Bioinformatics Lessons Schedule

- RNA-seq
- single cell RNA-seq
- RRBS

Date	Subject
12-24	Christmas break
12-31	Christmas break
01-07	Process RNA-seq
01-14	Process RNA-seq, continued
01-21	Process RNA-seq, continued
01-28	Analyze RNA-seq
01-28	Analyze RNA-seq, continued

RNA-seq

Quality Check

FastQC

- Before going forward, we want to check the quality of the data
  - How much did the sequencer fail?
  - Did we sequence mostly our sample DNA?
- FastQC is a program from the Babraham Institute in the UK that creates an html report on the quality of the sequencing data
  - Has 11 quality control checks that it does

Basic Statistics

Good Quality

Basic Statistics

Measure	Value
Filename	good_sequence_short.txt
File type	Conventional base calls
Encoding	Illumina 1.5
Total Sequences	250000
Sequences flagged as poor quality	0
Sequence length	40
%GC	45

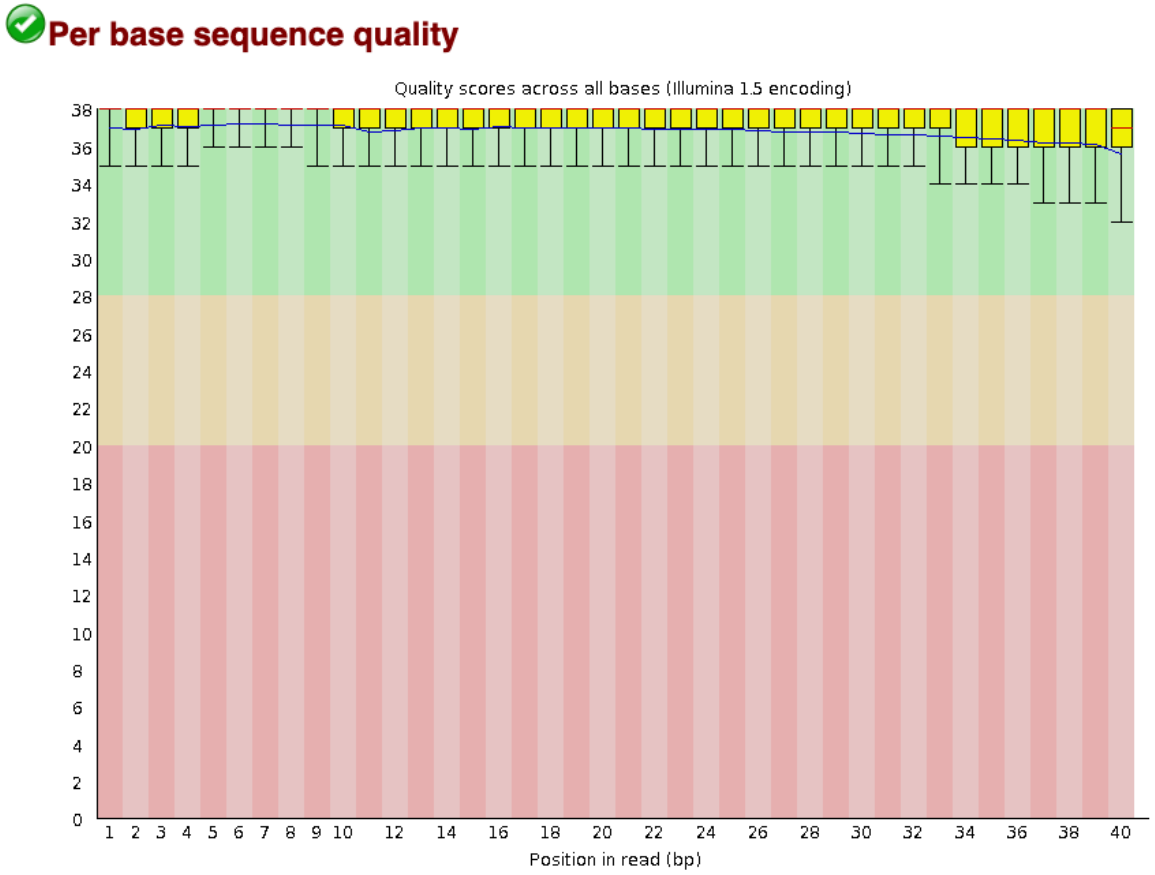
Bad Quality

Basic Statistics

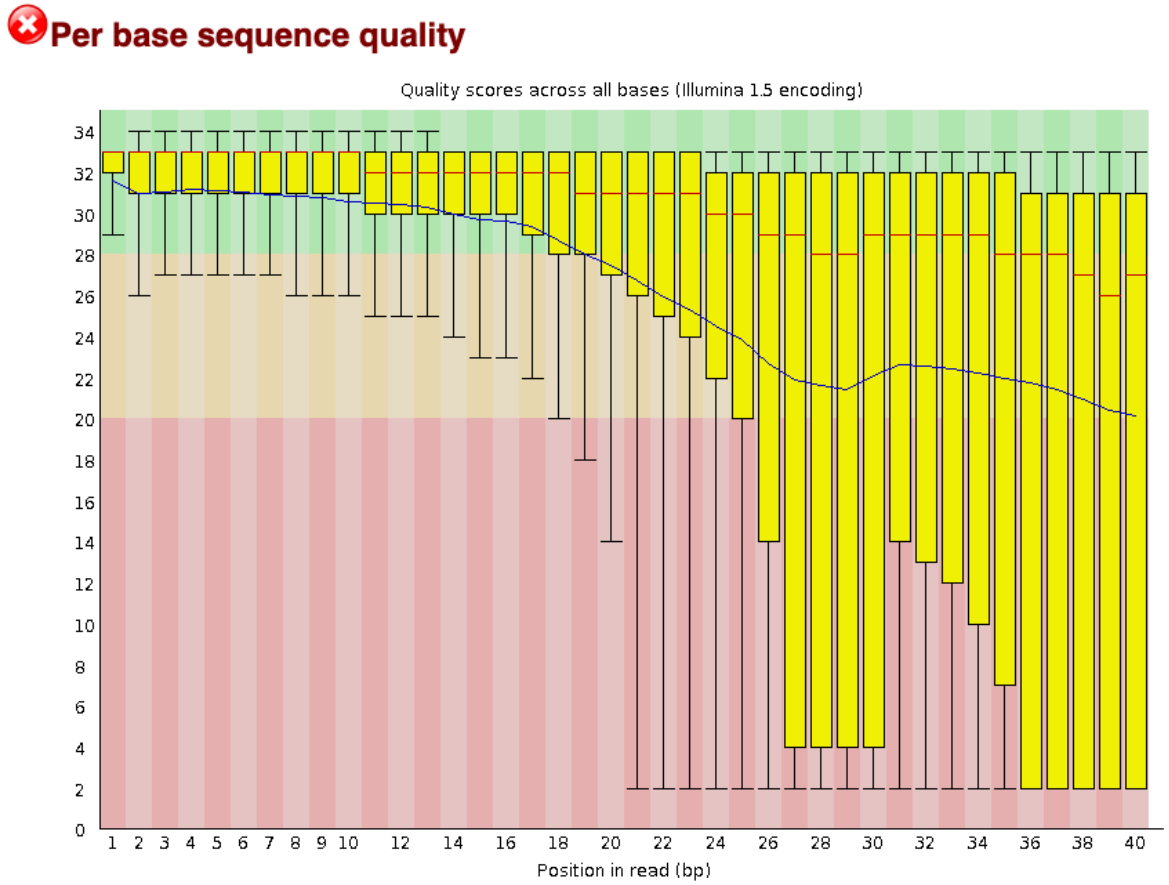
Measure	Value
Filename	bad_sequence.txt
File type	Conventional base calls
Encoding	Illumina 1.5
Total Sequences	395288
Sequences flagged as poor quality	0
Sequence length	40
%GC	47

