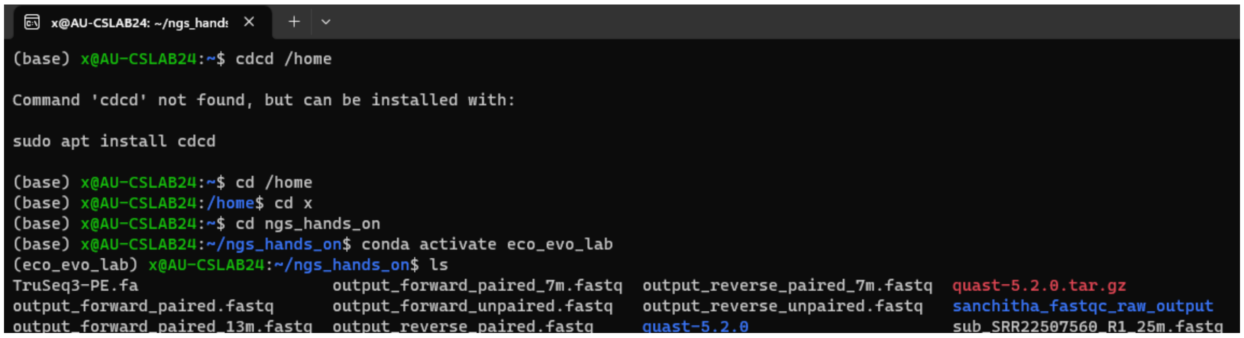
Genome Assembly

STEPS:

1. Navigate to the directory containing our data, using the following commands:
   cd /home
   cd ngs_hands_on

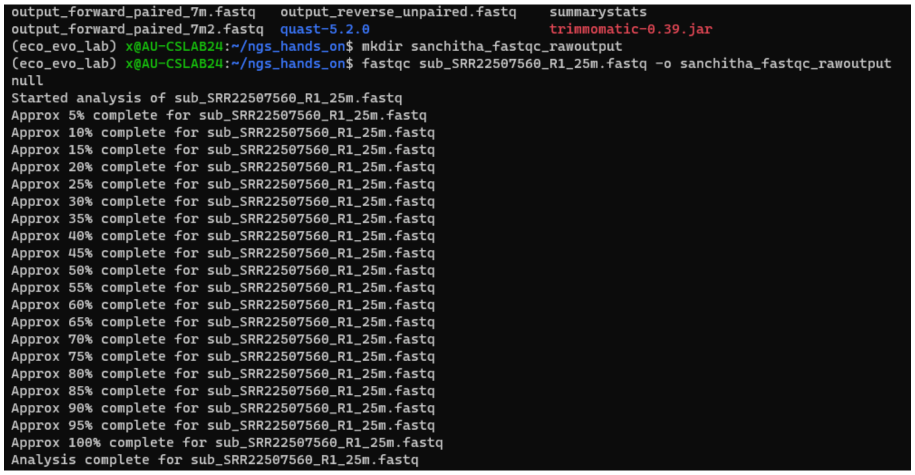
2. The conda environment containing the software needed for analysis is activated:
   conda activate eco_evo_lab

3. A directory is made to store the outputs we obtain from FastQC quality analysis:
   mkdir sanchitha_fastqc_raw_output

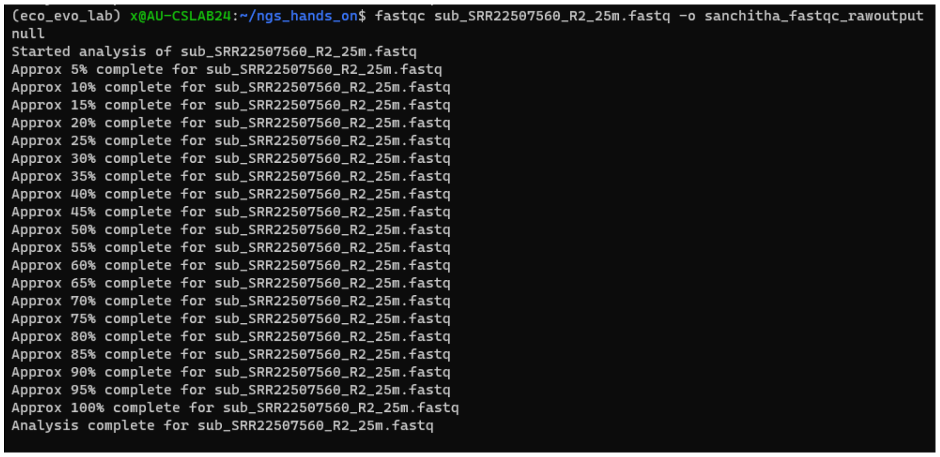


```
(base) x@AU-CSLAB24:~$ cdcd /home

Command 'cdcd' not found, but can be installed with:

sudo apt install cdcd

(base) x@AU-CSLAB24:~$ cd /home
(base) x@AU-CSLAB24:/home$ cd x
(base) x@AU-CSLAB24:~$ cd ngs_hands_on
(base) x@AU-CSLAB24:~/ngs_hands_on$ conda activate eco_evo_lab
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ ls
TruSeq3-PE.fa                   output_forward_paired_7m.fastq  output_reverse_paired_7m.fastq  quast-5.2.0.tar.gz
output_forward_paired.fastq     output_forward_unpaired.fastq   output_reverse_unpaired.fastq   sanchitha_fastqc_raw_output
output_forward_paired_13m.fastq output_reverse_paired.fastq     quast-5.2.0                     sub_SRR22507560_R1_25m.fastq
```

4. Then, we analyze the quality of the given 25 million reads of DNA using FastQC. Two sets of 25 million reads each - the forward and the reverse reads - are analyzed:
   fastq sub_SRR22507560_R1_25m.fastq -o sanchitha_fastqc_raw_output
   fastq sub_SRR22507560_R2_25m.fastq -o sanchitha_fastqc_raw_output



```
output_forward_paired_7m.fastq   output_reverse_unpaired.fastq    summarystats
output_forward_paired_7m2.fastq  quast-5.2.0                      trimmomatic-0.39.jar
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ mkdir sanchitha_fastqc_rawoutput
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ fastqc sub_SRR22507560_R1_25m.fastq -o sanchitha_fastqc_rawoutput
null
Started analysis of sub_SRR22507560_R1_25m.fastq
Approx 5% complete for sub_SRR22507560_R1_25m.fastq
Approx 10% complete for sub_SRR22507560_R1_25m.fastq
Approx 15% complete for sub_SRR22507560_R1_25m.fastq
Approx 20% complete for sub_SRR22507560_R1_25m.fastq
Approx 25% complete for sub_SRR22507560_R1_25m.fastq
Approx 30% complete for sub_SRR22507560_R1_25m.fastq
Approx 35% complete for sub_SRR22507560_R1_25m.fastq
Approx 40% complete for sub_SRR22507560_R1_25m.fastq
Approx 45% complete for sub_SRR22507560_R1_25m.fastq
Approx 50% complete for sub_SRR22507560_R1_25m.fastq
Approx 55% complete for sub_SRR22507560_R1_25m.fastq
Approx 60% complete for sub_SRR22507560_R1_25m.fastq
Approx 65% complete for sub_SRR22507560_R1_25m.fastq
Approx 70% complete for sub_SRR22507560_R1_25m.fastq
Approx 75% complete for sub_SRR22507560_R1_25m.fastq
Approx 80% complete for sub_SRR22507560_R1_25m.fastq
Approx 85% complete for sub_SRR22507560_R1_25m.fastq
Approx 90% complete for sub_SRR22507560_R1_25m.fastq
Approx 95% complete for sub_SRR22507560_R1_25m.fastq
Approx 100% complete for sub_SRR22507560_R1_25m.fastq
Analysis complete for sub_SRR22507560_R1_25m.fastq
```

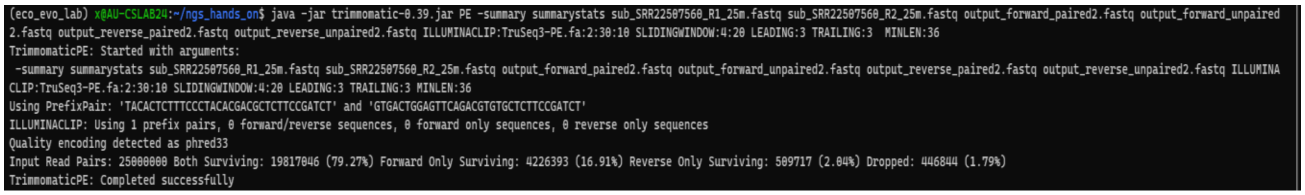


```
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ fastqc sub_SRR22507560_R2_25m.fastq -o sanchitha_fastqc_rawoutput
null
Started analysis of sub_SRR22507560_R2_25m.fastq
Approx 5% complete for sub_SRR22507560_R2_25m.fastq
Approx 10% complete for sub_SRR22507560_R2_25m.fastq
Approx 15% complete for sub_SRR22507560_R2_25m.fastq
Approx 20% complete for sub_SRR22507560_R2_25m.fastq
Approx 25% complete for sub_SRR22507560_R2_25m.fastq
Approx 30% complete for sub_SRR22507560_R2_25m.fastq
Approx 35% complete for sub_SRR22507560_R2_25m.fastq
Approx 40% complete for sub_SRR22507560_R2_25m.fastq
Approx 45% complete for sub_SRR22507560_R2_25m.fastq
Approx 50% complete for sub_SRR22507560_R2_25m.fastq
Approx 55% complete for sub_SRR22507560_R2_25m.fastq
Approx 60% complete for sub_SRR22507560_R2_25m.fastq
Approx 65% complete for sub_SRR22507560_R2_25m.fastq
Approx 70% complete for sub_SRR22507560_R2_25m.fastq
Approx 75% complete for sub_SRR22507560_R2_25m.fastq
Approx 80% complete for sub_SRR22507560_R2_25m.fastq
Approx 85% complete for sub_SRR22507560_R2_25m.fastq
Approx 90% complete for sub_SRR22507560_R2_25m.fastq
Approx 95% complete for sub_SRR22507560_R2_25m.fastq
Approx 100% complete for sub_SRR22507560_R2_25m.fastq
Analysis complete for sub_SRR22507560_R2_25m.fastq
```

5. The output contains, among other files, a web page containing a thorough analysis of the basis statistics, per base sequence quality, per sequence, per sequence GC content and adaptor quantity.

