Whole Exome Pipeline

Copying the files to my directory:

```
cp -R /mnt/gkhazen/NGS-Fall2020/FinalProject/* .
zcat 392_1.fastq.gz | more
```

Number of lines in fastq file:

```
zcat 392_1.fastq.gz | wc -l
#output:
123580348
#reads=30895087

zcat 392_2.fastq.gz | wc -l
#output:
123580348
#reads=30895087
```

Reference chromosome:

```
wget
https://hgdownload.soe.ucsc.edu/goldenPath/hg38/chromosomes/chr13.fa.g
z
```

Running FastQC

```
time fastqc -o FastQCResults 392_1.fastq.gz  392_2.fastq.gz

#output
real    12m33.402s
user    12m24.276s
sys     0m14.858s
```

Copy html report files into my machine:

```
scp -r
pia.chouaifaty@linuxdev.accbyblos.lau.edu.lb:FunctionalFinalProject/Fa
stQCResults /Users/piachouaifaty
```

FastQC Results:
The scores are encoded using Illumina 1.9 (BASE-33)

The FastQC report showed the presence of Illumina Universal Adapters.
After looking it up, I found the following adapter sequences that FastQC checks for:

Illumina Universal Adapter	AGATCGGAAGAG
Illumina Small RNA 3' Adapter	TGGAATTCTCGG
Illumina Small RNA 5' Adapter	GATCGTCGGACT
Nextera Transposase Sequence	CTGTCTCTTATA
SOLID Small RNA Adapter	CGCCTTGGCCGT

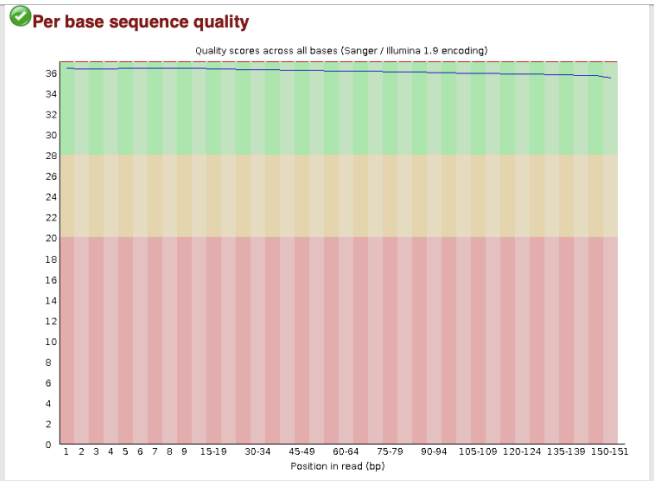
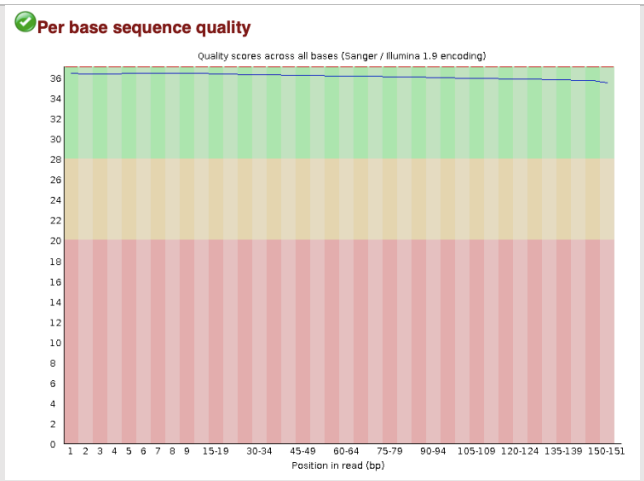
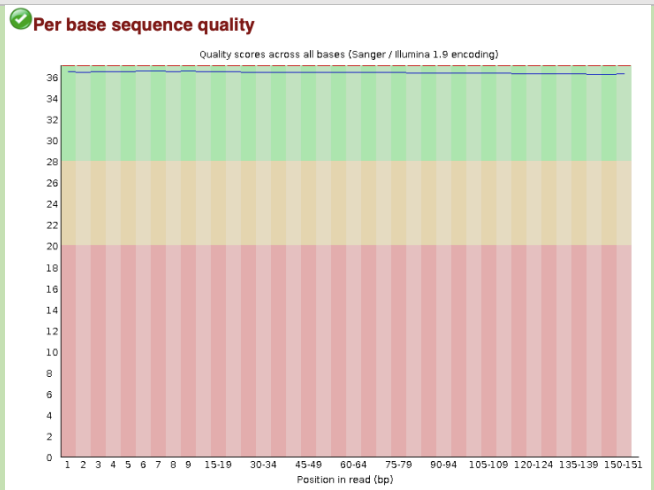
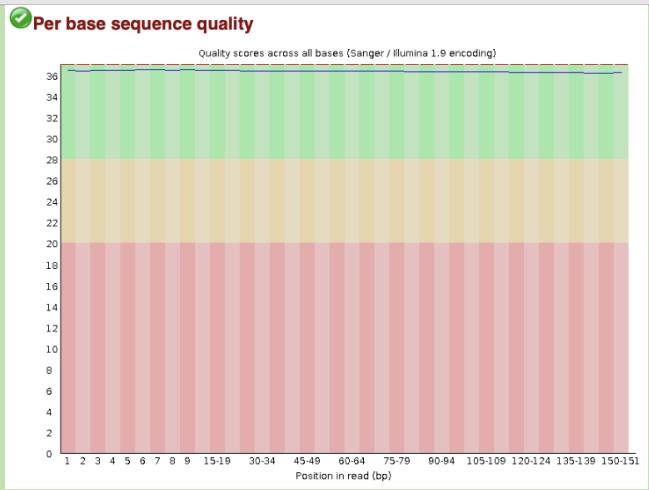
I checked the adapter folder in trimmomatic, and cross-referenced the Nextera sequence above to the Nextera adapter file, and found it to match. So, I checked each of the TruSeq PE files for the presence of AGATCGGAAGAG
Both TruSeq2-PE.fa and TruSeq3-PE-2.fa contain it, but TruSeq2 didn't remove the adapters so I decided to go for TruSeq3-PE-2.fa, which removed (most) of the adapters, except very few at the very end.
The FastQC results are compared in a table below.

Trimming step to trim adapters

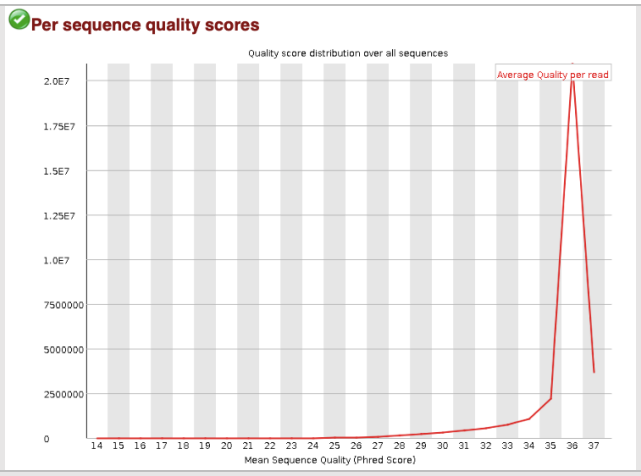
```
java -jar NGS/trim/Trimmomatic-0.39/trimmomatic-0.39.jar PE \
-threads 8 \
-trimlog ./FunctionalFinalProject/392.log \
./FunctionalFinalProject/392_1.fastq.gz \
./FunctionalFinalProject/392_2.fastq.gz \
./FunctionalFinalProject/392_1_trimmed_R1_paired.fastq.gz \
./FunctionalFinalProject/392_1_trimmed_R1_unpaired.fastq.gz \
./FunctionalFinalProject/392_2_trimmed_R2_paired.fastq.gz \
./FunctionalFinalProject/392_2_trimmed_R2_unpaired.fastq.gz \
ILLUMINACLIP:NGS/trim/Trimmomatic-0.39/adapters/TruSeq3-PE-
2.fa:2:30:10 \
LEADING:3 \
TRAILING:3 \
SLIDINGWINDOW:4:20 \
MINLEN:36
```