Per base sequence quality

Good Quality

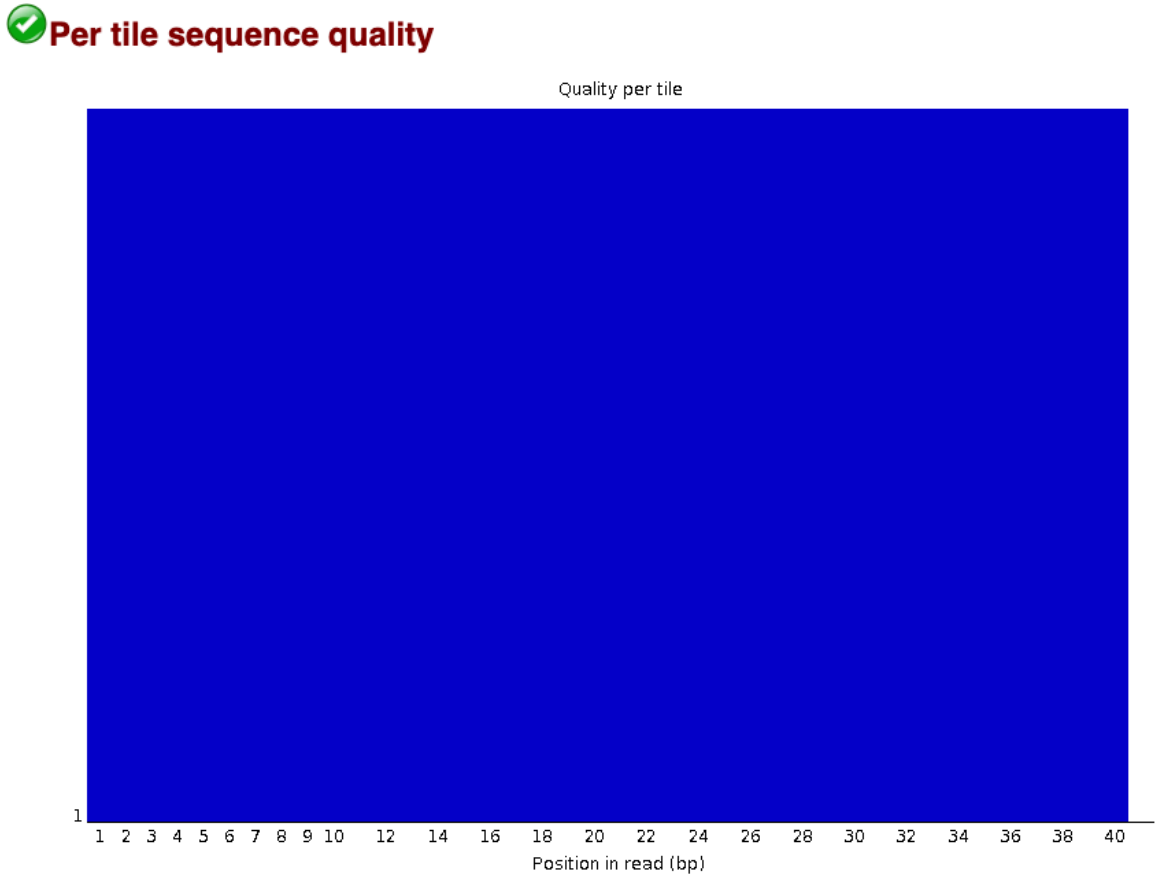


Bad Quality

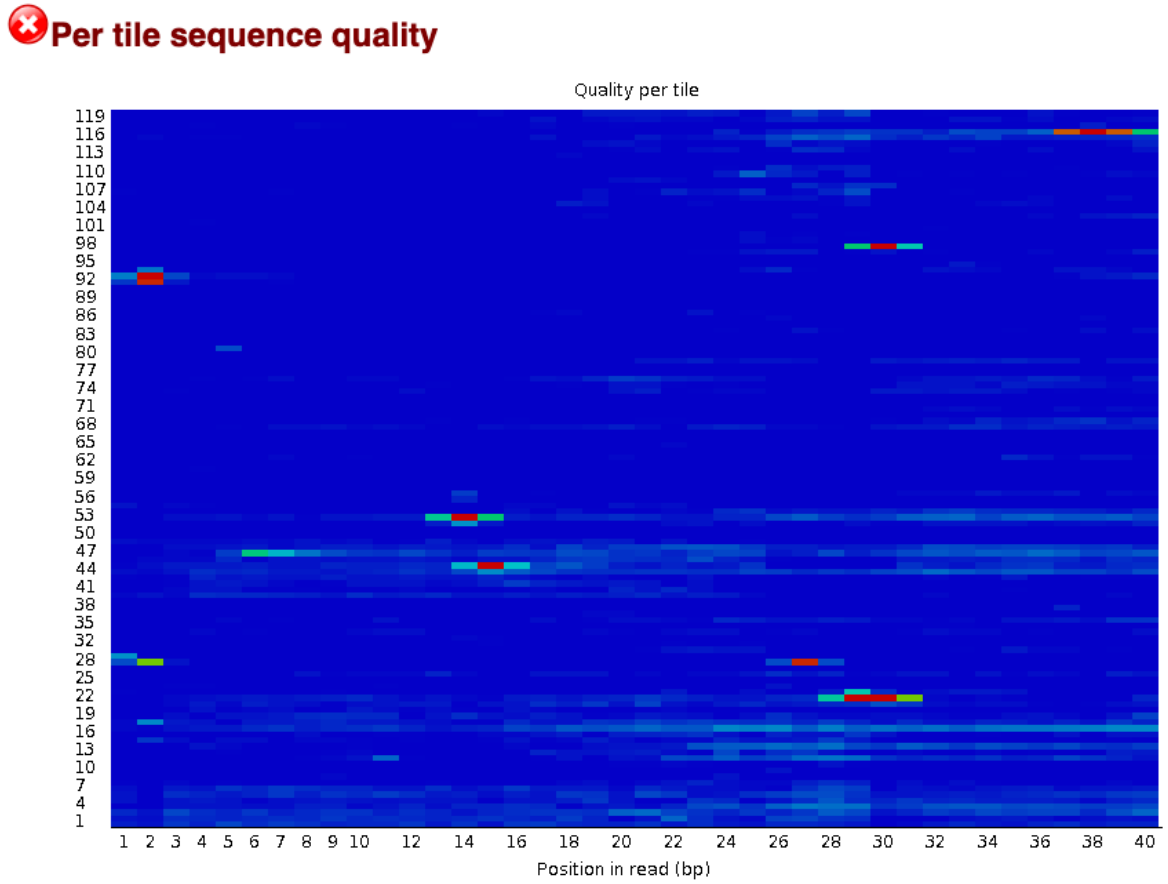


Per tile sequence quality