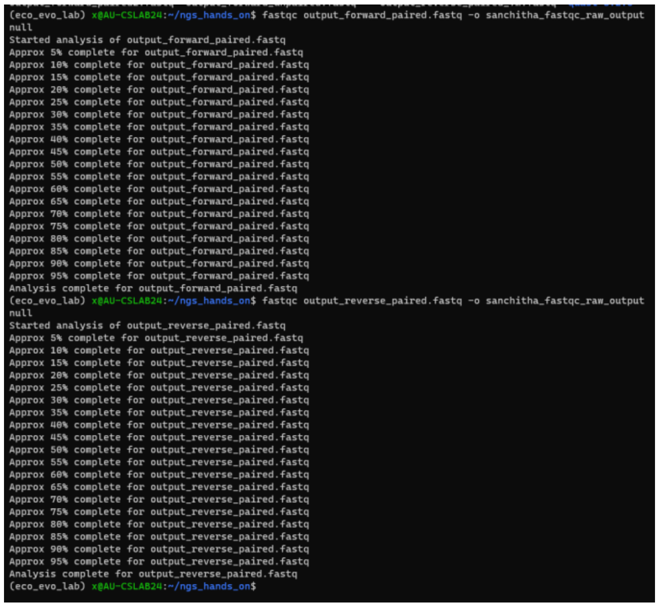
6. Since it was observed that the adaptor sequences were quite high in the raw data, the reads are trimmed to remove the adapters so that high quality reads could be obtained and further used for assembly. Very low quality reads are also removed using Trimmomatic software, but most of them are conserved.

   java -jar trimmomatic-0.39.jar PE -summary summarystats input_forward.fastq input_reverse.fastq output_forward_paired.fastq output_forward_unpaired.fastq output_reverse_paired.fastq output_reverse_unpaired.fastq ILLUMINACLIP:TruSeq3-PE.fa:2:30:10 SLIDINGWINDOW:4:20 LEADING:3 TRAILING:3 MINLEN:36



```
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ java -jar trimmomatic-0.39.jar PE -summary summarystats sub_SRR22507560_R1_25m.fastq sub_SRR22507560_R2_25m.fastq output_forward_paired2.fastq output_forward_unpaired
2.fastq output_reverse_paired2.fastq output_reverse_unpaired2.fastq ILLUMINACLIP:TruSeq3-PE.fa:2:30:10 SLIDINGWINDOW:4:20 LEADING:3 TRAILING:3  MINLEN:36
TrimmomaticPE: Started with arguments:
 -summary summarystats sub_SRR22507560_R1_25m.fastq sub_SRR22507560_R2_25m.fastq output_forward_paired2.fastq output_forward_unpaired2.fastq output_reverse_paired2.fastq output_reverse_unpaired2.fastq ILLUMINA
CLIP:TruSeq3-PE.fa:2:30:10 SLIDINGWINDOW:4:20 LEADING:3 TRAILING:3 MINLEN:36
Using PrefixPair: 'TACACTCTTTCCCTACACGACGCTCTTCCGATCT' and 'GTGACTGGAGTTCAGACGTGTGCTCTTCCGATCT'
ILLUMINACLIP: Using 1 prefix pairs, 0 forward/reverse sequences, 0 forward only sequences, 0 reverse only sequences
Quality encoding detected as phred33
Input Read Pairs: 25000000 Both Surviving: 19817046 (79.27%) Forward Only Surviving: 4226393 (16.91%) Reverse Only Surviving: 509717 (2.04%) Dropped: 446844 (1.79%)
TrimmomaticPE: Completed successfully
```

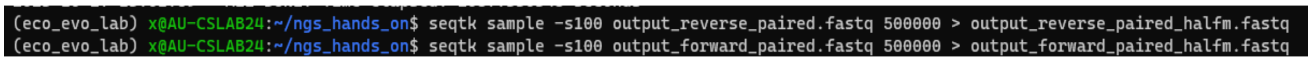
7. Then, we use FastQC to check if the quality of the trimmed reads is higher than that of the previous reads:



```
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ fastqc output_forward_paired.fastq -o sanchitha_fastqc_raw_output
null
Started analysis of output_forward_paired.fastq
Approx 5% complete for output_forward_paired.fastq
Approx 10% complete for output_forward_paired.fastq
Approx 15% complete for output_forward_paired.fastq
Approx 20% complete for output_forward_paired.fastq
Approx 25% complete for output_forward_paired.fastq
Approx 30% complete for output_forward_paired.fastq
Approx 35% complete for output_forward_paired.fastq
Approx 40% complete for output_forward_paired.fastq
Approx 45% complete for output_forward_paired.fastq
Approx 50% complete for output_forward_paired.fastq
Approx 55% complete for output_forward_paired.fastq
Approx 60% complete for output_forward_paired.fastq
Approx 65% complete for output_forward_paired.fastq
Approx 70% complete for output_forward_paired.fastq
Approx 75% complete for output_forward_paired.fastq
Approx 80% complete for output_forward_paired.fastq
Approx 85% complete for output_forward_paired.fastq
Approx 90% complete for output_forward_paired.fastq
Approx 95% complete for output_forward_paired.fastq
Analysis complete for output_forward_paired.fastq
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ fastqc output_reverse_paired.fastq -o sanchitha_fastqc_raw_output
null
Started analysis of output_reverse_paired.fastq
Approx 5% complete for output_reverse_paired.fastq
Approx 10% complete for output_reverse_paired.fastq
Approx 15% complete for output_reverse_paired.fastq
Approx 20% complete for output_reverse_paired.fastq
Approx 25% complete for output_reverse_paired.fastq
Approx 30% complete for output_reverse_paired.fastq
Approx 35% complete for output_reverse_paired.fastq
Approx 40% complete for output_reverse_paired.fastq
Approx 45% complete for output_reverse_paired.fastq
Approx 50% complete for output_reverse_paired.fastq
Approx 55% complete for output_reverse_paired.fastq
Approx 60% complete for output_reverse_paired.fastq
Approx 65% complete for output_reverse_paired.fastq
Approx 70% complete for output_reverse_paired.fastq
Approx 75% complete for output_reverse_paired.fastq
Approx 80% complete for output_reverse_paired.fastq
Approx 85% complete for output_reverse_paired.fastq
Approx 90% complete for output_reverse_paired.fastq
Approx 95% complete for output_reverse_paired.fastq
Analysis complete for output_reverse_paired.fastq
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$
```

8. Next genome assembly using the trimmed reads is done of the forward and reverse sequences. Genome assembly using 25 million reads would take a long time and might be inefficient. We subsampled 500,000 reads out of the 25 million to perform the assembly.

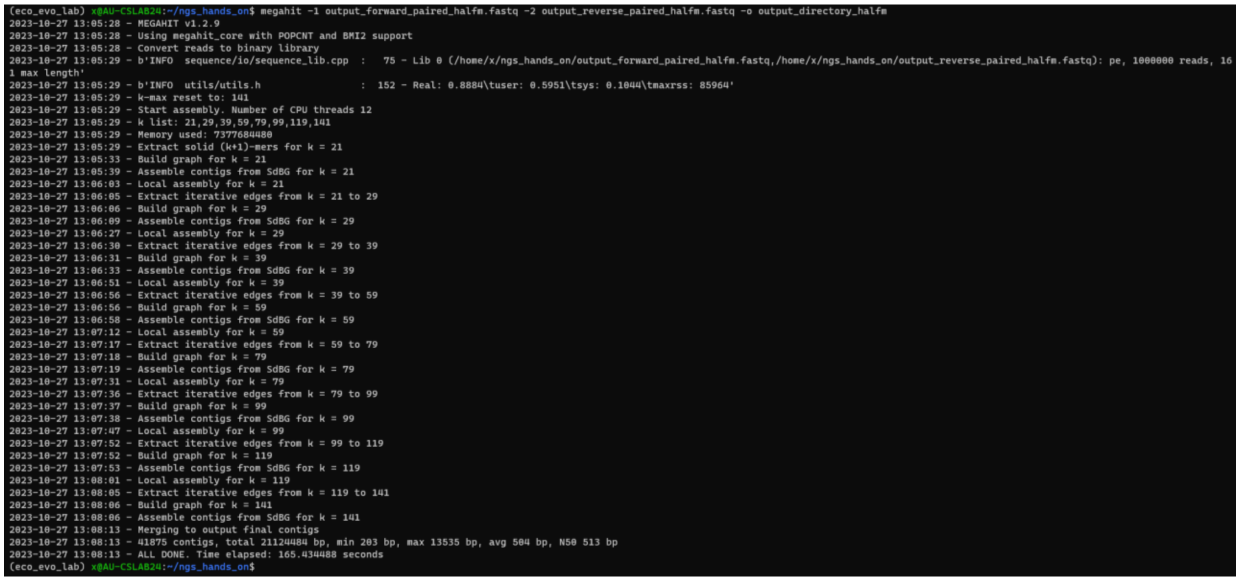
   Subsampling:
   seqtk sample -s100 output_forward_paired.fastq 500000 > output_forward_paired_halfm.fastq
   seqtk sample -s100 output_reverse_paired.fastq 500000 > output_reverse_paired_halfm.fastq



```
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ seqtk sample -s100 output_reverse_paired.fastq 500000 > output_reverse_paired_halfm.fastq
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ seqtk sample -s100 output_forward_paired.fastq 500000 > output_forward_paired_halfm.fastq
```