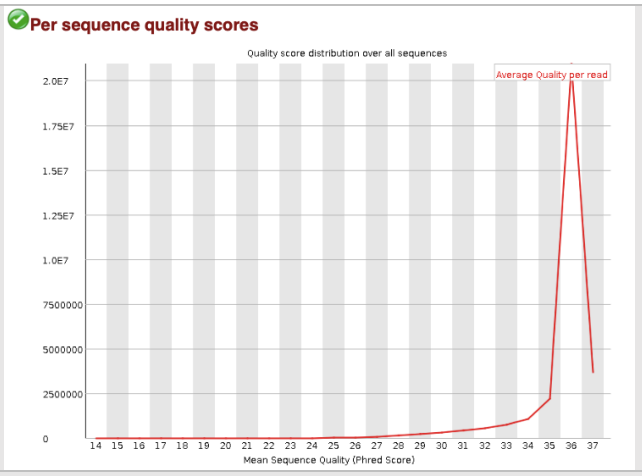
I kept them as recommended but increased the average score for the sliding window to 20 since most reads have a very high score.
After trimming, the length of reads became between 36-151 because of the MINLEN parameter.

FastQC Plots Pre and Post Trimming

	Read 1	Read 2
Pre-Trimming		
Both reads have great per base sequence quality. The medians all have Phred scores way above 30 (most are round 36), with a very slight drop towards the end, which is to be expected. No boxplots are apparent in the graph, probably because the variation in scores per position is extremely small.		
Post-Trimming		
The scores are even better after trimming.		

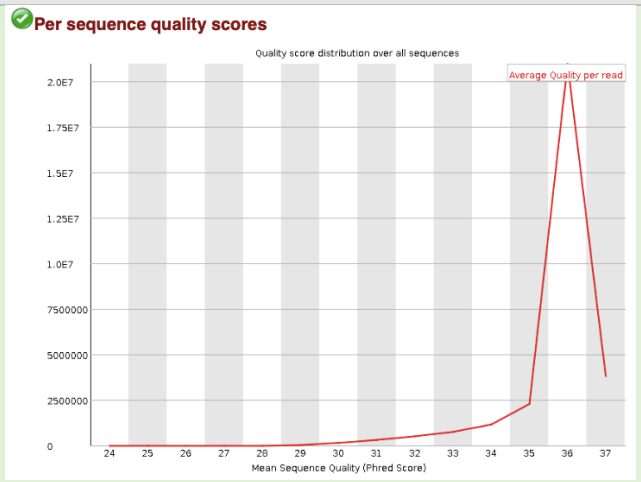
Pre-Trimming

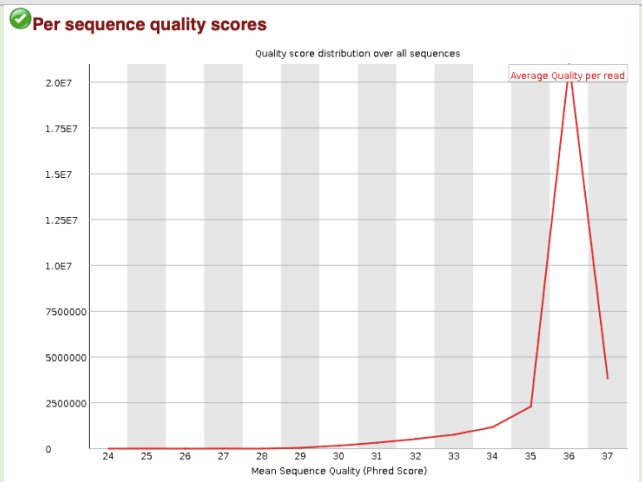




The overwhelming majority of reads have QS above 30. One single peak towards the end indicates that there are no groups of reads with bad scores that need to be eliminated.

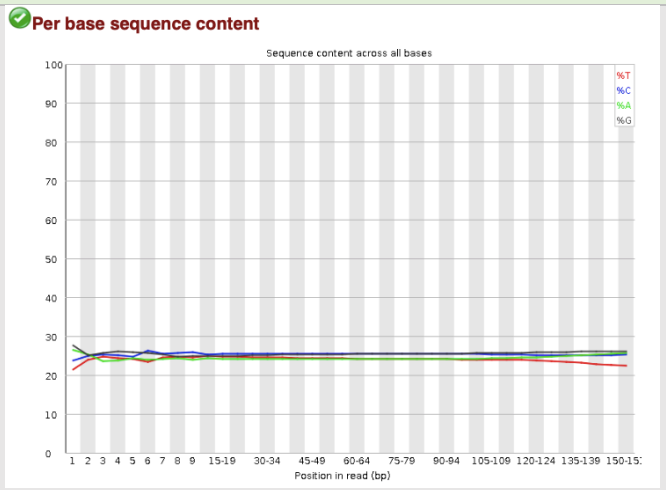
Post-Trimming

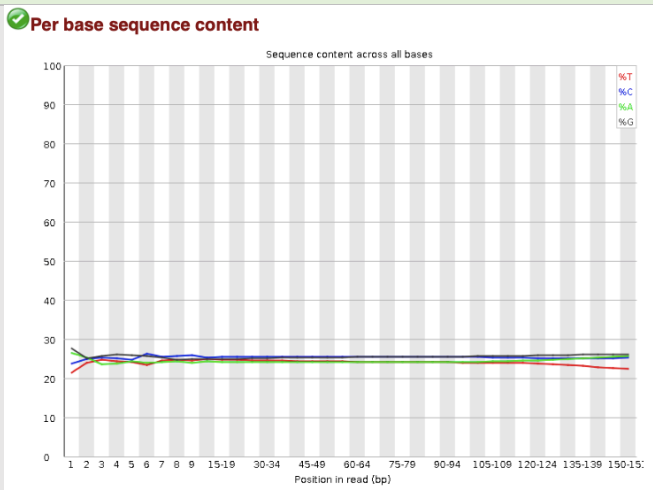




After trimming, even more sequences have scores above 30 and the number of sequences with scores below 30 greatly reduces.

