_data.table \
--log_to_file data/test-fastq/star-pass2/BaseQRecal/SRX729580/SRX729580_recal_step1_log

gatk -T BaseRecalibrator -R path_hg19_reference/genome.fa \
-I data/test-fastq/star-pass2/SRX729580Aligned.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 data/test-fastq/star-pass2/BaseQRecal/SRX729580/SampleName_recal_data.table \
-o data/test-fastq/star-pass2/BaseQRecal/SRX729580/SRX729580_post_recal_data.table \
--log_to_file data/test-fastq/star-pass2/BaseQRecal/SRX729580/SRX729580_recal_step2_log

gatk -T AnalyzeCovariates -R path_hg19_reference/genome.fa \
-before data/test-fastq/star-pass2/BaseQRecal/SRX729580/SSRX729580_recal_data.table \
-after data/test-fastq/star-pass2/BaseQRecal/SRX729580/SRX729580_post_recal_data.table \
-csv data/test-fastq/star-pass2/BaseQRecal/SRX729580/SSRX729580_recalibration_plots.csv \
-plots data/test-fastq/star-pass2/BaseQRecal/SRX729580/SRX729580_recalibration_plots.pdf \
--log_to_file data/test-fastq/star-pass2/BaseQRecal/SRX729580/SRX729580_recal_analyseCov_log

gatk -T PrintReads -R path_hg19_reference/genome.fa \
-I data/test-fastq/star-pass2/SRX729580Aligned.reorderedDupl.rg.split.bam \
-o data/test-fastq/star-pass2/SRX729580Recal.reorderedDupl.rg.split.bam \
-nct 8 -BQSR data/test-fastq/star-pass2/BaseQRecal/SRX729580/SRX729580_post_recal_data.table \
--log_to_file data/test-fastq/star-pass2/BaseQRecal/SRX729580/SRX729580_Log_recalibrated_bases
```

MultiQC was used to create summary after running pre-processing with GATK https://multiqc.info/docs/#gatk.

3.3 Variant calling

Variants are called on regions of interest corresponding to the gene bodies of the genes whose variants are reported in **Supplemental Table 3** of [1]. The `bed` file with genomic regions is `./data/target_regions.bed`.

Variants are called with `MuTect2`, `VarScan` and `VarDict` in tumour-only mode and annotated using the Variant Effect Preditor (VEP). The directory needed for variant calling with VarDict (`path_to_vardict/VarDict`) was downloaded from VarDict GitHub page https://github.com/AstraZeneca-NGS/VarDict.

The `./functions/call_variants.R` function is a wrapper to call and annotate variants with the callers listed above.

```
module load R/3.5.2 # needed for vardict
module load vardict/1.5.1
module load vcftools/0.1.13 # needed for variant parsing
module load ensembl-vep/89.0

# Vardict tumour-only call
Rscript ./functions/call_variants.R \
--reference_fasta path_hg19_reference/genome.fa \
--bamfile data/test-fastq/star-pass2/SRX729580Aligned.sortedByCoord.out.RG.split.bam \
```

```
--sampleName SRX729580 \
--regions ./data/target_regions.bed \
--genome_assembly 'GRCh37' \
--VarDict_dir path_to_vardict_directory \
--caller vardict \
--outputdir_suffix "caller_defaults" \
--VEPcall 'vep --dir_cache path_to_vep_cache/.vep --offline' \
--output_directory ./results/

# VarScan tumour-only call
module load R/3.5.2
module load varscan/2.3.9
module load vcftools/0.1.13 # needed for variant parsing
module load ensembl-vep/89.0

Rscript ./functions/call_variants.R \
--reference_fasta path_to_genome.fa \
--bamfile data/test-fastq/star-pass2/SRX729580Recal.reorderedDupl.rg.split.bam \
--sampleName SRX729580 \
--regions ./data/target_regions.bed \
--genome_assembly 'GRCh37' \
--caller varscan \
--outputdir_suffix "caller_defaults" \
--VEPcall 'vep --dir_cache path_to_vep_cache/.vep --offline' \
--output_directory ./results/


# MuTect tumour-only call
module load R/3.5.2
module load gatk/3.7.0
module load vcftools/0.1.13
module load ensembl-vep/89.0

Rscript ./functions/call_variants.R \
--reference_fasta path_to_genome.fa \
--bamfile data/test-fastq/star-pass2/SRX729580Recal.reorderedDupl.rg.split.bam \
--sampleName SRX729580 \
--regions ./data/target_regions.bed \
--genome_assembly 'GRCh37' \
--caller mutect \
--outputdir_suffix "caller_defaults" \
--VEPcall 'vep --dir_cache path_to_vep_cache/.vep --offline' \
--output_directory ./results/
```

The calls using the function above are wrappers for the calls done by each caller using the settings below.

3.3.1 MuTect2 settings

```
gatk -T MuTect2 -R path_to_genome.fa \
-I:tumor data/test-fastq/star-pass2/SRX729580Recal.reorderedDupl.rg.split.bam \
-L ./data/target_regions.bed \
-o ./results/mutect/regions_caller_defaults/SRX729580_germline_snvs_indels.vcf \
```

```
-log ./results/mutect/regions_caller_defaults/SRX729580_germline_snvs_indels_log
```

3.3.2 Samtools + VarScan2 settings

```
samtools mpileup --output-tags AD,ADF,ADR,DP,SP \
--fasta-ref path_to_genome.fa \
-l ./data/target_regions.bed data/test-fastq/star-pass2/SSRX729580Aligned.sortedByCoord.out.RG.split.re
```

3.3.3 VarDict settings