9. Megahit is used to perform the genome assembly using the 2 sets of 500,000 reads:

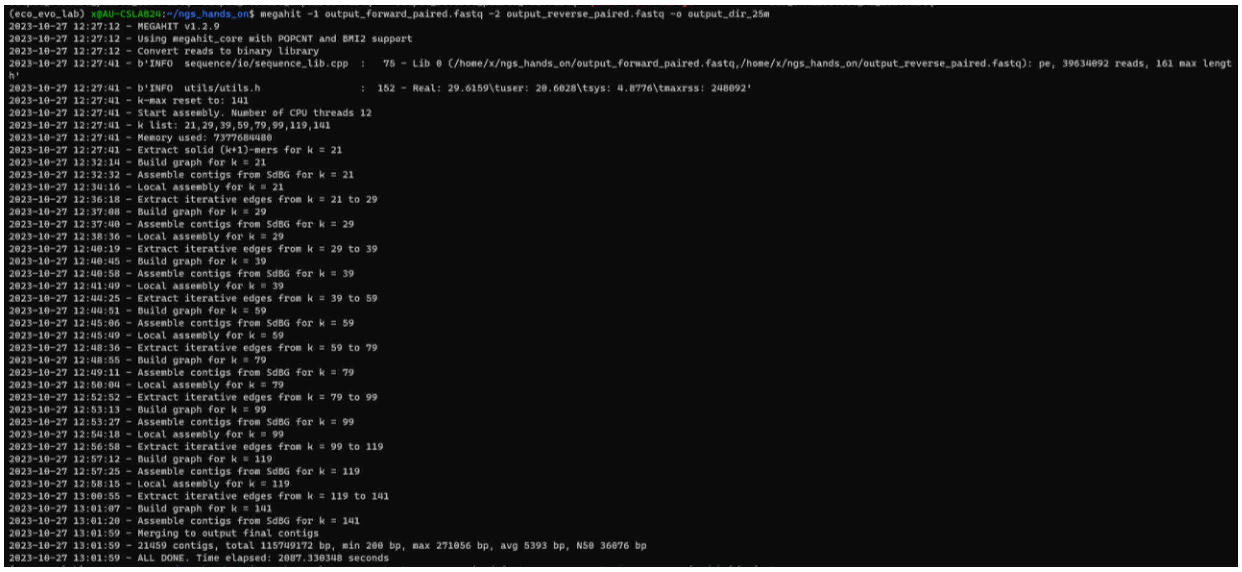
   megahit -1 output_forward_paired_halfm.fastq -2 output_reverse_paired_halfm.fastq -o output_directory_halfm



```
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ megahit -1 output_forward_paired_halfm.fastq -2 output_reverse_paired_halfm.fastq -o output_directory_halfm
2023-10-27 13:05:28 - MEGAHIT v1.2.9
2023-10-27 13:05:28 - Using megahit_core with POPCNT and BMI2 support
2023-10-27 13:05:28 - Convert reads to binary library
2023-10-27 13:05:29 - b'INFO  sequence/io/sequence_lib.cpp  :   75 - Lib 0 (/home/x/ngs_hands_on/output_forward_paired_halfm.fastq,/home/x/ngs_hands_on/output_reverse_paired_halfm.fastq): pe, 1000000 reads, 16
1 max length'
2023-10-27 13:05:29 - b'INFO  utils/utils.h                :  152 - Real: 0.8884\tuser: 0.5951\tsys: 0.1044\tmaxrss: 85964'
2023-10-27 13:05:29 - k-max reset to: 141
2023-10-27 13:05:29 - Start assembly. Number of CPU threads 12
2023-10-27 13:05:29 - k list: 21,29,39,59,79,99,119,141
2023-10-27 13:05:29 - Memory used: 7377684480
2023-10-27 13:05:29 - Extract solid (k+1)-mers for k = 21
2023-10-27 13:05:33 - Build graph for k = 21
2023-10-27 13:05:39 - Assemble contigs from SdBG for k = 21
2023-10-27 13:06:03 - Local assembly for k = 21
2023-10-27 13:06:05 - Extract iterative edges from k = 21 to 29
2023-10-27 13:06:06 - Build graph for k = 29
2023-10-27 13:06:09 - Assemble contigs from SdBG for k = 29
2023-10-27 13:06:27 - Local assembly for k = 29
2023-10-27 13:06:30 - Extract iterative edges from k = 29 to 39
2023-10-27 13:06:31 - Build graph for k = 39
2023-10-27 13:06:33 - Assemble contigs from SdBG for k = 39
2023-10-27 13:06:51 - Local assembly for k = 39
2023-10-27 13:06:56 - Extract iterative edges from k = 39 to 59
2023-10-27 13:06:56 - Build graph for k = 59
2023-10-27 13:06:58 - Assemble contigs from SdBG for k = 59
2023-10-27 13:07:12 - Local assembly for k = 59
2023-10-27 13:07:17 - Extract iterative edges from k = 59 to 79
2023-10-27 13:07:18 - Build graph for k = 79
2023-10-27 13:07:19 - Assemble contigs from SdBG for k = 79
2023-10-27 13:07:31 - Local assembly for k = 79
2023-10-27 13:07:36 - Extract iterative edges from k = 79 to 99
2023-10-27 13:07:37 - Build graph for k = 99
2023-10-27 13:07:38 - Assemble contigs from SdBG for k = 99
2023-10-27 13:07:47 - Local assembly for k = 99
2023-10-27 13:07:52 - Extract iterative edges from k = 99 to 119
2023-10-27 13:07:52 - Build graph for k = 119
2023-10-27 13:07:53 - Assemble contigs from SdBG for k = 119
2023-10-27 13:08:01 - Local assembly for k = 119
2023-10-27 13:08:05 - Extract iterative edges from k = 119 to 141
2023-10-27 13:08:06 - Build graph for k = 141
2023-10-27 13:08:06 - Assemble contigs from SdBG for k = 141
2023-10-27 13:08:13 - Merging to output final contigs
2023-10-27 13:08:13 - 41875 contigs, total 21124484 bp, min 203 bp, max 13535 bp, avg 504 bp, N50 513 bp
2023-10-27 13:08:13 - ALL DONE. Time elapsed: 165.434488 seconds
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$
```

10. We then do the genome assembly using 25 million reads. This took quite a bit of time.

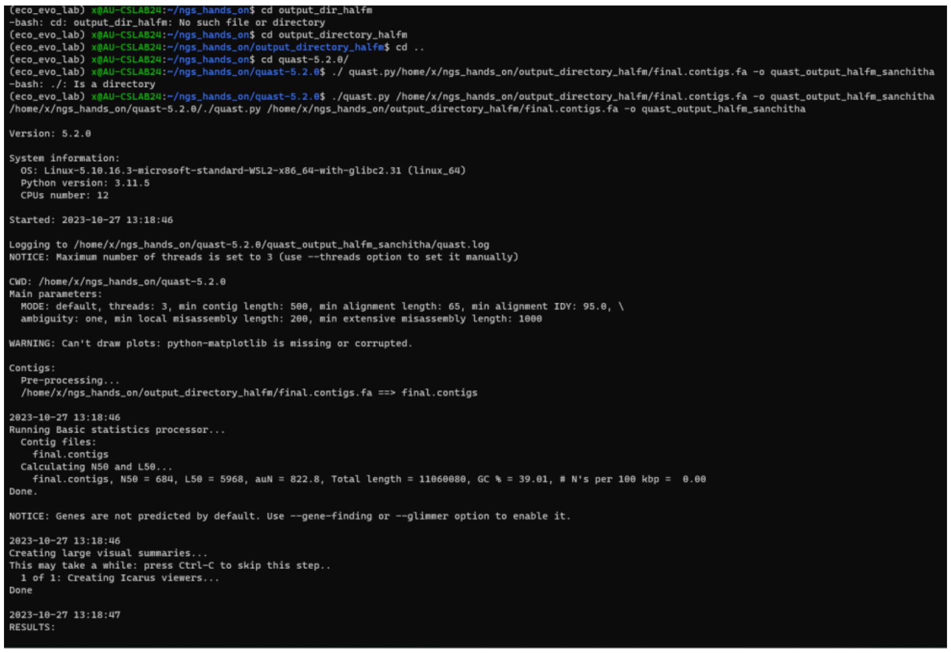
    megahit -1 output_forward_paired.fastq -2 output_reverse_paired.fastq -o output_dir_25m