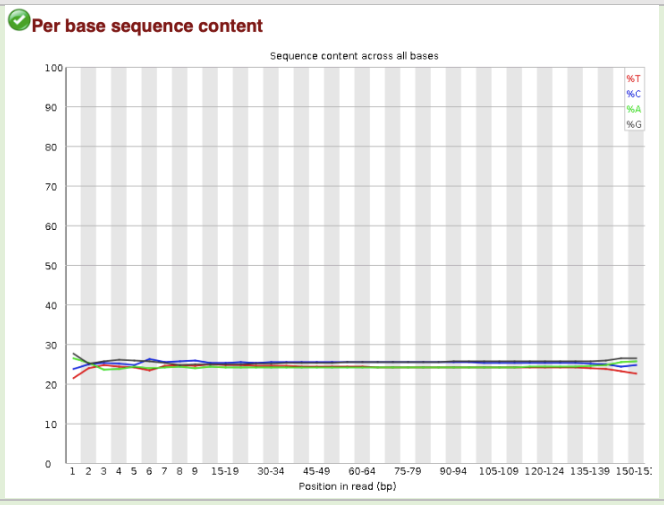
Pre-Trimming

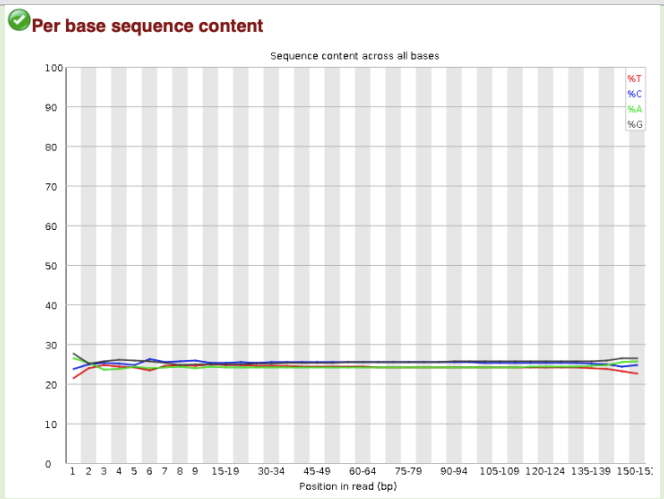




The distribution of base pairs is more or less equal, except for a lower percentage of T's towards the end, but that may be fixed after trimming. The discrepancy at the beginning is normal due to primers.

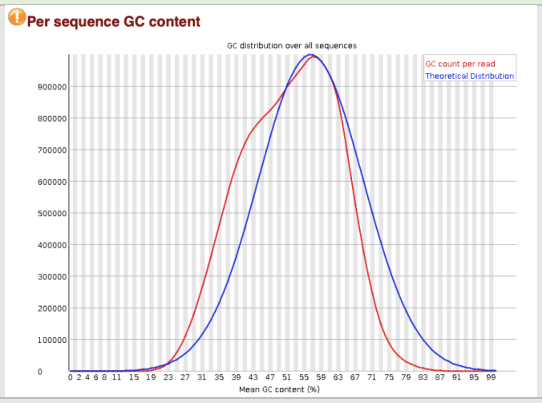
Post-Trimming

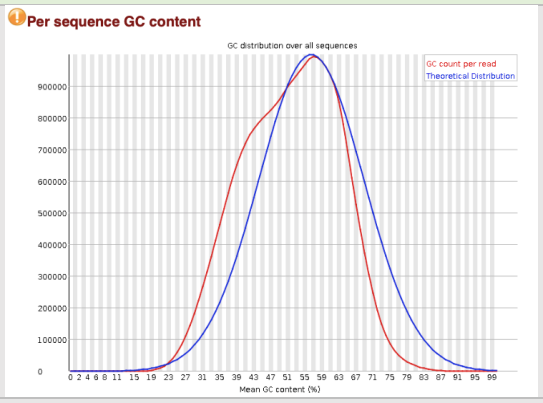




Doesn't change much.

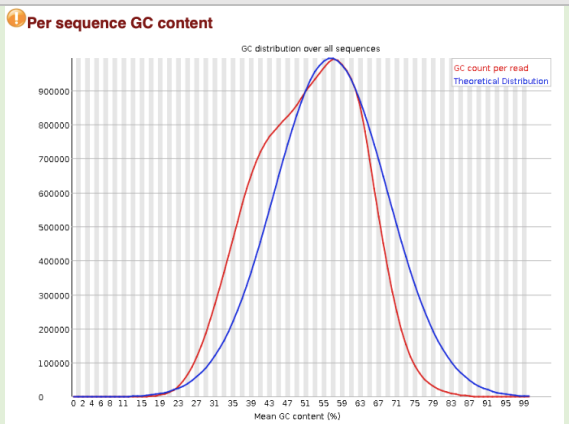
Pre-Trimming

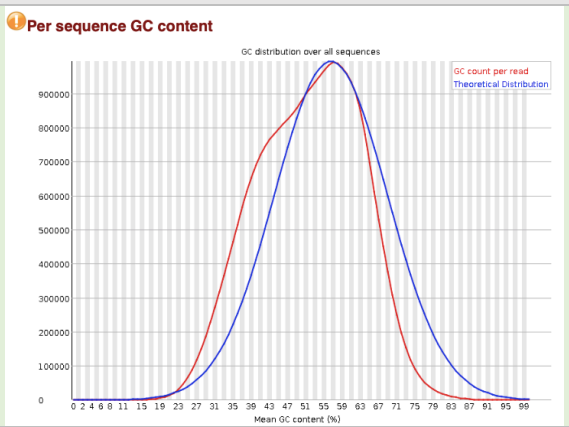




Slight overoccurence (normal since exome sequencing, and in exons, GC content is higher)