Good Quality

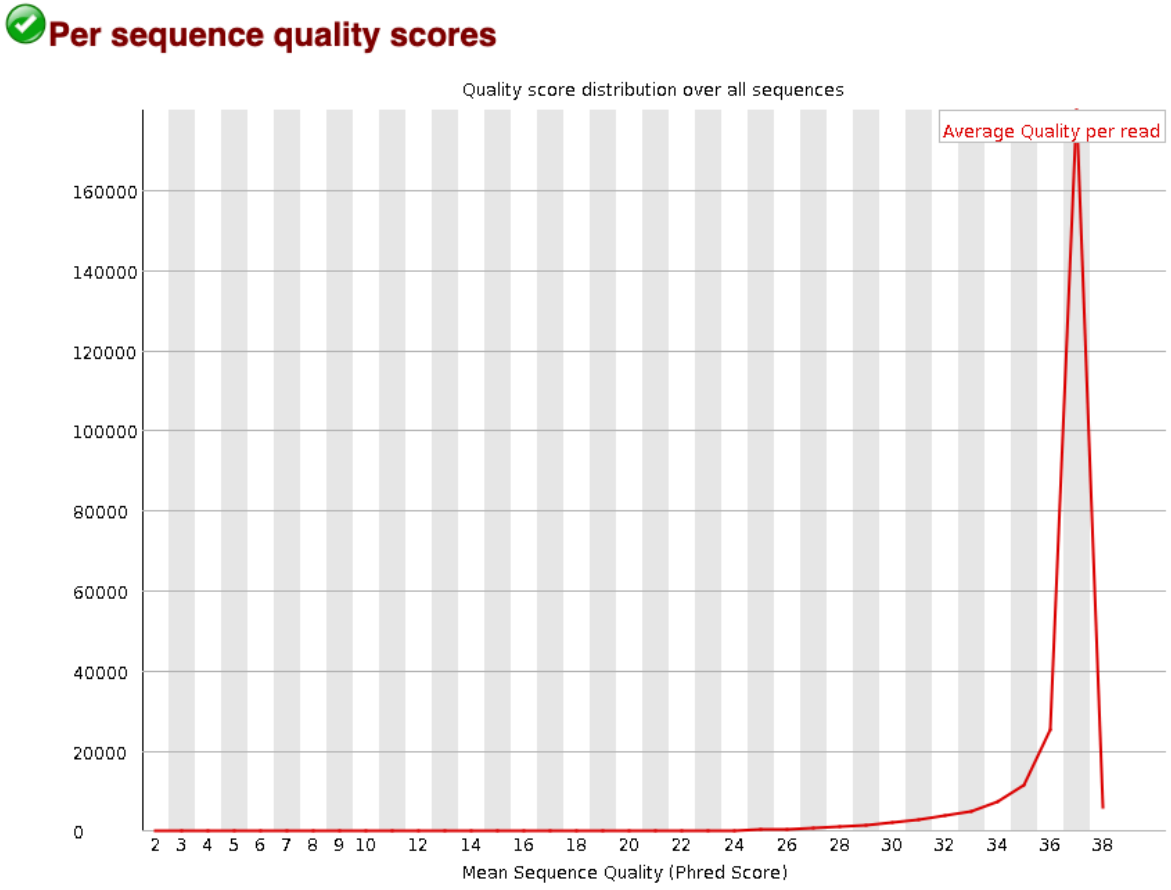


Bad Quality

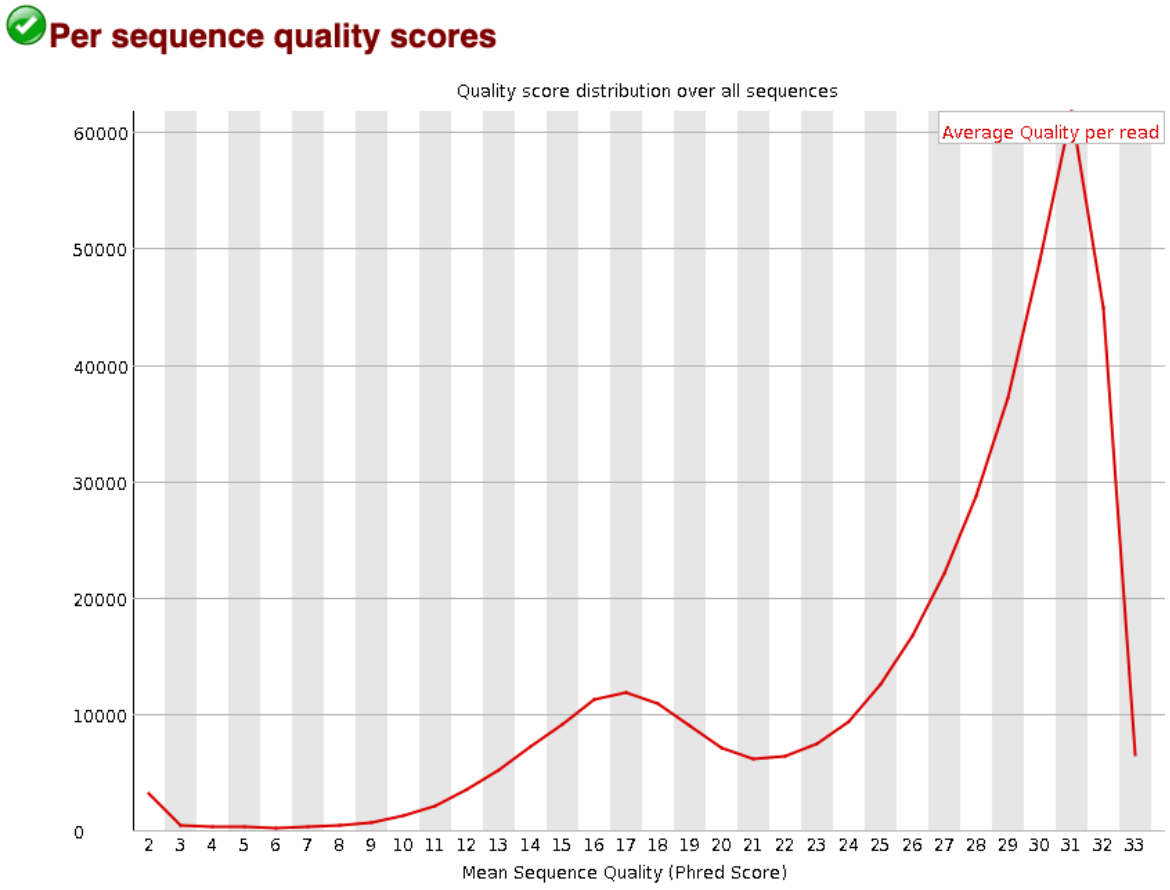


Per sequence quality scores

Good Quality