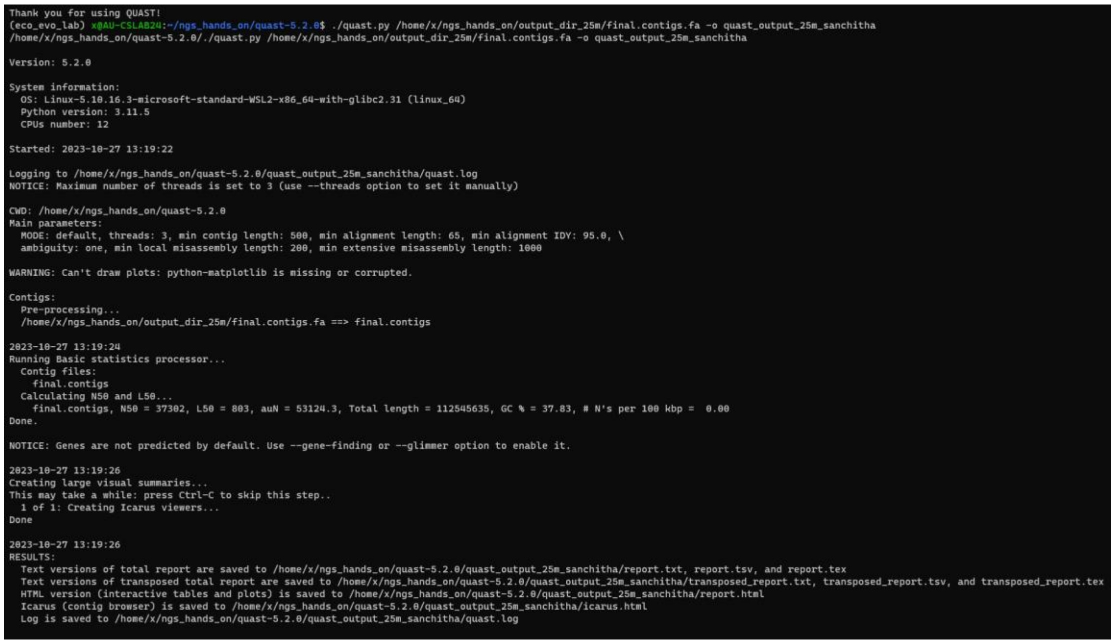
```
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ megahit -1 output_forward_paired.fastq -2 output_reverse_paired.fastq -o output_dir_25m
2023-10-27 12:27:12 - MEGAHIT v1.2.9
2023-10-27 12:27:12 - Using megahit_core with POPCNT and BMI2 support
2023-10-27 12:27:12 - Convert reads to binary library
2023-10-27 12:27:41 - b'INFO  sequence/io/sequence_lib.cpp  :   75 - Lib 0 (/home/x/ngs_hands_on/output_forward_paired.fastq,/home/x/ngs_hands_on/output_reverse_paired.fastq): pe, 39634092 reads, 161 max lengt
h'
2023-10-27 12:27:41 - b'INFO  utils/utils.h                :  152 - Real: 29.6159\tuser: 20.6028\tsys: 4.8776\tmaxrss: 248092'
2023-10-27 12:27:41 - k-max reset to: 141
2023-10-27 12:27:41 - Start assembly. Number of CPU threads 12
2023-10-27 12:27:41 - k list: 21,29,39,59,79,99,119,141
2023-10-27 12:27:41 - Memory used: 7377684480
2023-10-27 12:27:41 - Extract solid (k+1)-mers for k = 21
2023-10-27 12:32:14 - Build graph for k = 21
2023-10-27 12:32:32 - Assemble contigs from SdBG for k = 21
2023-10-27 12:34:16 - Local assembly for k = 21
2023-10-27 12:36:18 - Extract iterative edges from k = 21 to 29
2023-10-27 12:37:08 - Build graph for k = 29
2023-10-27 12:37:40 - Assemble contigs from SdBG for k = 29
2023-10-27 12:38:36 - Local assembly for k = 29
2023-10-27 12:40:19 - Extract iterative edges from k = 29 to 39
2023-10-27 12:40:45 - Build graph for k = 39
2023-10-27 12:40:58 - Assemble contigs from SdBG for k = 39
2023-10-27 12:41:49 - Local assembly for k = 39
2023-10-27 12:44:25 - Extract iterative edges from k = 39 to 59
2023-10-27 12:44:51 - Build graph for k = 59
2023-10-27 12:45:06 - Assemble contigs from SdBG for k = 59
2023-10-27 12:45:49 - Local assembly for k = 59
2023-10-27 12:48:36 - Extract iterative edges from k = 59 to 79
2023-10-27 12:48:55 - Build graph for k = 79
2023-10-27 12:49:11 - Assemble contigs from SdBG for k = 79
2023-10-27 12:50:04 - Local assembly for k = 79
2023-10-27 12:52:52 - Extract iterative edges from k = 79 to 99
2023-10-27 12:53:13 - Build graph for k = 99
2023-10-27 12:53:27 - Assemble contigs from SdBG for k = 99
2023-10-27 12:54:18 - Local assembly for k = 99
2023-10-27 12:56:58 - Extract iterative edges from k = 99 to 119
2023-10-27 12:57:12 - Build graph for k = 119
2023-10-27 12:57:25 - Assemble contigs from SdBG for k = 119
2023-10-27 12:58:15 - Local assembly for k = 119
2023-10-27 13:00:55 - Extract iterative edges from k = 119 to 141
2023-10-27 13:01:07 - Build graph for k = 141
2023-10-27 13:01:20 - Assemble contigs from SdBG for k = 141
2023-10-27 13:01:59 - Merging to output final contigs
2023-10-27 13:01:59 - 21459 contigs, total 115749172 bp, min 200 bp, max 271056 bp, avg 5393 bp, N50 36076 bp
2023-10-27 13:01:59 - ALL DONE. Time elapsed: 2087.330348 seconds
```

11. The two genome assemblies are analyzed using QUAST software. We go to the directory where the QUAST software is installed and use the following commands to do quality analysis of the half million reads' assembly and the 25 million reads' assembly.
    ./quast.py /home/x/ngs_hands_on/output_directory_halfm/final.contigs.fa -o quast_output_halfm_sanchitha

./quast.py /home/x/ngs_hands_on/output_dir_25m/final.contigs.fa -o quast_output_25m_sanchitha



```
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ cd output_dir_halfm
-bash: cd: output_dir_halfm: No such file or directory
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ cd output_directory_halfm
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on/output_directory_halfm$ cd ..
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on$ cd quast-5.2.0/
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on/quast-5.2.0$ ./ quast.py/home/x/ngs_hands_on/output_directory_halfm/final.contigs.fa -o quast_output_halfm_sanchitha
-bash: ./: Is a directory
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on/quast-5.2.0$ ./quast.py /home/x/ngs_hands_on/output_directory_halfm/final.contigs.fa -o quast_output_halfm_sanchitha
/home/x/ngs_hands_on/quast-5.2.0/./quast.py /home/x/ngs_hands_on/output_directory_halfm/final.contigs.fa -o quast_output_halfm_sanchitha

Version: 5.2.0

System information:
  OS: Linux-5.10.16.3-microsoft-standard-WSL2-x86_64-with-glibc2.31 (linux_64)
  Python version: 3.11.5
  CPUs number: 12

Started: 2023-10-27 13:18:46

Logging to /home/x/ngs_hands_on/quast-5.2.0/quast_output_halfm_sanchitha/quast.log
NOTICE: Maximum number of threads is set to 3 (use --threads option to set it manually)

CWD: /home/x/ngs_hands_on/quast-5.2.0
Main parameters:
  MODE: default, threads: 3, min contig length: 500, min alignment length: 65, min alignment IDY: 95.0, \
  ambiguity: one, min local misassembly length: 200, min extensive misassembly length: 1000

WARNING: Can't draw plots: python-matplotlib is missing or corrupted.

Contigs:
  Pre-processing...
  /home/x/ngs_hands_on/output_directory_halfm/final.contigs.fa ==> final.contigs

2023-10-27 13:18:46
Running Basic statistics processor...
  Contig files:
    final.contigs
  Calculating N50 and L50...
    final.contigs, N50 = 684, L50 = 5968, auN = 822.8, Total length = 11060080, GC % = 39.01, # N's per 100 kbp =  0.00
Done.

NOTICE: Genes are not predicted by default. Use --gene-finding or --glimmer option to enable it.

2023-10-27 13:18:46
Creating large visual summaries...
This may take a while: press Ctrl-C to skip this step..
  1 of 1: Creating Icarus viewers...
Done

2023-10-27 13:18:47
RESULTS:
```



```
Thank you for using QUAST!
(eco_evo_lab) x@AU-CSLAB24:~/ngs_hands_on/quast-5.2.0$ ./quast.py /home/x/ngs_hands_on/output_dir_25m/final.contigs.fa -o quast_output_25m_sanchitha
/home/x/ngs_hands_on/quast-5.2.0/./quast.py /home/x/ngs_hands_on/output_dir_25m/final.contigs.fa -o quast_output_25m_sanchitha

Version: 5.2.0

System information:
  OS: Linux-5.10.16.3-microsoft-standard-WSL2-x86_64-with-glibc2.31 (linux_64)
  Python version: 3.11.5
  CPUs number: 12

Started: 2023-10-27 13:19:22

Logging to /home/x/ngs_hands_on/quast-5.2.0/quast_output_25m_sanchitha/quast.log
NOTICE: Maximum number of threads is set to 3 (use --threads option to set it manually)

CWD: /home/x/ngs_hands_on/quast-5.2.0
Main parameters:
  MODE: default, threads: 3, min contig length: 500, min alignment length: 65, min alignment IDY: 95.0, \
  ambiguity: one, min local misassembly length: 200, min extensive misassembly length: 1000

WARNING: Can't draw plots: python-matplotlib is missing or corrupted.

Contigs:
  Pre-processing...
  /home/x/ngs_hands_on/output_dir_25m/final.contigs.fa ==> final.contigs

2023-10-27 13:19:24
Running Basic statistics processor...
  Contig files:
    final.contigs
  Calculating N50 and L50...
    final.contigs, N50 = 37302, L50 = 803, auN = 53124.3, Total length = 112545635, GC % = 37.83, # N's per 100 kbp =  0.00
Done.

NOTICE: Genes are not predicted by default. Use --gene-finding or --glimmer option to enable it.

2023-10-27 13:19:26
Creating large visual summaries...
This may take a while: press Ctrl-C to skip this step..
  1 of 1: Creating Icarus viewers...
Done

2023-10-27 13:19:26
RESULTS:
  Text versions of total report are saved to /home/x/ngs_hands_on/quast-5.2.0/quast_output_25m_sanchitha/report.txt, report.tsv, and report.tex
  Text versions of transposed total report are saved to /home/x/ngs_hands_on/quast-5.2.0/quast_output_25m_sanchitha/transposed_report.txt, transposed_report.tsv, and transposed_report.tex
  HTML version (interactive tables and plots) is saved to /home/x/ngs_hands_on/quast-5.2.0/quast_output_25m_sanchitha/report.html
  Icarus (contig browser) is saved to /home/x/ngs_hands_on/quast-5.2.0/quast_output_25m_sanchitha/icarus.html
  Log is saved to /home/x/ngs_hands_on/quast-5.2.0/quast_output_25m_sanchitha/quast.log
```