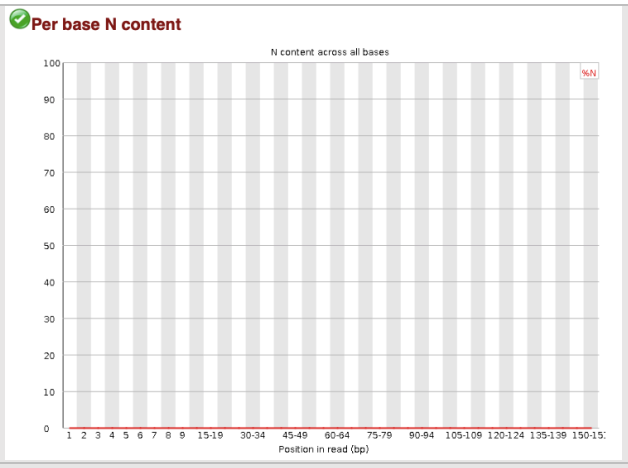
Post-Trimming

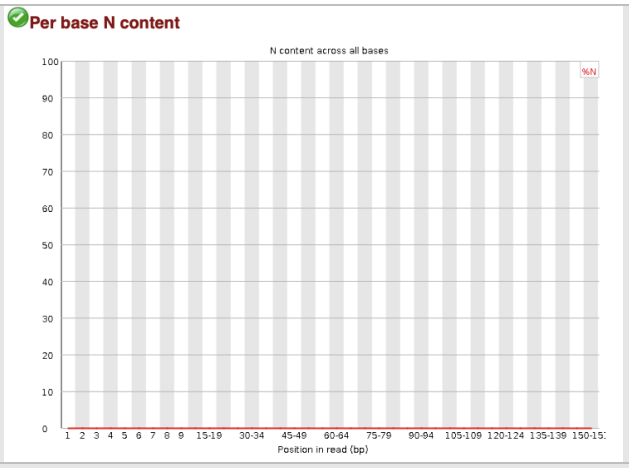




No change

Pre-Trimming



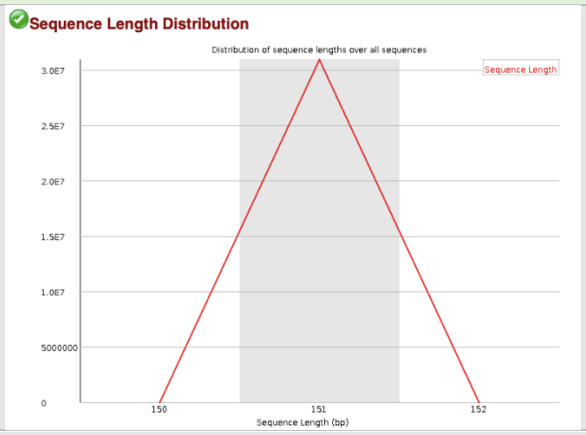


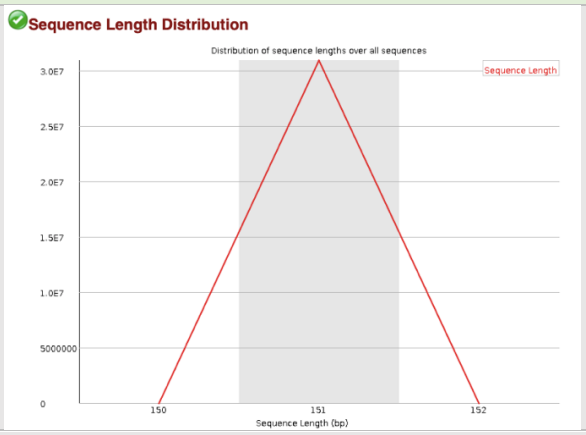
There are no unknown nucleotides.

Post-Trimming

Unchanged

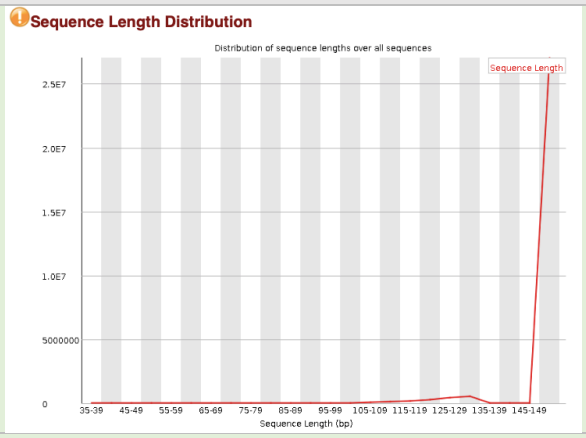
Pre-Trimming

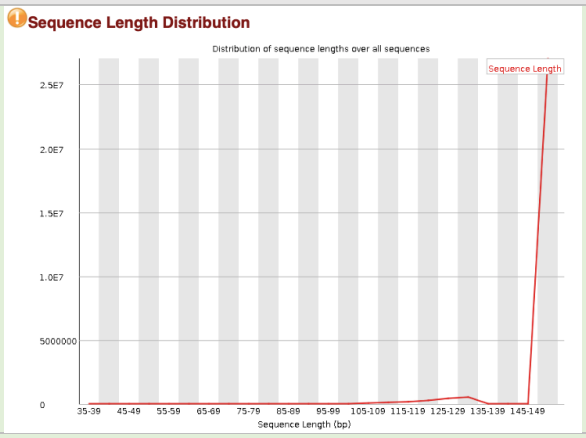




Sequence length are all 151, only one peak.

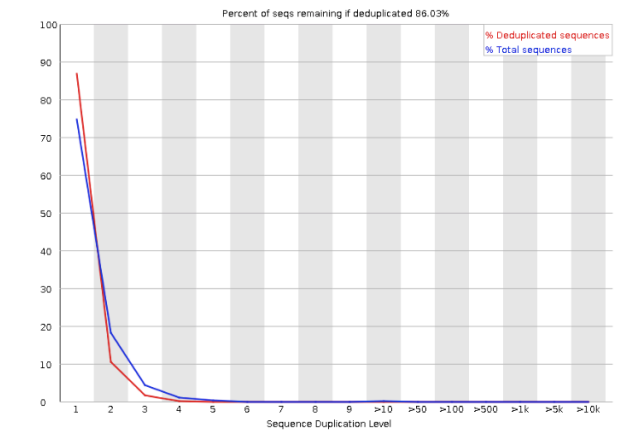
Post-Trimming

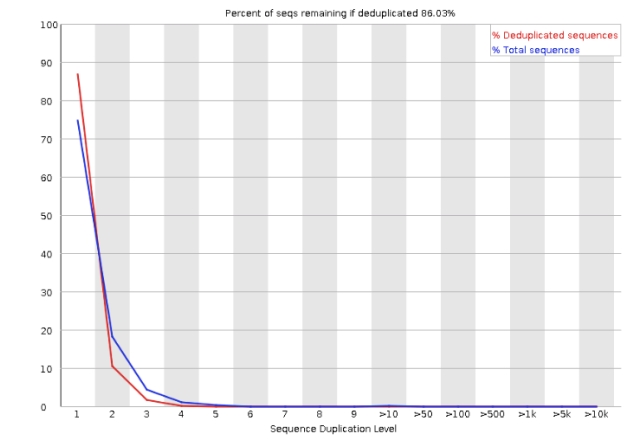




Small peak at around 115-135, but very small, not significant, and it is due to trimming. Much higher peak at 151

Pre-Trimming



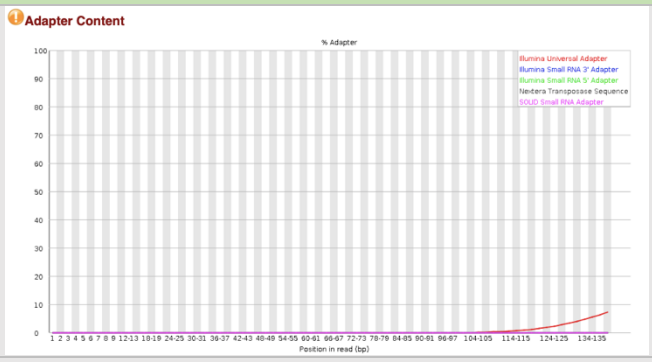


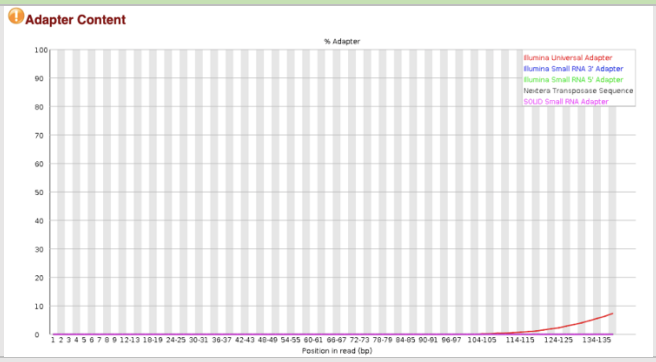
No sequences are replicated more than around 3-4 times
Also no overrepresented sequences