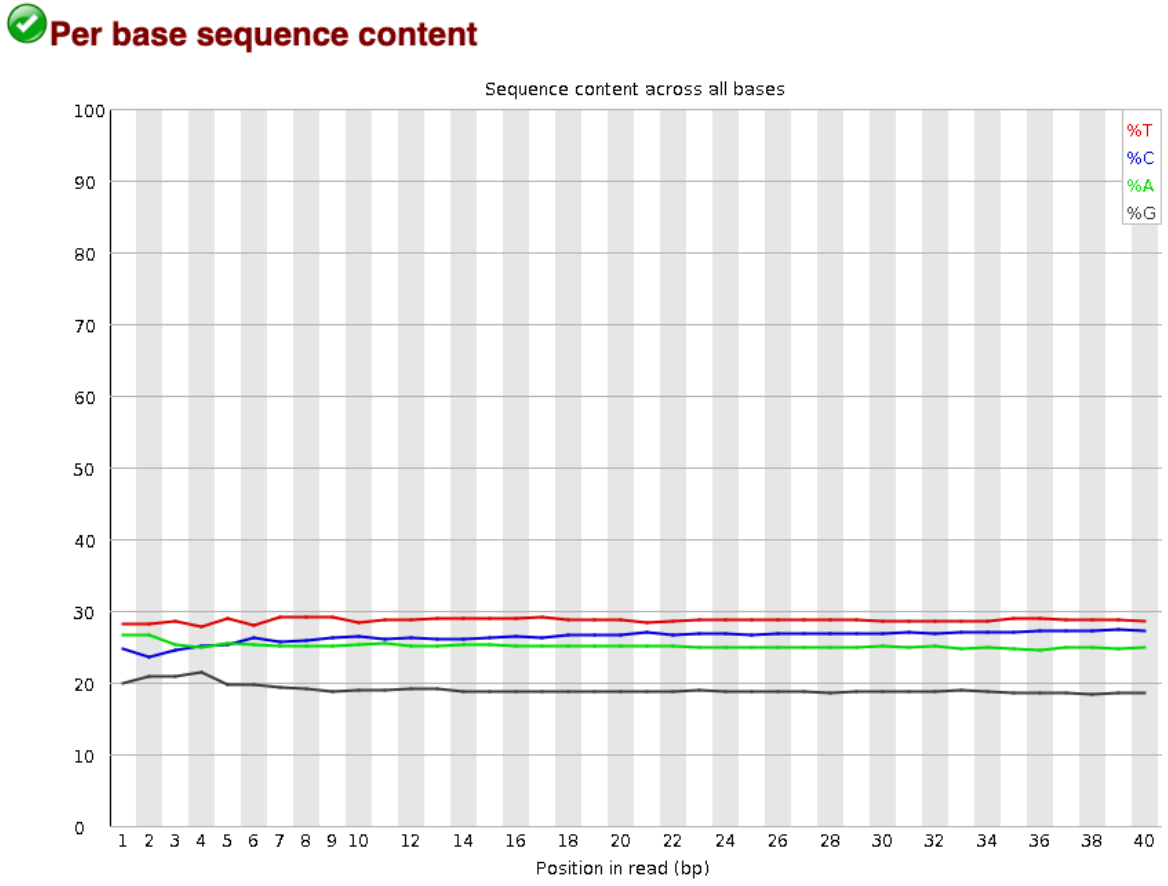


Bad Quality

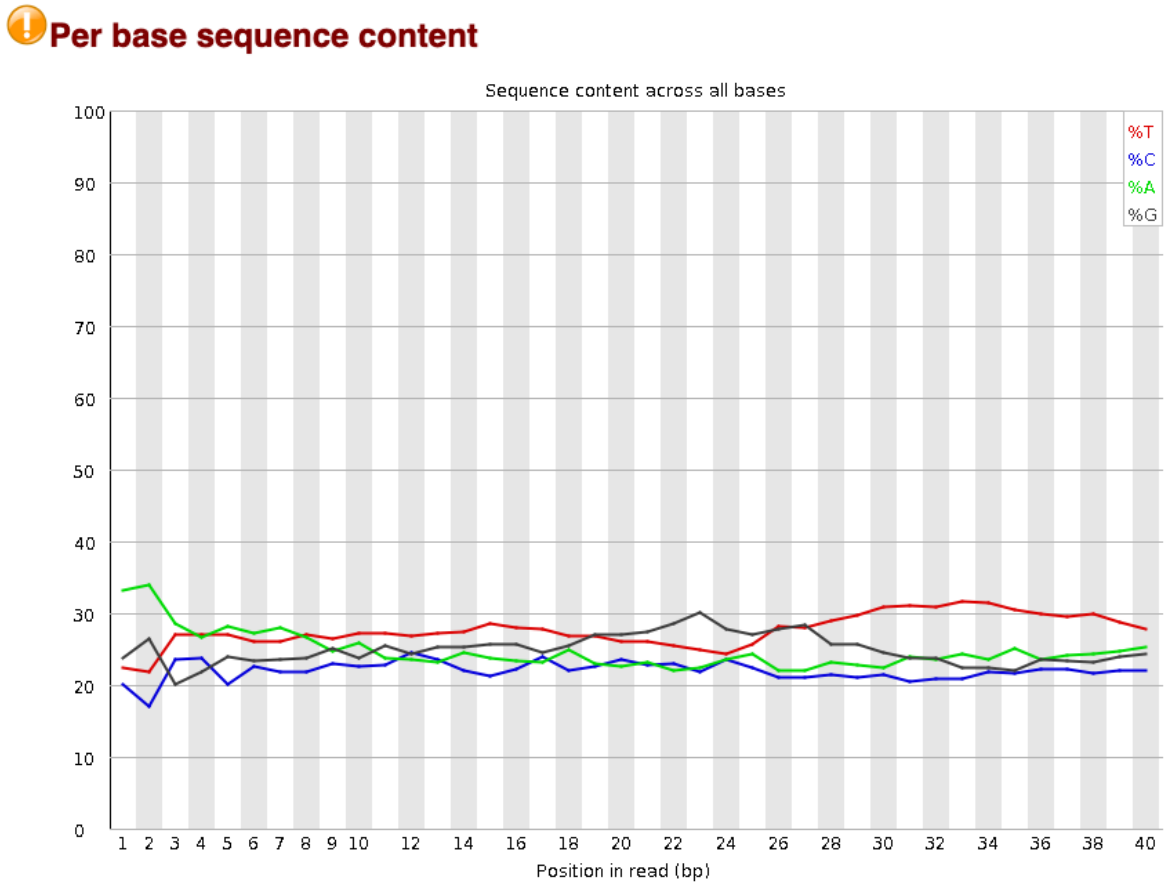


Per base sequence content

Good Quality

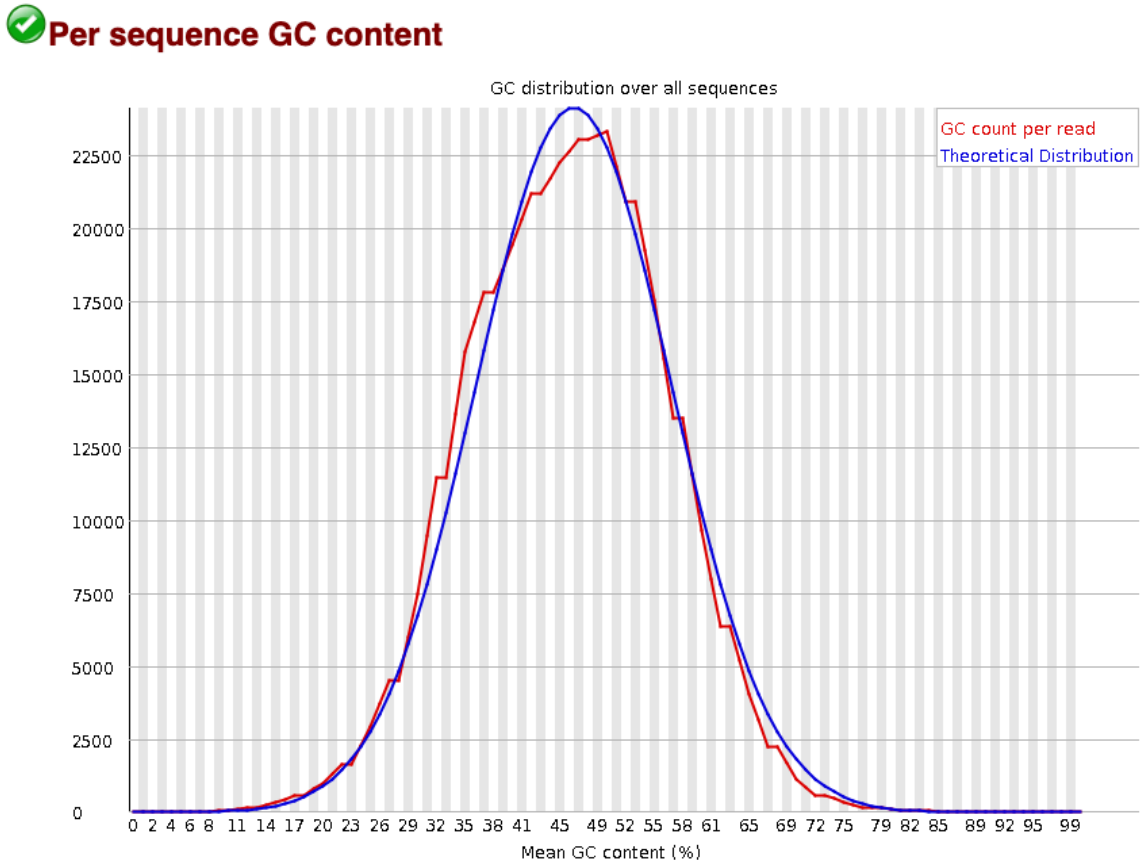