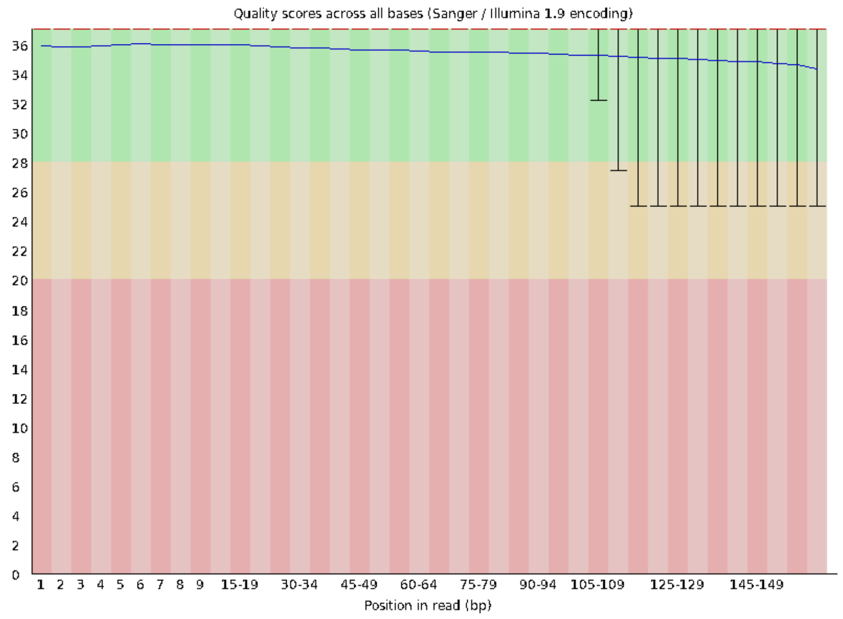
RESULTS:

1. FastQC Results of Raw Reads:
   a. Forward Raw Reads:

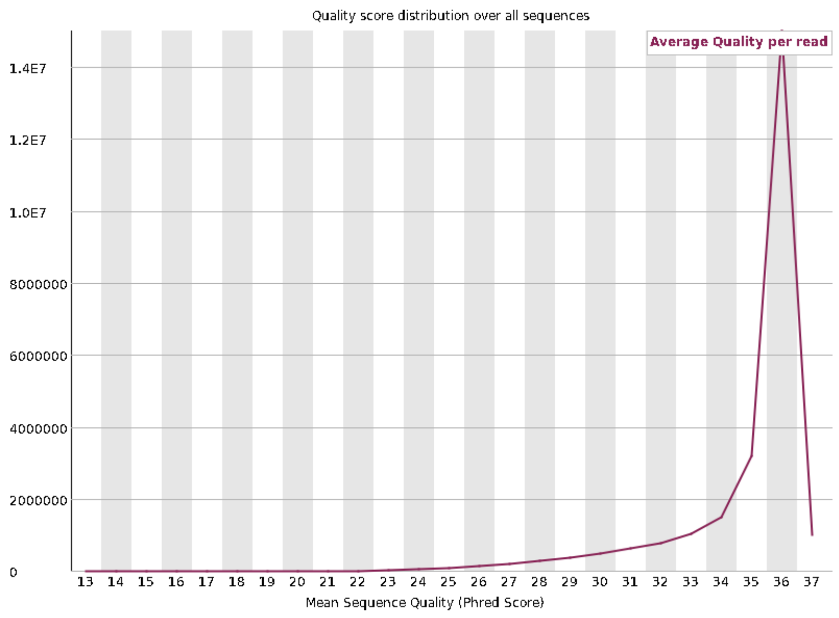
✓ Basic Statistics

Measure	Value
Filename	sub_SRR22507560_R1_25m.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	25000000
Total Bases	4 Gbp
Sequences flagged as poor quality	0
Sequence length	161
%GC	38