Post-Trimming

No change

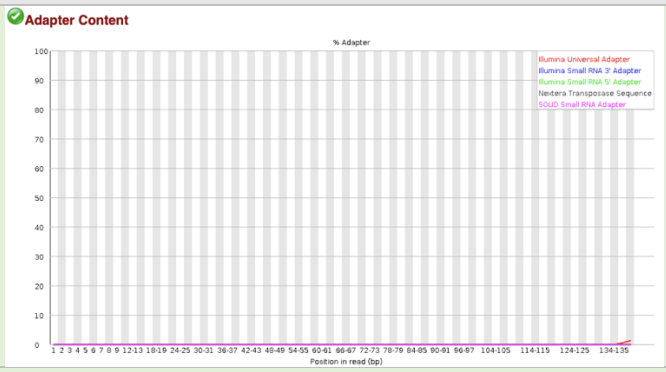
Pre-Trimming

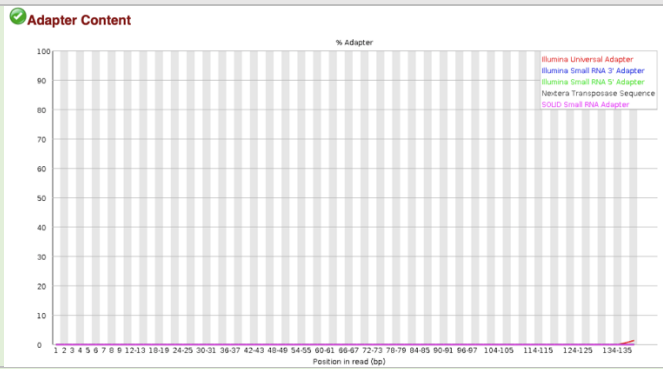




Some of the sequences have Illumina Universal Adapters. Will be removed by trimming

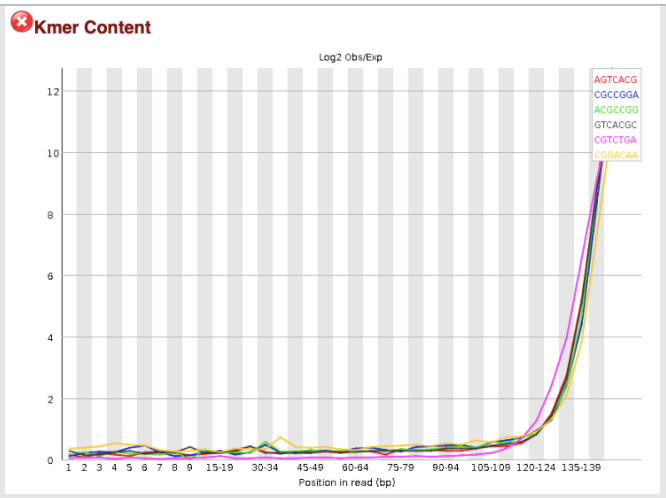
Post-Trimming

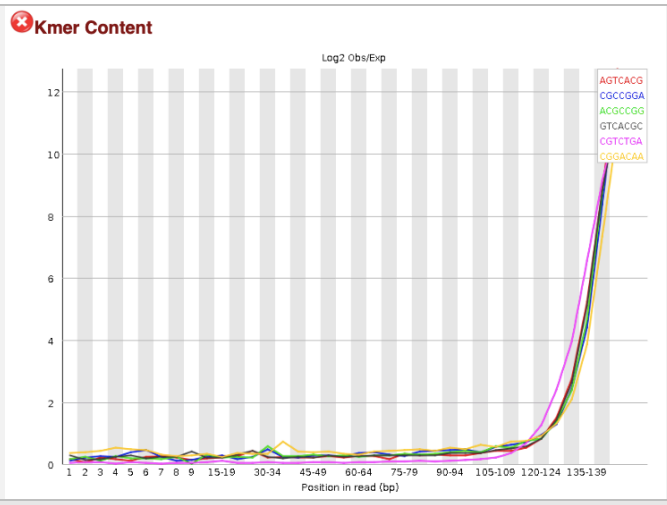




Almost all adapters removed, very very small peak towards the end.

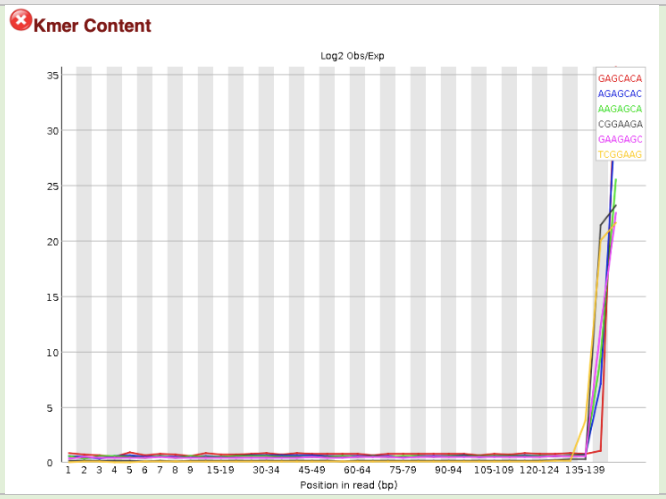
Pre-Trimming

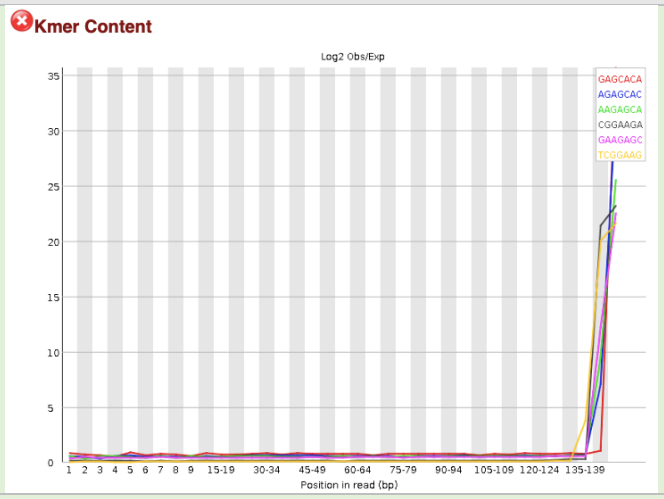




Some K-mers are repeated towards the end

Post-Trimming





There are no tiles with bad quality.

Fastq Processing Scripts

Length of Remaining Reads after Trimming
Script: Find_Read_Lengths.R
Takes trimmed fastq file path as input
Call: Rscript Find_Read_Lengths.R test1.fastq

```
#!/usr/bin/env Rscript

#Rscript Find_Read_Lengths.R path

if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

if (!requireNamespace("ShortRead", quietly = TRUE))
  BiocManager::install("ShortRead")

library(ShortRead)

#Find_Read_Lengths
main = function(fstq_file_path)
{
  fstq_file=readFastq(fstq_file_path)
  l=length(sread(fstq_file)) #number of reads
  vectlen=c()

  for (i in 1:l)
  {
    seq_vect=as.vector(sread(fstq_file)[i]) #the sequence as a vector
    slen=nchar(seq_vect) #length of the vector
    vectlen=append(vectlen, slen)
  }

  print(paste0("The lengths after trimming (unique): "))
  return(sort(unique(vectlen)))
}

args = commandArgs(trailingOnly = TRUE)
fstq_file_path = args[1]

main(fstq_file_path)
```

```
[1] "The lengths after trimming (unique): "
 [1]  47  69  73  86  89  95 104 107 115 117 121 123 124 125 127 128 129 130 131
[20] 132 133 141 150 151
```

