**# Build custom database for Kraken2**

**kraken2-build** --download-taxonomy --db $DBNAME

**kraken2-build -**-add-to-library strain.fa --db $DBNAME

**kraken2-build** --build --db $DBNAME

**# Run Kraken2 to identify strains**

**kraken2 –**db $ DBNAME –report result.txt $sample\_1.fastq $sample\_2.fastq

**# Build custom database for Krakenuniq**

**krakenuniq-build** --db Krakenuniq\_DB --taxids-for-genomes --taxids-for-sequences

**# Run Krakenuniq to identify strains**

**krakenuniq** --db Kraken\_DB --threads 8 --fastq-input --paired $sample\_1.fastq $sample\_2.fastq --report\_file result.tsv> result.report

**# Build custom database for Centrifuge**

**centrifuge-build** --conversion-table $prefix.conv --taxonomy-tree $prefix.tree --name-table $prefix.name $prefix.fasta $prefix

**# Run Centrifuge to identify strains**

**centrifuge** -x $prefix -1 $sample\_1.fastq -2 $sample\_2.fastq --report\_file report.txt -S result.txt -p 8

**# Build custom database for Sigma**

**sigma-index-genomes** -c config.cfg -p 8

**# Run Sigma to identify strains**

**sigma-align-reads** -c config.cfg -p 8

**sigma** -c config.cfg -t 8

**# Run Sigma to identify strains**

**python pathoscope2.py** MAP -1 $sample\_1.fastq -2 $sample\_2.fastq -targetRefFiles Ref.fasta -filterRefFiles Human.fasta -outDir Results -outAlign out.sam -expTag tutorial

**python pathoscope2.py** ID -alignFile Results/out.sam -fileType sam -outDir Results -expTag tutorial

**# Run VirStrain to identify strains**

**python VirStrain.py** -d VirStrain\_DB -i $sample\_1.fastq -p $sample\_2.fastq -o Results

**# Build MegaBLAST index for strain genomes**

**makeblastdb** -in strains.fasta -out $INDEX -dbtype nucl

**# Run MegaBLAST to find the closest relative**

**blastn** -query strain.fasta -db $INDEX -outfmt "7 qseqid sseqid pident length mismatch evalue" -out Blast\_Out

**# Assembly for Washington data**

**megahit** -r $sample.fastq -o assembly\_result

**# Run bowtie2 to align reads to reference genome obtained by megahit**

**bowtie2-build** reference.fasta reference

**bowtie2** -p 6 -x reference -U reads.fastq -N 0 --no-unal --end-to-end -S result.sam

**# Assembly for 4 real data sets using Megahit (Supplementary Section 2.7)**

**megahit** -1 $sample\_1.fastq -2 $sample\_2.fastq -o assembly\_result

**# Assembly for 4 real data sets using Spades (Supplementary Section 2.7)**

**sapdes.py** --meta -1 $sample\_1.fastq -2 $sample\_2.fastq -o assembly\_result

**# Assembly for 4 real data sets using IVA (Supplementary Section 2.7)**

**iva** -f $sample\_1.fastq -r $sample\_2.fastq assembly\_result

- The reference sequences of different viruses for all haplotype reconstruction tools

|  |  |
| --- | --- |
| Viruses | Reference (NCBI accession IDs) |
| SARS-CoV-2 | NC\_045512.2 |
| H1N1 | J02176.1 |
| HIV | K03455.1 |
| HBV | MT622522.1 |
| HCMV | NC\_006273.2 |

**# Run BWA and Samtools to obtain sorted bam file**

**bwa** mem reference $sample\_1.fastq $sample\_2.fastq > result.sam

**samtools** view -b -T reference -o result.bam result.sam

**samtools** sort result.bam -o sorted.bam

**# Run ShoRAH to reconstruct haplotypes**

**shorah** shotgun -b sorted.bam -f reference

**# Run PredictHaplo to reconstruct haplotypes**

**predicthaplo** --sam result.sam --reference reference --have\_true\_haplotypes 0 --nSample 2

**# Run HaROLD to reconstruct haplotypes**

**bwa** mem reference $sample\_1.fastq $sample\_2.fastq > result.sam

**samtools** view -S -b result.sam > result.bam

**samtools** view -h -G69 result.bam | samtools view -h -G133 > file.bam

java -cp HaROLD/lib/htsjdk-unspecified-SNAPSHOT.jar:\

HaROLD/lib/picocli-4.1.2.jar:lib/pal-1.5.1.jar:\

HaROLD/lib/cache2k-all-1.0.2.Final.jar:\

HaROLD/lib/commons-math3-3.6.1.jar:\

HaROLD/jar/MakeReadCount.jar \

makereadcount.MakeReadCount file.bam

java -jar HaROLD/jar/Cluster\_RG/dist/Cluster\_RG.jar \

--count-file sample.txt --haplotypes 2 --alpha-frac 0.5 --gamma-cache 10000 \

-H -L --threads 8 -p Step1\_results

java -cp HaROLD/lib/htsjdk-unspecified-SNAPSHOT.jar:\

HaROLD/lib/picocli-4.1.2.jar:\

HaROLD/lib/pal-1.5.1.jar:\

HaROLD/lib/commons-math3-3.6.1.jar:\

HaROLD/lib/cache2k-all-1.0.2.Final.jar:\

HaROLD/lib/flanagan.jar:\

HaROLD/jar/RefineHaplotypes.jar refineHaplotypes.RefineHaplotypes \

-t sample2 --bam file.bam \

--baseFreq Step1\_results.lld --refSequence reference \

**# Run aBayesQR to reconstruct haplotypes**

aBayesQR config

config:

filename of reference sequence (FASTA) : reference

filname of the aligned reads (sam format) : result.sam

paired-end (1 = true, 0 = false) : 1

SNV\_thres : 0.05

reconstruction\_start : 1

reconstruction\_stop: $reference\_length

min\_mapping\_qual : 10

min\_read\_length : 80

max\_insert\_length : 250

characteristic zone name : reconstructed\_result

seq\_err (assumed sequencing error rate(%)) : 0.1

MEC improvement threshold : 0.0395

**# Run TenSQR to reconstruct haplotypes**

./ExtractMatrix config

./TenSQR.py config

config:

filename of reference sequence (FASTA) : reference

filname of the aligned reads (sam format) : result.sam

SNV\_thres : 0.05

reconstruction\_start : 1

reconstruction\_stop: $reference\_length

min\_mapping\_qual : 10

min\_read\_length : 80

max\_insert\_length : 250

characteristic zone name : reconstructed\_result

seq\_err (assumed sequencing error rate(%)) : 0.1

MEC improvement threshold : 0.0312

initial population size : 2

**# Run CliqueSNV to reconstruct haplotypes**

java -Xmx100G -jar ./clique-snv.jar -m snv-illumina -thread 8 -tl 604800 -tf 0.05 -in result.sam -outDir reconstructed\_result