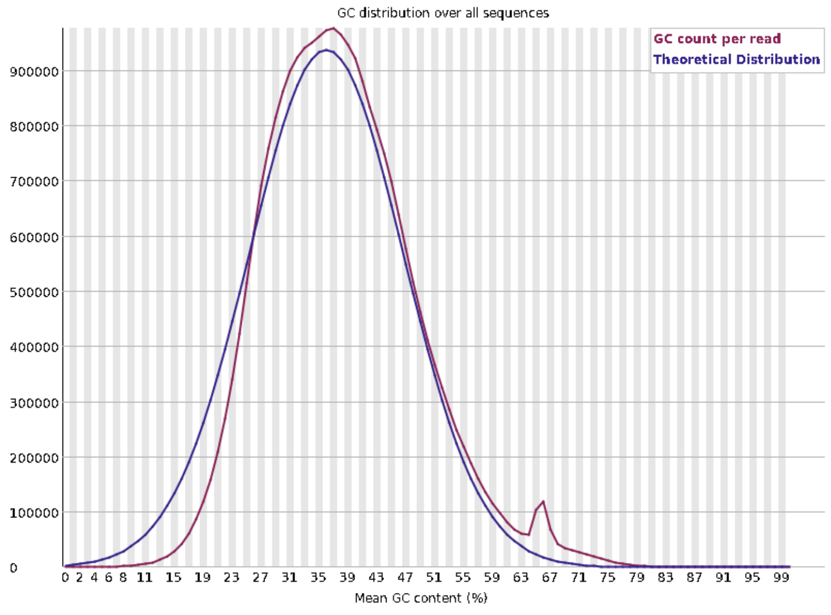
✓ Per base sequence quality



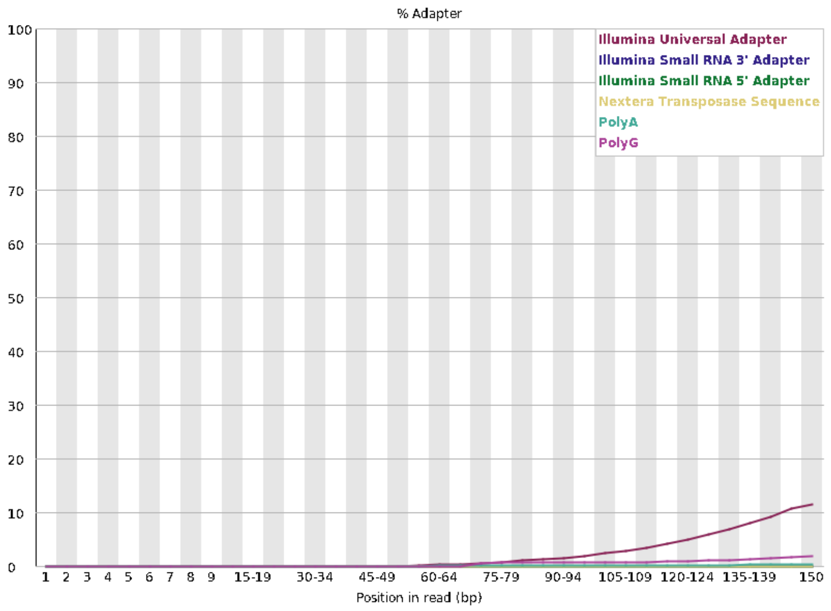
✓ Per sequence quality scores



✓ Per sequence GC content

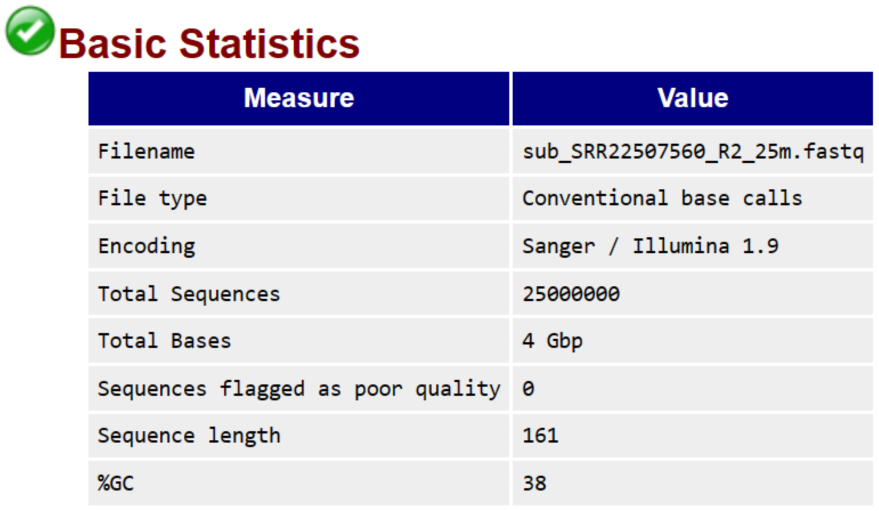


Adapter Content



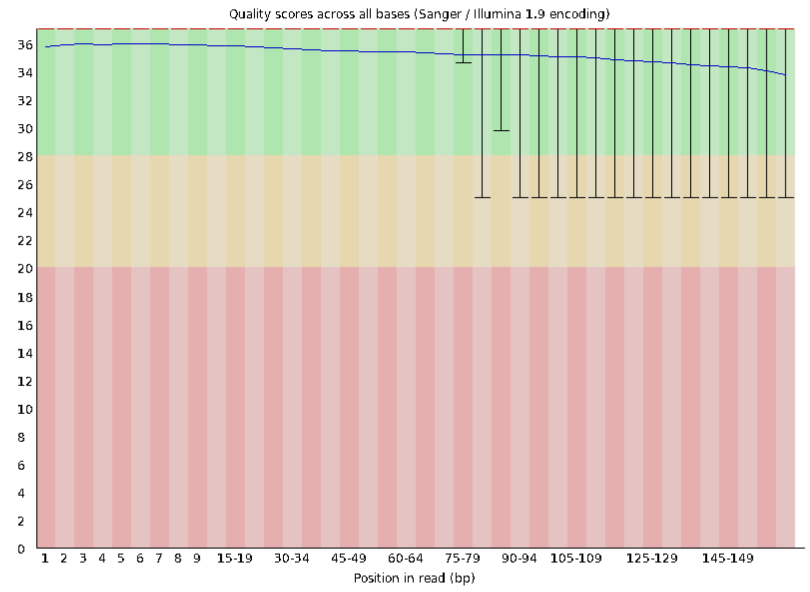
b. Reverse Raw Reads:

Basic Statistics

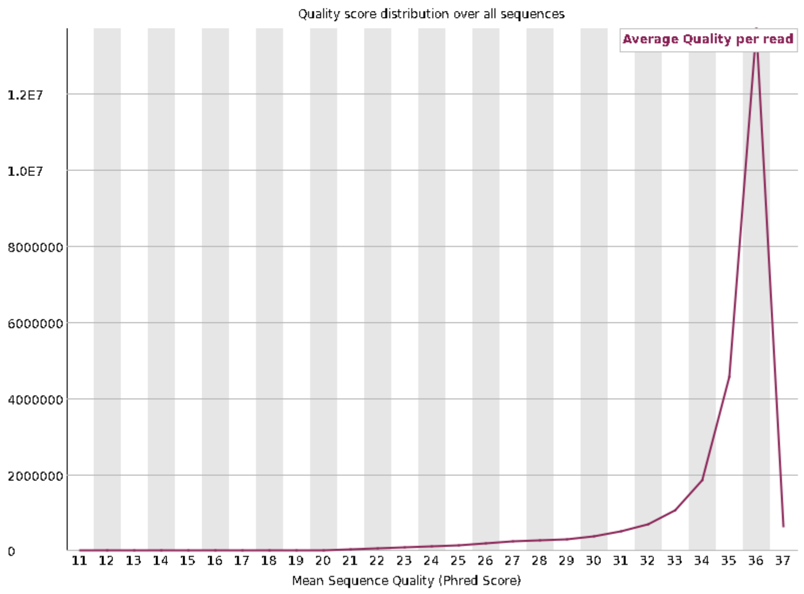


Measure	Value
Filename	sub_SRR22507560_R2_25m.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	25000000
Total Bases	4 Gbp
Sequences flagged as poor quality	0
Sequence length	161
%GC	38