Bad Quality

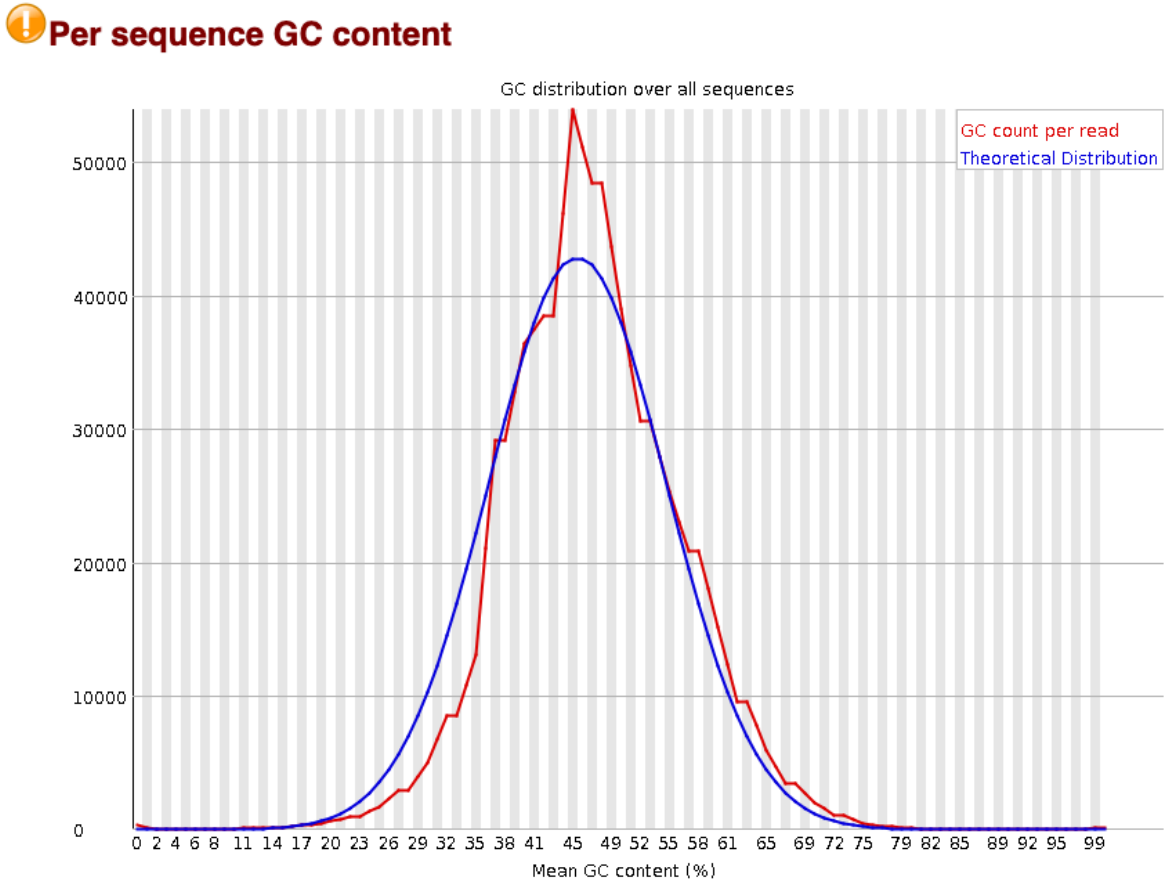


Per sequence GC content

Good Quality

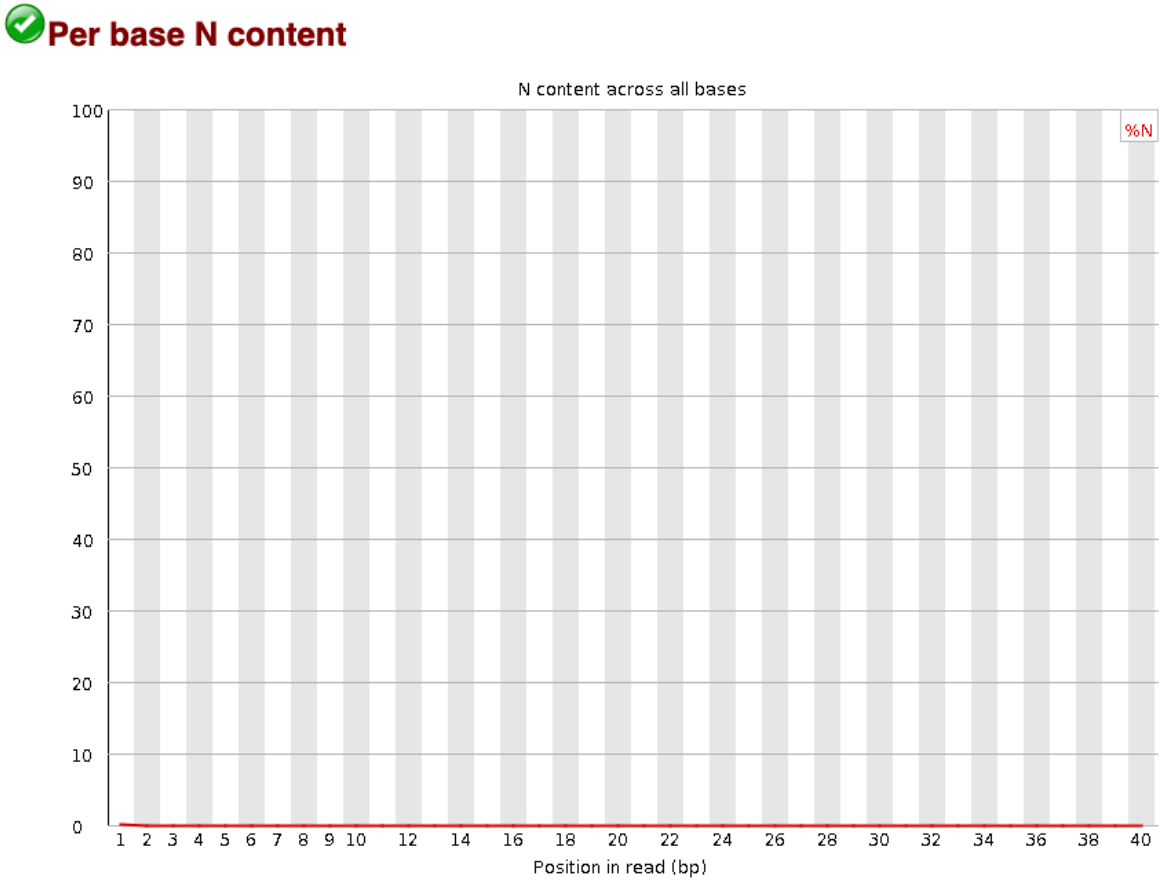