```
vardict -f 0.05 -c 1 -S 2 -E 3 -g 4 -r 2 -t -th 10 -v -G  path_to_genome.fa \
-b data/test-fastq/star-pass2/SRX729580Aligned.sortedByCoord.out.RG.split.bam ./data/target_regions.bed
```

3.3.4 VEP settings

```
module load ensembl-vep/89.0

vep --dir_cache dir_to_VEP_cache/.vep --offline \
-i ./results/vardict/regions_default-settings/SSRX729580_germline_snvs_indels.vcf \
-o ./results/vardict/regions_default-settings/annotated_variants/SSRX729580_germline_annotated.vcf \
--cache --everything --force_overwrite --assembly GRCh37 --fork 12 --vcf --port 3337
```

Chapter 4

Standardise variants output

The function `Rscript ./functions/call_variants.R` will return as output variants in a standardised format so that they can be easily compared across callers. This means that fields of interest, like variant allele frequency and depth at a variant site, which are usually provided in with different names across callers, are returned with the same names. The standardised variants are returned as `.txt` files. The functions in `call_variants.R` aimed at standardising variants are also the same ones present in the `varikondo`[2] R package available on github. Examples on how to use its functions are provided in the package website https://annaquaglieri16.github.io/varikondo/articles/vignette.html and below.

The earlier version of the functions returns some duplicated rows compared to using `varikondo` but this won't affect the results of the study as we only looked at unique variants.

The `.txt` files `SRX729580_germline_final.txt` produced for each caller are used for sensitivity analysis and they are processed as discussed in the Methods paper.

```
# devtools::install_github("annaquaglieri16/varikondo")

# detach("package:varikondo")
library(varikondo)
utils::packageVersion("varikondo")
```

```
## [1] '0.10.8'
```

```
# Variant file created with ./functions/call_variants.R
mutect.table <- read.delim("data/test-variants/mutect/regions_caller_defaults/annotated_variants/SRX729
dim(mutect.table)
```

```
## [1] 3005   82
```

```
sum(duplicated(mutect.table))
```

```
## [1] 2
```

```
# Read and parse with varikondo
mutect.varik <- varikondo::parse_vcf_output("data/test-variants/mutect/regions_caller_defaults/annotate
dim(mutect.varik)
```

```
## [1] 3003   87
```

```
# Variant file created with ./functions/call_variants.R
varscan.table <- read.delim("data/test-variants/varscan/regions_caller_defaults/annotated_variants/SRX7
dim(varscan.table)
```

```
## [1] 1031   82
```

```r
# Read and parse with varikondo
varscan.varik <- varikondo::parse_vcf_output("data/test-variants/varscan/regions_caller_defaults/annota
dim(varscan.varik)
```

```
## [1] 1031   87
```

```r
# VarDict file created with ./functions/call_variants.R
vardict.table <- read.delim("data/test-variants/vardict/regions_caller_defaults/annotated_variants/SRX7
vardict.table$key <- paste(vardict.table$Location,vardict.table$ref,
                           vardict.table$alt,vardict.table$Existing_variation,vardict.table$Consequence

dim(vardict.table)
```

```
## [1] 2799   84
```

```r
sum(duplicated(vardict.table))
```

```
## [1] 4
```

```r
# Read and parse with varikondo
vardict.varik <- varikondo::parse_vcf_output("data/test-variants/vardict/regions_caller_defaults/annota
dim(vardict.varik)
```

```
## [1] 2795   85
```

```r
vardict.varik$key <- paste(vardict.varik$Location,vardict.varik$ref,
                           vardict.varik$alt,vardict.varik$Existing_variation,vardict.varik$Consequence
```

```r
# All variants parsed with the previous function are also present when using varikondo
vardict.table$key[!(vardict.table$key %in% vardict.varik$key)]
```

```
## character(0)
```

Chapter 5

TCGA-LAML downsampling

1.24 is taken as the ratio between the mean proportion of mapped reads in the Leucegene and the TCGA-LAML cohorts. This is discussed in the Methods paper.

`TCGA-LAML-multiqc_samtools_flagstat.txt` was created using the `samtools flagstat` function.

```
library(tidyverse)

tcga <- read.delim("data/TCGA-LAML-multiqc_samtools_flagstat.txt")
info_tcga <- subset(tcga,select=c("Sample","mapped_passed","total_passed","total_failed","secondary_pass
  mutate(total_reads = (total_failed - secondary_failed) + (total_passed - secondary_passed),
         prop_mapped_passed = (mapped_passed - secondary_passed)/total_reads)

info_tcga$total_passed_no_secondary <- info_tcga$mapped_passed - info_tcga$secondary_passed
summary(info_tcga$total_passed_no_secondary/2/1000000) # divide by two as flagstats reports the number
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   36.70   54.98   58.68   57.72   61.53   69.62
```

```
tot_reads <- info_tcga %>%
  dplyr::select(Sample,total_reads) %>% # total_reads is the sum of both Reads1 and Reads2
  mutate(Prop40 = round((80000000/total_reads)*1.24,2)) %>%
  separate(Sample, into = c("Name"),sep="[.]") %>%   # 80000000 = target number of reads, double the fr
  dplyr::select(Name,Prop40,total_reads)

write_csv(tot_reads,"data/downsampling-proportion-TCGA-LAML.csv")
```

- Script to downsample at a certain proportion

```
module load picard-tools/2.9.4
module load samtools/1.6

DownsampleSam I=input.bam O=downsampled_output.bam PROBABILITY=sample_specific_prop RANDOM_SEED=200

# Used to check the size of the doensampled file
samtools flagstat downsampled_output.bam
```

Chapter 6

References

Bibliography

[1] 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).

[2] Anna Quaglieri and Christoffer Flensburg. *varikondo: an R package to standardise and integrate genetic variants across callers*. Unpublished.