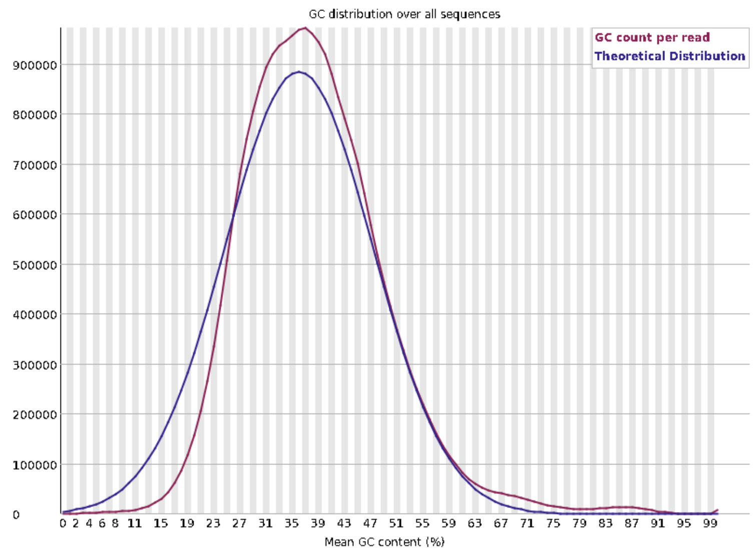
✓ Per base sequence quality



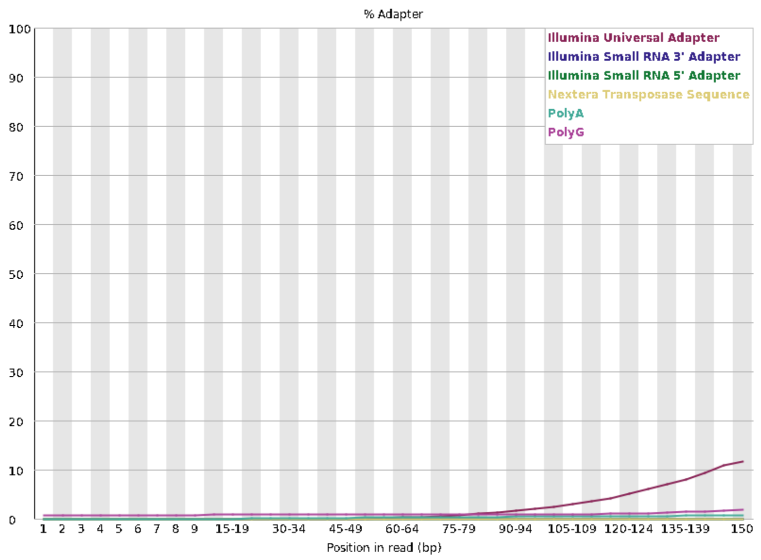
✓ Per sequence quality scores



Per sequence GC content



Adapter Content

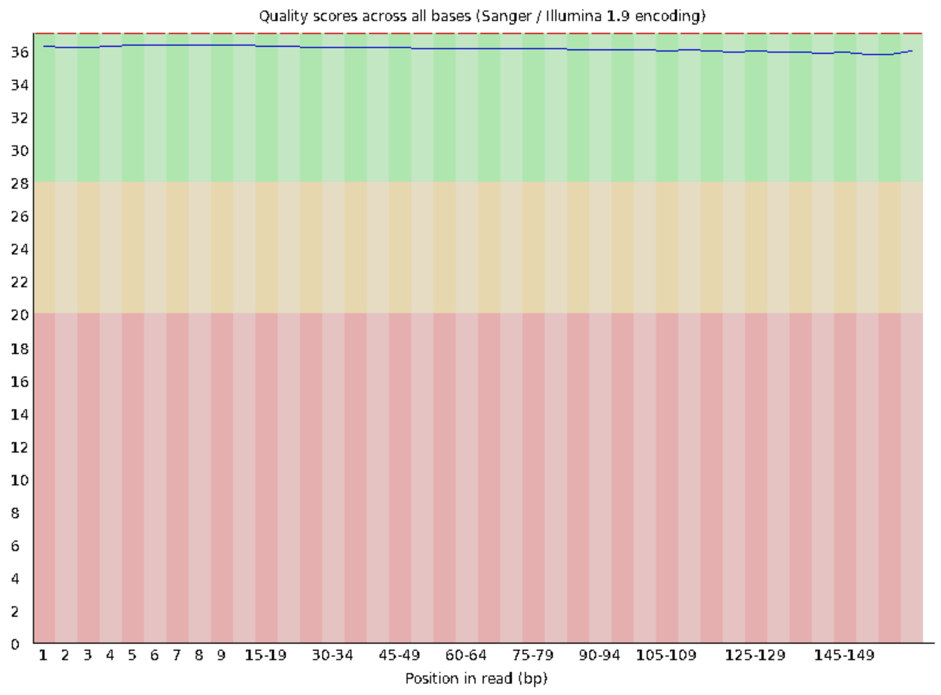


2. FastQC Results of Trimmed Reads:
    a. Forward Paired Reads:

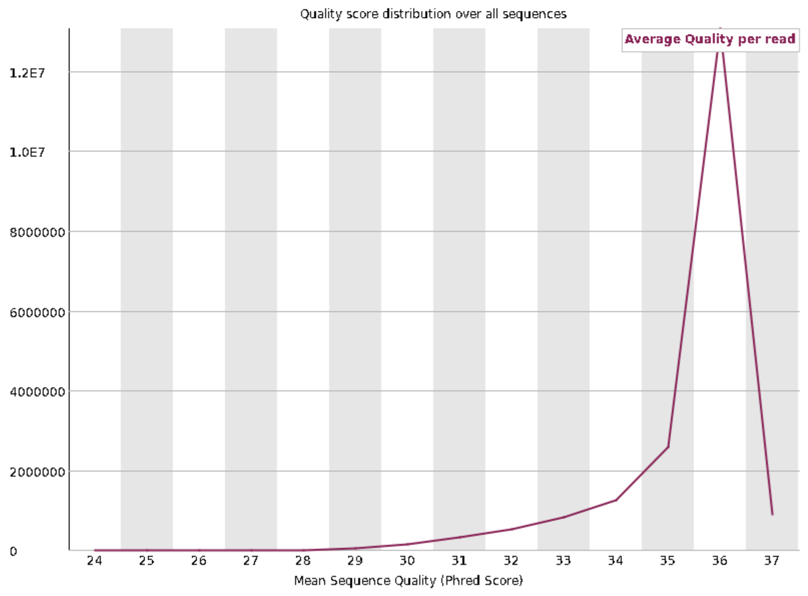
✓ Basic Statistics

Measure	Value
Filename	output_forward_paired.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	19817046
Total Bases	3 Gbp
Sequences flagged as poor quality	0
Sequence length	36-161
%GC	37

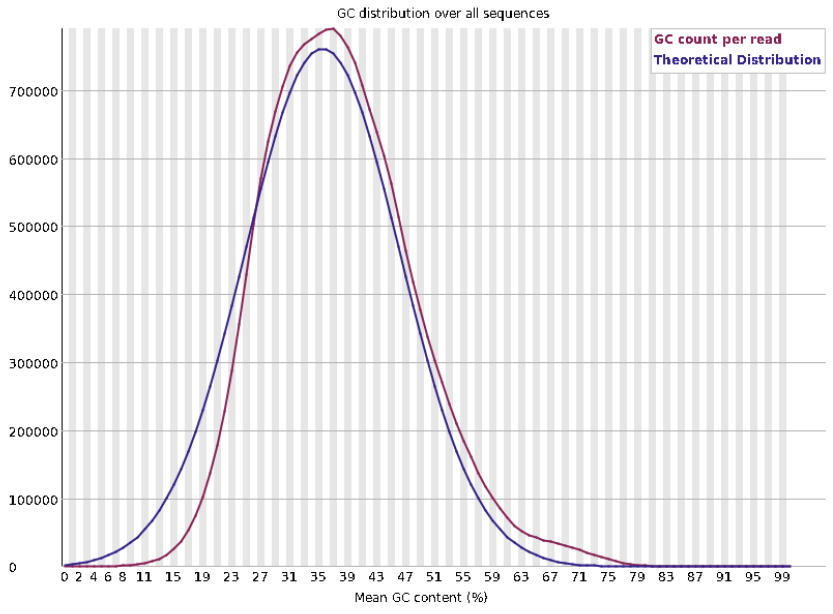
✓ Per base sequence quality



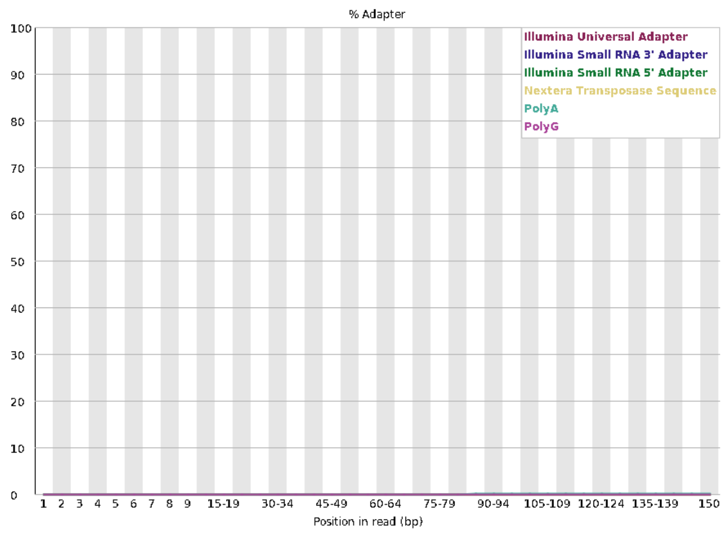
Per sequence quality scores



Per sequence GC content



✓ Adapter Content

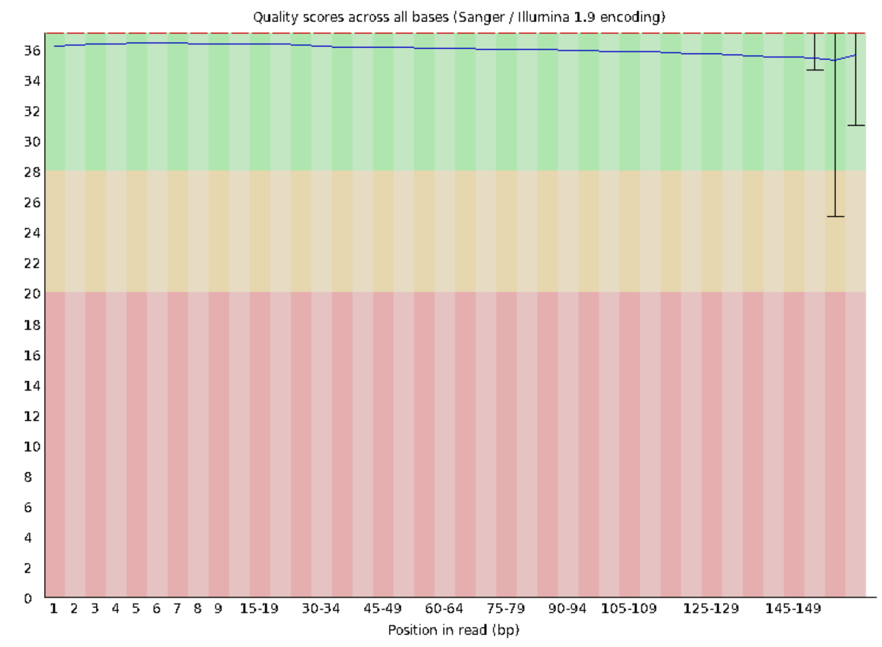


b. Reverse Paired Reads:

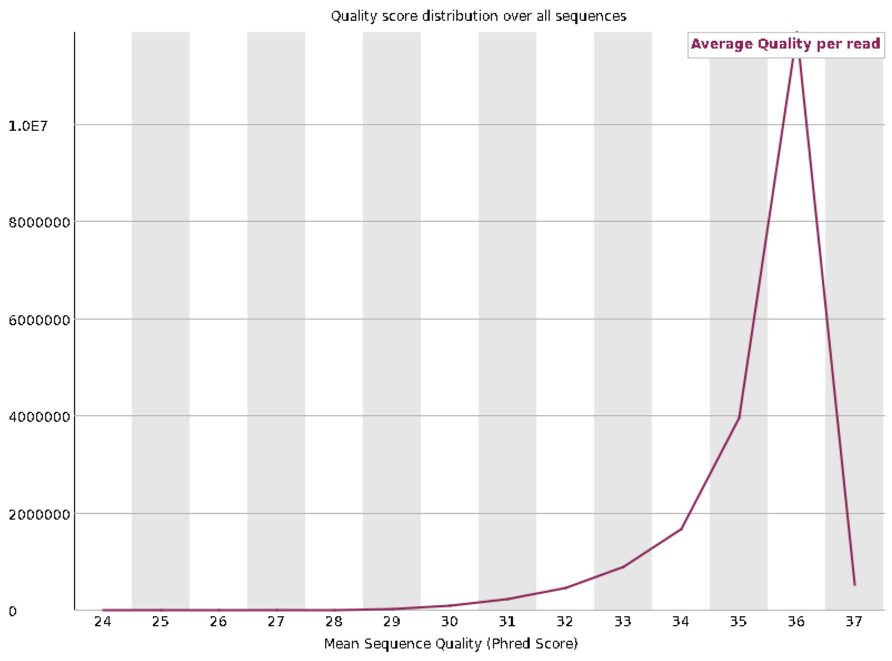
✓ Basic Statistics

Measure	Value
Filename	output_reverse_paired.fastq
File type	Conventional base calls
Encoding	Sanger / Illumina 1.9
Total Sequences	19817046
Total Bases	3 Gbp
Sequences flagged as poor quality	0
Sequence length	36-161
%GC	37