Maximum Average Read Phred Score + IDS
Script: Max_Score_IDs.R
Takes fastq file path as input
Call: Rscript Max_Score_IDs.R test1.fastq

```
#!/usr/bin/env Rscript

#Rscript Max_Score_IDs.R path

if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")
if (!requireNamespace("ShortRead", quietly = TRUE))
  BiocManager::install("ShortRead")

library(ShortRead)

#Max_Score_IDs

main = function(fstq_file_path)

{
  vectmeans=c() #empty vector that will contain average score for
each read
  fstq_file=readFastq(fstq_file_path)
  n=length(sread(fstq_file)) #number of reads

  for (i in 1:n) #for each read
  {
    seq_vect=as.vector(sread(fstq_file)[i]) #the sequence as a vector
    slen=nchar(seq_vect) #length of the vector
    qual=as(quality(fstq_file)[i], "matrix")[,1:slen] #qualities of
bases as a vector
    vectmeans=append(vectmeans, mean(qual)) #mean qs of read, add to
mean quality vector
  }

  #which.max(vectmeans) #get index of read with highest average
quality

  idxmax=which.max(vectmeans)

  max_score=vectmeans[idxmax] #maximum score

  print(paste0("Max Average quality: ", max_score))
```

```
  idxmaxes=which(vectmeans==max_score) #indeces of reads with the
maximum score

  print("Indeces of reads with max average quality: ")
  print(idxmaxes)
  print(paste0("Read ID: ", as.vector(id(fstq_file)[c(idxmaxes)])))
#headers of reads with max score
}

args = commandArgs(trailingOnly = TRUE)
fstq_file_path = args[1]

main(fstq_file_path)
```

```
[1] "Max Average quality: 37"
[1] "Indeces of reads with max average quality: "
 [1]  44  45  57  62  66  81  85 109 125 128 136 152 160 167 170 185 189 195 197
[20] 227 235 239 246
 [1] "Read ID: A00721:81:HNLHYDSXX:1:1101:10972:1047 1:N:0:GCCGGACA+TGTAAGAG"
 [2] "Read ID: A00721:81:HNLHYDSXX:1:1101:13955:1047 1:N:0:GCCGGACA+TGTAAGAG"
 [3] "Read ID: A00721:81:HNLHYDSXX:1:1101:19714:1063 1:N:0:GCCGGACA+TGTAAGAG"
 [4] "Read ID: A00721:81:HNLHYDSXX:1:1101:30092:1063 1:N:0:GCCGGACA+TGTAAGAG"
 [5] "Read ID: A00721:81:HNLHYDSXX:1:1101:18276:1078 1:N:0:GCCGGACA+TGTAAGAG"
 [6] "Read ID: A00721:81:HNLHYDSXX:1:1101:14724:1094 1:N:0:GCCGGACA+TGTAAGAG"
 [7] "Read ID: A00721:81:HNLHYDSXX:1:1101:25735:1094 1:N:0:GCCGGACA+TGTAAGAG"
 [8] "Read ID: A00721:81:HNLHYDSXX:1:1101:22896:1125 1:N:0:GCCGGACA+TGTAAGAG"
 [9] "Read ID: A00721:81:HNLHYDSXX:1:1101:31168:1141 1:N:0:GCCGGACA+TGTAAGAG"
[10] "Read ID: A00721:81:HNLHYDSXX:1:1101:5339:1157 1:N:0:GCCGGACA+TGTAAGAG"
[11] "Read ID: A00721:81:HNLHYDSXX:1:1101:21160:1157 1:N:0:GCCGGACA+TGTAAGAG"
[12] "Read ID: A00721:81:HNLHYDSXX:1:1101:20157:1172 1:N:0:GCCGGACA+TGTAAGAG"
[13] "Read ID: A00721:81:HNLHYDSXX:1:1101:3947:1188 1:N:0:GCCGGACA+TGTAAGAG"
[14] "Read ID: A00721:81:HNLHYDSXX:1:1101:24957:1188 1:N:0:GCCGGACA+TGTAAGAG"
[15] "Read ID: A00721:81:HNLHYDSXX:1:1101:30960:1188 1:N:0:GCCGGACA+TGTAAGAG"
[16] "Read ID: A00721:81:HNLHYDSXX:1:1101:24786:1204 1:N:0:GCCGGACA+TGTAAGAG"
[17] "Read ID: A00721:81:HNLHYDSXX:1:1101:30969:1204 1:N:0:GCCGGACA+TGTAAGAG"
[18] "Read ID: A00721:81:HNLHYDSXX:1:1101:9136:1219 1:N:0:GCCGGACA+TGTAAGAG"
[19] "Read ID: A00721:81:HNLHYDSXX:1:1101:12915:1219 1:N:0:GCCGGACA+TGTAAGAG"
[20] "Read ID: A00721:81:HNLHYDSXX:1:1101:18900:1251 1:N:0:GCCGGACA+TGTAAGAG"
[21] "Read ID: A00721:81:HNLHYDSXX:1:1101:16179:1266 1:N:0:GCCGGACA+TGTAAGAG"
[22] "Read ID: A00721:81:HNLHYDSXX:1:1101:31204:1266 1:N:0:GCCGGACA+TGTAAGAG"
[23] "Read ID: A00721:81:HNLHYDSXX:1:1101:11053:1282 1:N:0:GCCGGACA+TGTAAGAG"
```

IDs of the 10 Shortest Reads
Script: Min_10_Length_IDs.R
Takes fastq file path as input
Call: Rscript Min_10_Length_IDs.R test1.fastq

```
#!/usr/bin/env Rscript

#Rscript Min_10_Length_IDs.R path

if (!requireNamespace("BiocManager", quietly = TRUE))
  install.packages("BiocManager")

if (!requireNamespace("ShortRead", quietly = TRUE))
  BiocManager::install("ShortRead")

library(ShortRead)

#Min_10_Length_IDs
main = function(fstq_file_path)
{
  vectlen=c()
  fstq_file=readFastq(fstq_file_path)
  n=length(sread(fstq_file)) #number of reads

  for (i in 1:n) #for each read
  {
    seq_vect=as.vector(sread(fstq_file)[i]) #the sequence as a vector
    slen=nchar(seq_vect) #length of the vector
    vectlen=append(vectlen, slen) #add length to vector of lengths
  }

  vectlen=cbind(vectlen, 1:n) #matrix with length and index of reads
  colnames(vectlen)=c("length", "index")

  srtd=vectlen[order(vectlen[,"length"]),] #sort matrix by length
  min10=srtd[1:10,] #10 shortest
  min10idx=min10["index"] #indeces of 10 shortes
  min10ids=as.vector(id(fstq_file)[c(min10idx)]) #ids of 10 shortest

  min10=cbind(min10, min10ids)
  colnames(min10)=c("length", "index", "id")
  print("10 Shortest Reads")
  print(min10)
}
```

```
args = commandArgs(trailingOnly = TRUE)
fstq_file_path = args[1]

main(fstq_file_path)
```

```
[1] "10 Shortest Reads"
      length index
 [1,] "47"   "201"
 [2,] "69"   "65"
 [3,] "73"   "166"
 [4,] "86"   "18"
 [5,] "89"   "54"
 [6,] "95"   "69"
 [7,] "104"  "154"
 [8,] "107"  "27"
 [9,] "115"  "171"
[10,] "117"  "73"
      id
 [1,] "A00721:81:HNLHYDSXX:1:1101:16441:1219 1:N:0:GCCGGACA+TGTAAGAG"
 [2,] "A00721:81:HNLHYDSXX:1:1101:16613:1078 1:N:0:GCCGGACA+TGTAAGAG"
 [3,] "A00721:81:HNLHYDSXX:1:1101:23439:1188 1:N:0:GCCGGACA+TGTAAGAG"
 [4,] "A00721:81:HNLHYDSXX:1:1101:25256:1016 1:N:0:GCCGGACA+TGTAAGAG"
 [5,] "A00721:81:HNLHYDSXX:1:1101:10619:1063 1:N:0:GCCGGACA+TGTAAGAG"
 [6,] "A00721:81:HNLHYDSXX:1:1101:27624:1078 1:N:0:GCCGGACA+TGTAAGAG"
 [7,] "A00721:81:HNLHYDSXX:1:1101:24026:1172 1:N:0:GCCGGACA+TGTAAGAG"
 [8,] "A00721:81:HNLHYDSXX:1:1101:9733:1031 1:N:0:GCCGGACA+TGTAAGAG"
 [9,] "A00721:81:HNLHYDSXX:1:1101:32859:1188 1:N:0:GCCGGACA+TGTAAGAG"
[10,] "A00721:81:HNLHYDSXX:1:1101:3188:1094 1:N:0:GCCGGACA+TGTAAGAG"
```

Note on fastq scripts:
My scripts use packages that are not compatible with the R version on the server (3.6), so I ran them on subsets of the full files on my own machine. My machine isn't powerful enough to run the scripts on the full fastq files.
However, they work normally and get the required output. If the R version on the server gets updated, I would be able to run them on the whole file – I couldn't update it myself because I don't have sudo privileges.

