**Appendix A**

The appendix contains the typical bash commands for command line bioinformatics software used in this thesis.

**MACS2 Peak-calling:**

#inputs and outputs:

INPUT=/path/to/input/control/bam/file.bam

IP=/path/to/IP/bam/file.bam

OUT=/path/to/output/dir

mkdir -p $OUT

TEMPDIR=/path/to/temp/dir

#Peak-calling:

macs2 callpeak -t $IP -c $INPUT -f BAM --keep-dup auto --outdir $OUT --tempdir $TEMPDIR --call-summits -n HIV1 -g 2.7e9 --verbose 3

**Bam QC:**

#inputs and outputs:

bam=$(ls /path/to/bam/directory/\*.bam | sed -n -e "$SGE\_TASK\_ID p")

export JAVA\_OPTS="-Djava.io.tmpdir=/path/to/temp/directory"

gtf=/path/to/gtf/annotation/file/hg38.gtf

out=/path/to/output/

outfile=$out${basename $bam}.html

counts=$out${basename $bam}.counts

out2=$out${basename $bam}.RNA\_seq\_metrics.txt

refflat=/path/to/refflatt/annotation/file

ribo\_coords=/path/to/ribosomal/coordinate/file.txt

#qualimap bam QC:

qualimap rnaseq -bam $bam -gtf $gtf -outdir $outfile -oc $counts -pe -a proportional

#picard rna seq metrics QC:

java -Xmx20G -jar $picard CollectRnaSeqMetrics \

I=$file \

O=$out2 \

REF\_FLAT=$refflat \

RIBOSOMAL\_INTERVALS=$ribo\_coords \

STRAND\_SPECIFICITY=NONE \

RRNA\_FRAGMENT\_PERCENTAGE=0.1

**Bam Sorting:**

#inputs and outputs:

infile=/path/to/input/bam/file.bam

outfile=/path/to/putput/bam/file.bam

#sort bam file:

 samtools sort -m 6G -@ 5 -o $outfile $infile

**Bam Indexing:**

#inputs and outputs:

bam=$(ls /path/to/bam/files /\*bam | sed -n -e "$SGE\_TASK\_ID p")

#indexing:

samtools index $bam

samtools idxstats $bam > $bam.indexStats.txt

**Cutadapt:**

#inputs and outputs:

read1=$(ls /path/to/fastq/files/\*R1\_001.fastq.gz | sed -n -e "$SGE\_TASK\_ID p")

read2=$(echo $read1 | sed 's/R1/R2/g')

fname=`basename $read1`

fname2=`basename $read2`

dir=`dirname $read1`

mkdir -p $dir/trimmed

trimmed\_read1=$dir/trimmed/$fname

trimmed\_read2=$dir/trimmed/$fname2

#trimming in paired-end mode:

cutadapt -q 10,10 -m 5 -a AGATCGGAAGAGC -A AGATCGGAAGAGC -o $trimmed\_read1 -p $trimmed\_read2 $read1 $read2

**Picard MarkDuplicates:**

#inputs and outputs:

bam=$(ls /path/to/bam/files/\*.bam | sed -n -e "$SGE\_TASK\_ID p")

basename=$( basename "$bam" )

dir=$(dirname "$bam")

outDir=$dir/duplicates\_marked/

mkdir -p $outDir

outfile=$outDir/$basename

metricsFile=$outfile.metrics.txt

#marking duplicates:

java -Xmx19G -Xms1G -jar /nobackup/umaan/software/bin/picard.jar MarkDuplicates TAGGING\_POLICY=All VALIDATION\_STRINGENCY=LENIENT I=$bam M=$metricsFile TMP\_DIR=/nobackup/umaan/temp O=$outfile

**FastQC:**

#inputs and outputs:

out=/path/to/fastQC/reports/directory/

mkdir -p $out

in=/path/to/fastq/directory/

temp=/path/to/FastQC/temp/directory

#Run QC:

 fastqc -o $out --dir $temp $in/\*.fastq.gz

**STAR Genome Index:**

#inputs and outputs:

fasta=/path/to/hg38/and/KSHV/fasta.fa

fastaDir=/path/to/index/output/directory

#Generate combined genome index:

STAR --runMode genomeGenerate --genomeFastaFiles $fasta --genomeDir $fastaDir --runThreadN 10

**STAR Alignment:**

#inputs and outputs:

fastaDir=/path/to/STAR/index

read1=$(ls /path/to/fastqs/\*R1\_001.fastq.gz | sed -n -e "$SGE\_TASK\_ID p")

read2=$(echo $read1 | sed 's/R1/R2/g')

fname=`basename $read1`

dir=`dirname $read1`

outdir=/path/to/output/directory/

mkdir -p $outdir

prefix=$outdir/$fname

gtf=/path/to/gtf/file.gtf

#Alignment, in this case 151 bp paired-end reads:

 STAR --runMode alignReads --genomeDir $fastaDir --runThreadN 10 --readFilesIn $read1 $read2 \

 --outFilterMismatchNoverLmax 0.05 --outFileNamePrefix $prefix --outSAMtype BAM SortedByCoordinate \

 --outFilterMultimapNmax 50 \

 --sjdbGTFfile $gtf --sjdbOverhang 150 \

 --outFilterMatchNminOverLread 0.5 --outFilterScoreMinOverLread 0.5 \

 --readFilesCommand zcat --outSAMattributes All

**SPLADDER:**

#Inputs and outputs:

GTF=/path/to/GTF/file.gtf

BAMS=/paths/to/bam/files.bam

OUT=/path/to/output/directory

#Splicing calls:

python /nobackup/umaan/software/bin/spladder/spladder.py -a $GTF -b $BAMS -o $OUT --ignore\_mismatches=y -T n

#Differential splicing calls, in this case for KO vs Scramble:

python /nobackup/umaan/software/bin/spladder/spladder\_viz.py -o $OUT -b $BAMS -L KO,Scramble -f png -t exon\_skip,intron\_retention,alt\_3prime,alt\_5prime,mult\_exon\_skip