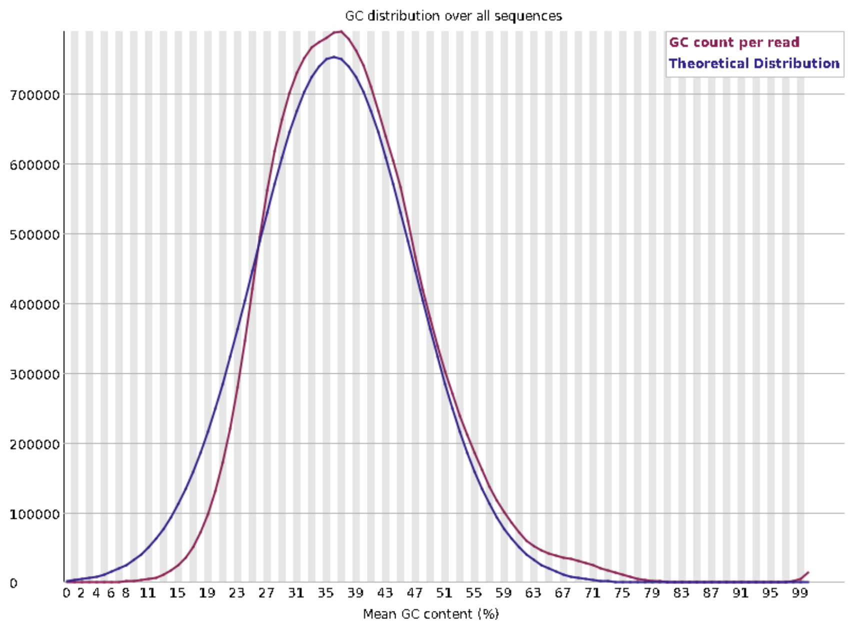
✓ Per base sequence quality



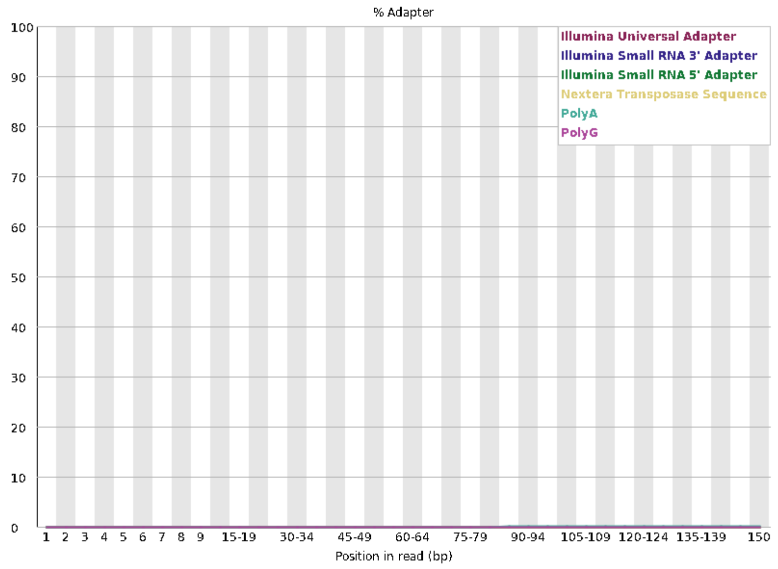
✓ Per sequence quality scores



✓Per sequence GC content



✓Adapter Content

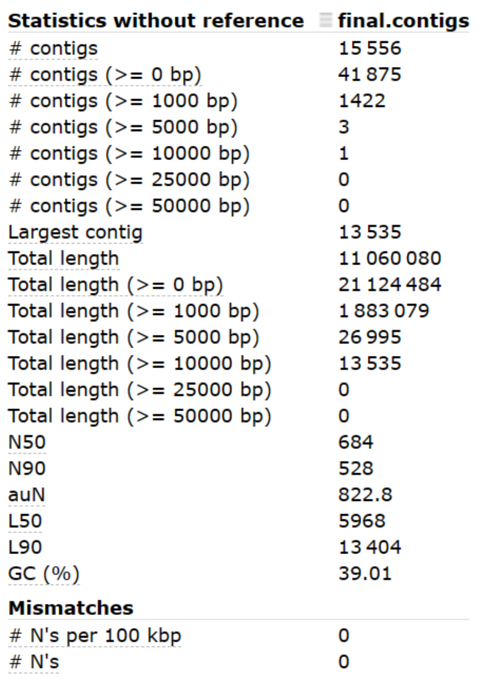


- Thus, comparing the above FastQC results from the raw and trimmed data, we can see that the number of reads decreases from 25 million to 19.8 million, indicating that around 5 million low quality reads were removed. We also observe that, on average, the per base sequence quality scores for the trimmed reads are higher than that of the raw data. The Adaptor Content in the raw data is around 10%, while that of the trimmed reads are about 0%, showing that the software has successfully removed the Adaptor sequences.

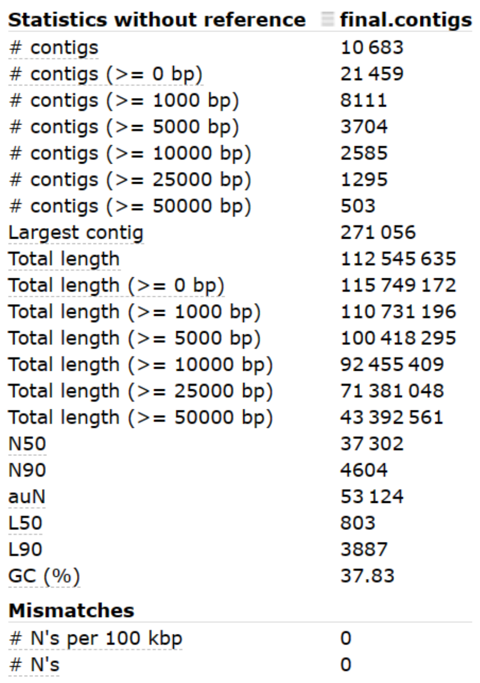
ANALYSIS:

Results for:

Half a Million Reads Assembly — 25 Million Reads:



Statistics without reference	final.contigs
# contigs	15 556
# contigs (>= 0 bp)	41 875
# contigs (>= 1000 bp)	1422
# contigs (>= 5000 bp)	3
# contigs (>= 10000 bp)	1
# contigs (>= 25000 bp)	0
# contigs (>= 50000 bp)	0
Largest contig	13 535
Total length	11 060 080
Total length (>= 0 bp)	21 124 484
Total length (>= 1000 bp)	1 883 079
Total length (>= 5000 bp)	26 995
Total length (>= 10000 bp)	13 535
Total length (>= 25000 bp)	0
Total length (>= 50000 bp)	0
N50	684
N90	528
auN	822.8
L50	5968
L90	13 404
GC (%)	39.01
Mismatches	
# N's per 100 kbp	0
# N's	0



Statistics without reference	final.contigs
# contigs	10 683
# contigs (>= 0 bp)	21 459
# contigs (>= 1000 bp)	8111
# contigs (>= 5000 bp)	3704
# contigs (>= 10000 bp)	2585
# contigs (>= 25000 bp)	1295
# contigs (>= 50000 bp)	503
Largest contig	271 056
Total length	112 545 635
Total length (>= 0 bp)	115 749 172
Total length (>= 1000 bp)	110 731 196
Total length (>= 5000 bp)	100 418 295
Total length (>= 10000 bp)	92 455 409
Total length (>= 25000 bp)	71 381 048
Total length (>= 50000 bp)	43 392 561
N50	37 302
N90	4604
auN	53 124
L50	803
L90	3887
GC (%)	37.83
Mismatches	
# N's per 100 kbp	0
# N's	0

The two assemblies compared are the one which is done using 25 millions reads, and one which is done by randomly sampling 500,000 out of the 25 million reads. Since the number of reads used is lesser, it will result in a smaller amount of the genome being assembled, as we have an overall smaller amount of data obtained from the sample. Thus, using a lesser number of reads leads to a less accurate assembly. We can see this in the 3 values tabulated below:

	Half a Million Reads	25 Million Reads
Number of Contigs (≥500bp)	15,556	10,683
N50	13,535	37,302
Length of Assembled Genome	11,060,080	112,545,635

1. Number of Contigs: We can see that the assembly which is done with more reads produces fewer contigs. Contigs are contiguous sequences of DNA that are assembled from the reads. With more reads, it is easier for the software to extend contigs and merge overlapping sequences, resulting in a reduced number of contigs and more coverage of the genome. Therefore, with less number of reads(half a million), there are more contigs, indicating that the resulting genome would be more fragmented since enough overlaps were not found.

2. N50 Value: The N50 number is a value that describes the contiguity of an assembly. It represents the contig length such that using equal or longer contigs will produce half of the bases of the assembly. When assembly is done with 25 million reads, the N50 value is more, implying that there are longer and more contiguous sequences in the assembly. Having a high N50 is an indicator of a good quality assembly. However, the N50 of the half million reads assembly is quite low, indicating a lower quality.

3. Length of Genome Assembled: When assembly is done with 25 million reads, it leads to longer contigs and, thus, a more complete assembly. Thus, the resulting assembled genome is longer. When assembly is done with half a million reads, the length of the genome assembled is much less, due to the fact that there is lesser coverage of the genome, resulting in the possibility for more fragmentation of the genome and more gaps being present in the sequence.

Thus, we can see that the genome assembly from the 25 million reads is of better quality than that of the subsampled half a million reads assembly. However, increasing the reads beyond that does not guarantee a higher quality of assembly and thus finding the optimal number of reads for assembly such that the quality is high but the time taken for assembly is not too much is an important step in the assembly process.