Subset Test Files

```
zcat 392_1_trimmed_R1_paired.fastq.gz | head -n 1000 > test1.fastq
zcat 392_2_trimmed_R2_paired.fastq.gz | head -n 1000 > test2.fastq
scp -r
pia.chouaifaty@linuxdev.accbyblos.lau.edu.lb:FunctionalFinalProject/te
st* /Users/piachouaifaty
```

Indexing Reference Genome (Chromosome 13)

```
bwa index -p chr13bwaidx -a bwtsw chr13.fa
#-p filename, by convention genome|algo|idx
#-a index algo (bwtsw for long genomes and is for short ones)
```

```
[bwt_gen] Finished constructing BWT in 72 iterations.
[bwa_index] 110.82 seconds elapse.
[bwa_index] Update BWT... 0.71 sec
[bwa_index] Pack forward-only FASTA... 0.81 sec
[bwa_index] Construct SA from BWT and Occ... 32.72 sec
[main] Version: 0.7.17-r1188
[main] CMD: bwa index -p chr13bwaidx -a bwtsw chr13.fa
[main] Real time: 146.446 sec; CPU: 146.302 sec
```

Assigning Read Group

The sample we are dealing with is from a single organism on the same flowcell lane, so all the reads belong to the same read group.

ID = Read group identifier: rg1
PU = Platform Unit:
{FLOWCELL_BARCODE}.{LANE}.{SAMPLE_BARCODE}.
HNLHYDSXX:1:GCCGGACA+TGTAAGAG

SM = Sample = 392
PL = Platform/technology used to produce the read = ILLUMINA
LB = DNA preparation library identifier = lib1

Full Read Group:
@RG\tID:rg1\tSM:392\tPL:ILLUMINA\tLB:lib1\t:PU:HNLHYDSXX:1:GCCGGACA+TGTAAGAG

Aligning to Reference Genome (BWA)

```
bwa mem -t 16 \
-R
'@RG\tID:rg1\tSM:392\tPL:ILLUMINA\tLB:lib1\t:PU:HNLHYDSXX:1:GCCGGACA+TGTAAGAG' \
/ref_chrom/chr13bwaidx \ #index file path #full path better
392_1_trimmed_R1_paired.fastq.gz  \ #file 1
392_2_trimmed_R2_paired.fastq.gz \ #file 2
> 392_aln.sam #redirect output to sam file
```

```
Processed 429766 reads in 120.953 CPU sec, 7.628 real sec
[main] Version: 0.7.17-r1188
[main] CMD: bwa mem -t 16 -R @RG\tID:rg1\tSM:392\tPL:ILLUMINA\tLB:lib1\t:PU:HNLHYDSXX:1:GCCGGACA+TGTAAGAG
ref_chrom/chr13bwaidx 392_1_trimmed_R1_paired.fastq.gz 392_2_trimmed_R2_paired.fastq.gz
[main] Real time: 1084.125 sec; CPU: 17303.121 sec
```

Cleaning up and Converting SAM to BAM

```
samtools fixmate -O bam 392_aln.sam 392_aln.bam
```

Validating SAM

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk --java-options "-Xmx16g"
ValidateSamFile INPUT=392_aln.bam MODE=SUMMARY
```

```
No errors found
[Mon Dec 14 20:32:42 EET 2020] picard.sam.ValidateSamFile done. Elapsed time: 5.01 minutes.
Runtime.totalMemory()=2985820160
Tool returned:
0
```

Sorting the SAM file

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk --java-options "-Xmx16g"
SortSam INPUT=392_aln.bam OUTPUT=392_sorted.bam SORT_ORDER=coordinate
```

```
[Mon Dec 14 21:41:00 EET 2020] picard.sam.SortSam done. Elapsed time: 8.33 minutes.
Runtime.totalMemory()=6858735616
Tool returned:
0
```

Marking Duplicates

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk --java-options "-Xmx16g"
MarkDuplicates INPUT=392_sorted.bam OUTPUT=392_dedup.bam
METRICS_FILE=392.metrics
```

```
[Mon Dec 14 21:52:24 EET 2020] picard.sam.markduplicates.MarkDuplicates done. Elapsed time: 6.75 minutes.
Runtime.totalMemory()=14134280192
Tool returned:
0
```

SAM/BAM Statistics

-f only output reads with that bit
-F only output reads WITHOUT that bit

Description	Code/Formula	Output
Count Duplicates *read is PCR or optical duplicate (0x400)*	`samtools view -c -f 0x400 392_dedup.bam`	934746
Total Reads in BAM may include unmapped and duplicated multi-aligned reads (each aligned location per mapped read)	`samtools view -c 392_dedup.bam`	61204902
Count Mapped (All) *read unmapped (0x4)* *-F to exclude*	`samtools view -c -F 0x4 392_dedup.bam`	5805578
Count Mapped (primary aligned) Reads *read unmapped (0x4)* *not primary alignment (0x100)* *Flag: 260* *-F to exclude*	`samtools view -c -F 260 392_dedup.bam`	5805578
Count Unique (without multimapping) *read unmapped (0x4)* *excludes unmapped reads, sorts and keeps only unique*	`samtools view -F 0x4 392_dedup.bam | cut -f 1 | sort | uniq | wc -l`	2619706
Percent Reads Mapped	$\frac{Count_{Mapped_{All}}}{Total\ Reads} * 100$ `(5805578/61204902)x100=9.4%`	9.4% mapped to chr13
Number of reads without a pair complement *An alignment with an unmapped mate is marked with a '*' in column 7*	`samtools view 392_dedup.bam | cut -f 7 | grep -c '*'`	55399324
Reads with Insertions/Deletions *column 6 has insertions and deletions*	`samtools view 392_dedup.bam | cut -f 6 | grep -c -E 'I|D'`	997161
Use the CIGAR string, to compute the **number of reads <u>without any Insertion or Deletion</u>**	$WITHOUT\_INDEL = TOTALMAPPED - WITH\_INDEL$ `5805578-997161`	4808417
Number of supplementary reads *supplementary alignment (0x800)*	`samtools view -c -f 0x800 392_dedup.bam`	566166

Average Mapping score/quality for the mapped reads *-F 0X4 excludes unmapped reads*	*awk gets the MapQ* `samtools view -F 0x4 392_dedup.bam | awk '{sum+=$5} END {print "Mean MAPQ =",sum/NR}'`	Mean MAPQ = 19.1783

Realignment
(need an index file and a dictionary file)

Creating Dictionary

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk CreateSequenceDictionary \
R=ref_chrom/chr13.fa \
O=ref_chrom/chr13.dict
```

```
[Mon Dec 14 22:14:21 EET 2020] picard.sam.CreateSequenceDictionary done. Elapsed time: 0.03 minutes.
Runtime.totalMemory()=2084569088
Tool returned:
0
```