Bad Quality

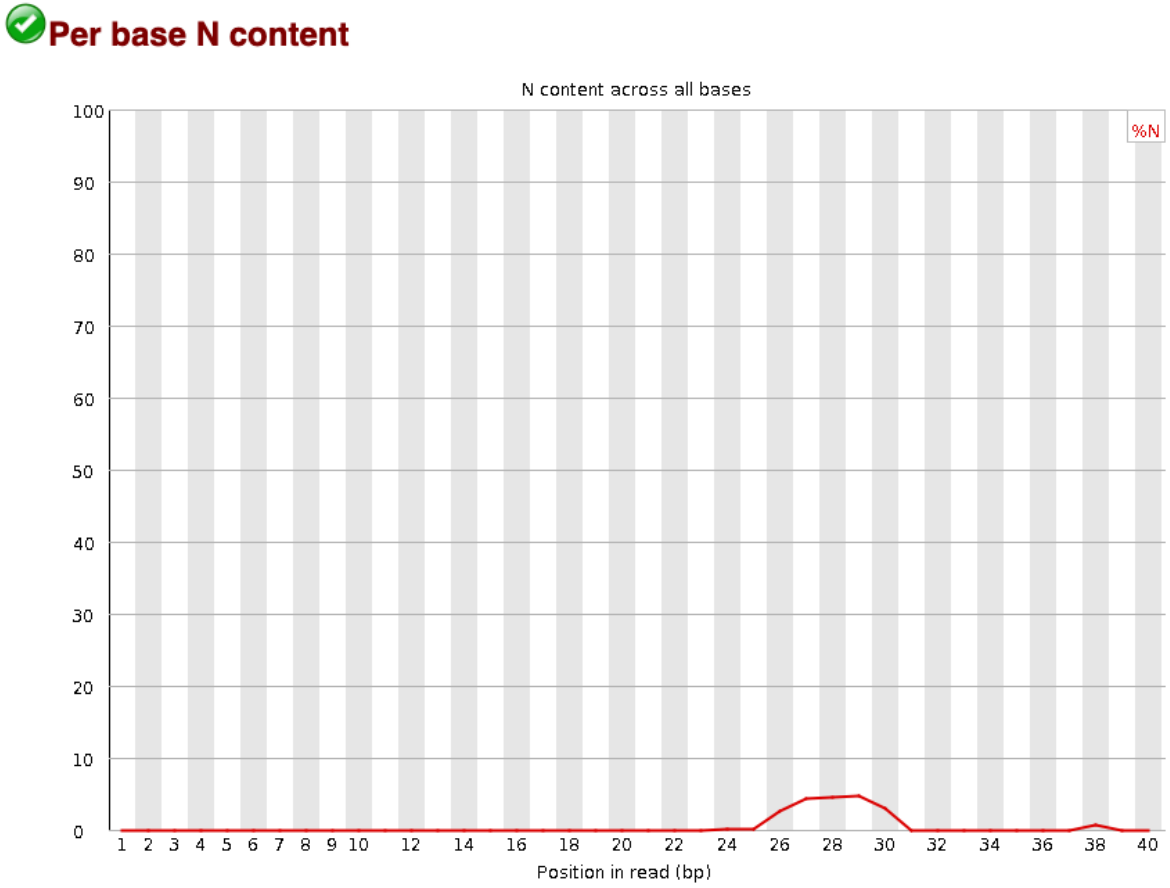


Per base sequence quality

Good Quality

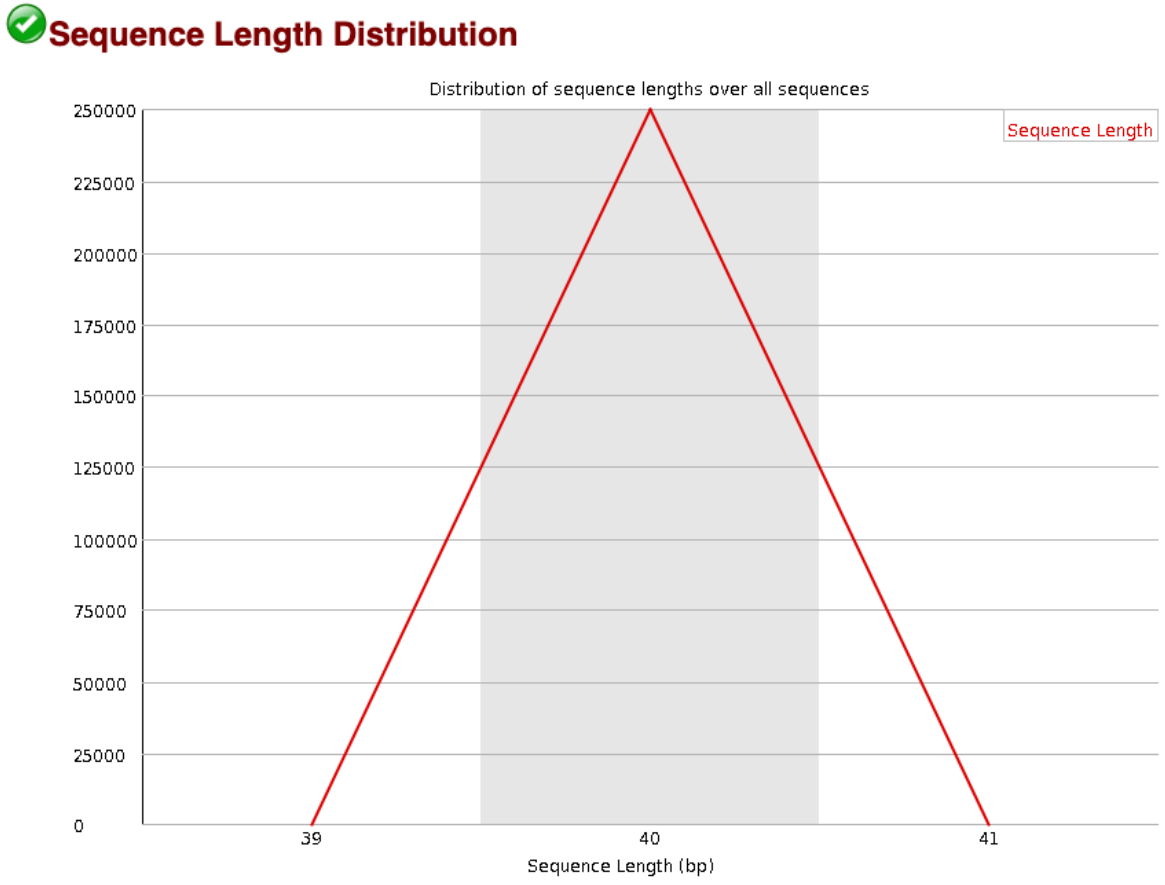


Bad Quality