Indexing Again

```
samtools faidx ref_chrom/chr13.fa
```

BaseRecalibrator

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk BaseRecalibrator \
-I 392_dedup.bam \
-R ref_chrom/chr13.fa \
--known-sites /mnt/NGSdata/snpdb151_All_20180418.vcf \
-O recal_data.table
```

```
[December 15, 2020 4:19:37 PM EET] org.broadinstitute.hellbender.tools.walkers.bqsr.BaseRecalibrator done.
Elapsed time: 6.55 minutes.
Runtime.totalMemory()=13362003968
Tool returned:
SUCCESS
```

ApplyBQSR

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk ApplyBQSR \
-R ref_chrom/chr13.fa \
-I 392_dedup.bam \
--bqsr-recal-file recal_data.table \
-O output_recal.bam
```

```
[December 15, 2020 4:52:01 PM EET] org.broadinstitute.hellbender.tools.walkers.bqsr.BaseRecalibrator done.
Elapsed time: 6.26 minutes.
Runtime.totalMemory()=14309392384
Tool returned:
SUCCESS
```

HaplotypeCaller
runs per-sample to generate an intermediate GVCF (not to be used in final analysis), which can then be used in GenotypeGVCFs for joint genotyping of multiple samples in a very efficient way.
gvcf file instead of vcf, for grouping samples according to genotype / case/controls

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk --java-options "-Xmx16g"
HaplotypeCaller \
-R ref_chrom/chr13.fa \
-I output_recal.bam \
-O output.g.vcf.gz \
-ERC GVCF
```

```
[December 15, 2020 7:48:11 PM EET]
org.broadinstitute.hellbender.tools.walkers.haplotypecaller.HaplotypeCaller done. Elapsed time: 72.30
minutes.
Runtime.totalMemory()=3538944000
```

GenotypeGVCF
joint genotyping on a single input, which may contain one or many samples

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk --java-options "-Xmx16g"
GenotypeGVCFs \
-R ref_chrom/chr13.fa \
-V output.g.vcf.gz \ #input of this command is output of Haplotype
Caller
-O output.vcf.gz
```

```
[December 15, 2020 8:06:46 PM EET] org.broadinstitute.hellbender.tools.walkers.GenotypeGVCFs done. Elapsed
time: 0.92 minutes.
Runtime.totalMemory()=5145886720
```

Variants

According to GATK Documentation:

Counting all variants

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk CountVariants -V
output.vcf.gz
```

```
[December 16, 2020 1:09:30 AM EET] org.broadinstitute.hellbender.tools.walkers.CountVariants done. Elapsed
time: 0.02 minutes.
Runtime.totalMemory()=2227699712
Tool returned:
22711
```

Counting SNPs

1. Selecting SNPs

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk SelectVariants -R
ref_chrom/chr13.fa -V output.vcf.gz  --select-type-to-include SNP -O
SNP_392.vcf
```

```
[December 16, 2020 1:12:58 AM EET] org.broadinstitute.hellbender.tools.walkers.variantutils.SelectVariants
done. Elapsed time: 0.04 minutes.
Runtime.totalMemory()=2216689664
```

2. Counting Selected SNPS

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk CountVariants -V
SNP_392.vcf
```

```
[December 16, 2020 1:16:23 AM EET] org.broadinstitute.hellbender.tools.walkers.CountVariants done. Elapsed
time: 0.01 minutes.
Runtime.totalMemory()=2205155328
Tool returned:21953
```

Counting INDELS

1. Selecting INDELs

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk SelectVariants -R
ref_chrom/chr13.fa -V output.vcf.gz  --select-type-to-include INDEL -O
INDEL_392.vcf
```

```
[December 16, 2020 1:17:10 AM EET] org.broadinstitute.hellbender.tools.walkers.variantutils.SelectVariants
done. Elapsed time: 0.02 minutes.
Runtime.totalMemory()=2123890688
```

2. Counting Selected Indels

```
/mnt/gkhazen/NGS-Fall2020/gatk-4.1.9.0/gatk CountVariants -V
INDEL_392.vcf
```

```
[December 16, 2020 1:17:51 AM EET] org.broadinstitute.hellbender.tools.walkers.CountVariants done. Elapsed
time: 0.01 minutes.
Runtime.totalMemory()=2286944256
Tool returned:
755
```

Homozygote Wild Type, Heterozygote, Homozygote Mutant

I write a script to get these stats
First, I copy the vcf file to my machine.

```
scp -r
pia.chouaifaty@linuxdev.accbyblos.lau.edu.lb:FunctionalFinalProject/ou
tput.vcf.gz /Users/piachouaifaty
```

Script: Count_Hom_Hetero.R
Takes vcf file path as input
Call: Rscript Count_Hom_Hetero.R output.vcf

```
#!/usr/bin/env Rscript

#Rscript Count_Hom_Hetero.R path

if (!requireNamespace("vcfR", quietly = TRUE))
  install.packages("vcfR")
library(vcfR)

main=function(vcfpath)
{
  vcf = read.vcfR(vcfpath, verbose = FALSE )
  opt=vcf@gt
  #hom_wild = c("0/0", "0|0") #homozygous wild type
  #hetero = c("0/1", "0|1") #heterozygous
  #hom_mut= c("1/1", "1|1") #homozygous mutant

  count_hom_wild=0
  count_hetero=0
  count_hom_mut=0

  for (i in 1:nrow(opt))
  {
    ind=strsplit(opt[i,"392"],":")[[1]][1] #split by first :

    if (ind=="0/0"|ind=="0|0")
    {count_hom_wild=count_hom_wild+1}
    else if (ind=="0/1"|ind=="0|1")
    {count_hetero=count_hetero+1}
    else if (ind=="1/1"|ind=="1|1")
    {count_hom_mut=count_hom_mut+1}
  }

  print(paste0("Homozygous Wild Type: ", count_hom_wild))
  print(paste0("Heterozygous: ", count_hetero))
  print(paste0("Homozygous Mutant: ", count_hom_mut))
```

```
}

args = commandArgs(trailingOnly = TRUE)
vcfpath = args[1]

main(vcfpath)
```

```
(base) Pias-MacBook-Air:~ piachouaifaty$ Rscript Count_Hom_Hetero.R output.vcf

   *****       ***   vcfR   ***       *****
   This is vcfR 1.12.0
     browseVignettes('vcfR') # Documentation
     citation('vcfR') # Citation
   *****       *****      *****       *****

[1] "Homozygous Wild Type: 0"
[1] "Heterozygous: 4253"
[1] "Homozygous Mutant: 18408"
(base) Pias-MacBook-Air:~ piachouaifaty$ time Rscript Count_Hom_Hetero.R output.vc

   *****       ***   vcfR   ***       *****
   This is vcfR 1.12.0
     browseVignettes('vcfR') # Documentation
     citation('vcfR') # Citation
   *****       *****      *****       *****

[1] "Homozygous Wild Type: 0"
[1] "Heterozygous: 4253"
[1] "Homozygous Mutant: 18408"

real    0m2.440s
user    0m2.186s
sys     0m0.208s
```

Homozygous Wild Type: 0
Heterozygous: 4253
Homozygous Mutant: 18408