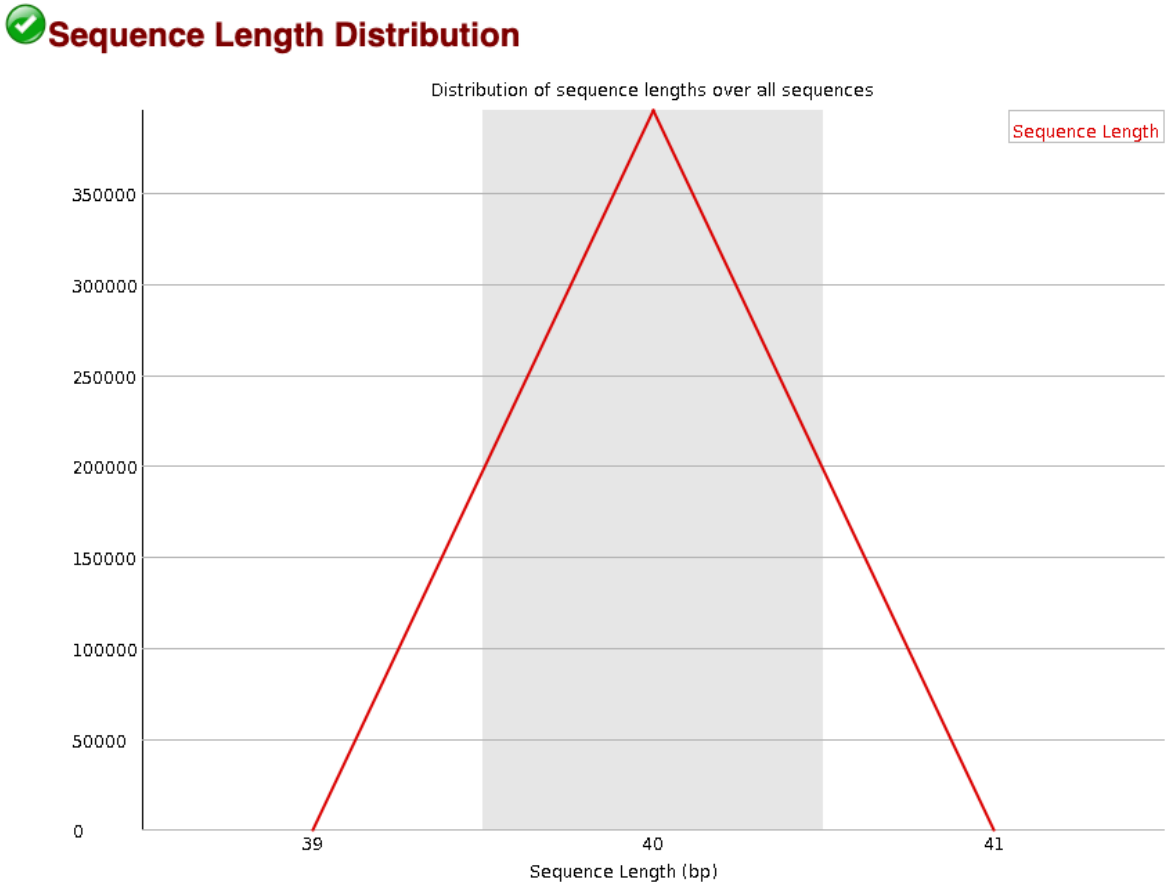


Per base sequence quality

Good Quality

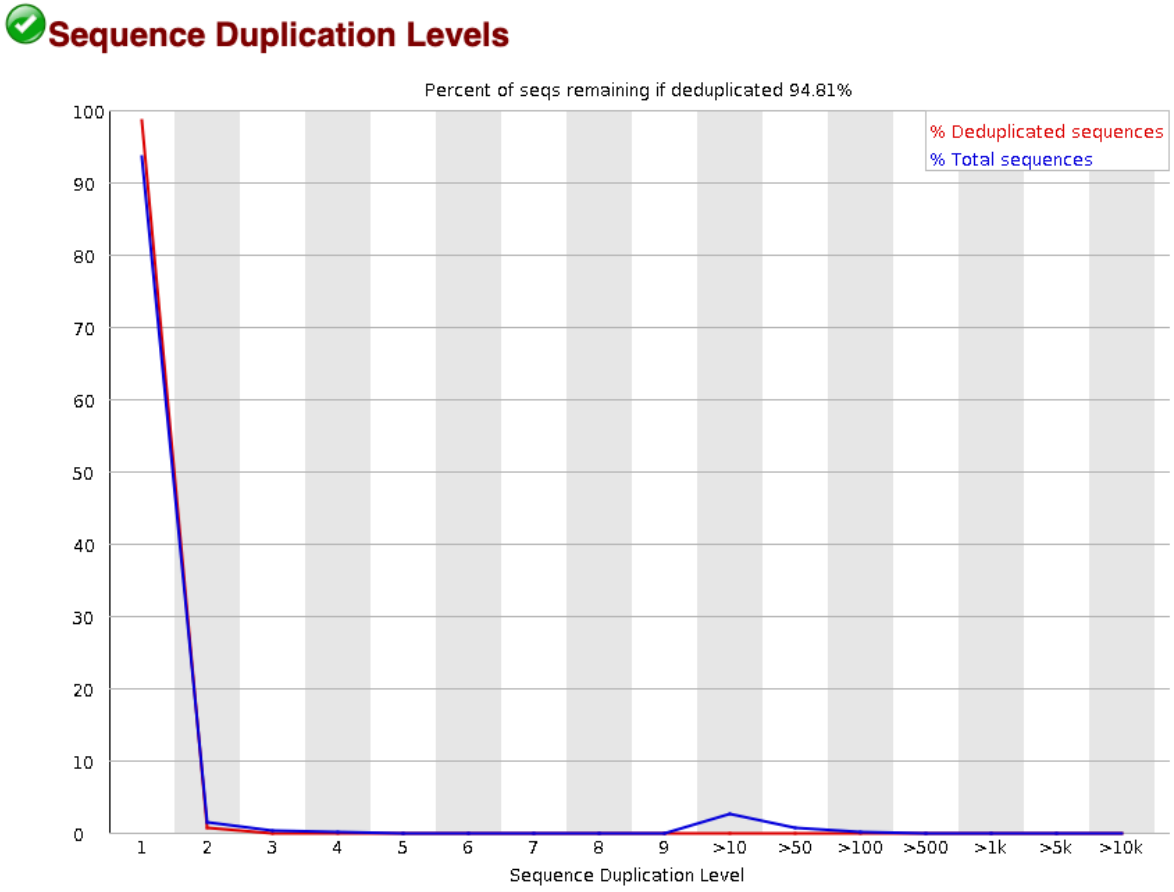


Bad Quality

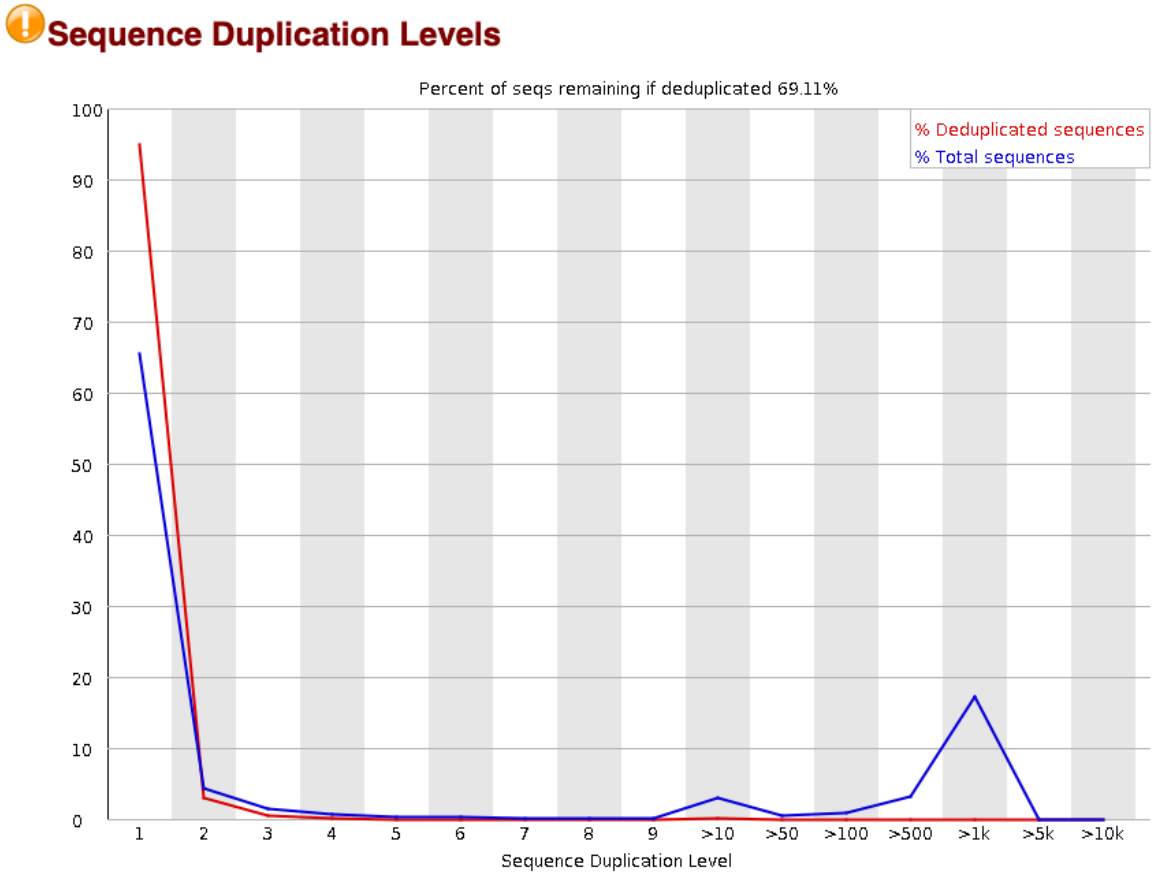


Sequence Duplication Levels

Good Quality



Bad Quality



Overrepresented sequences

Good Quality

Overrepresented sequences

No overrepresented sequences

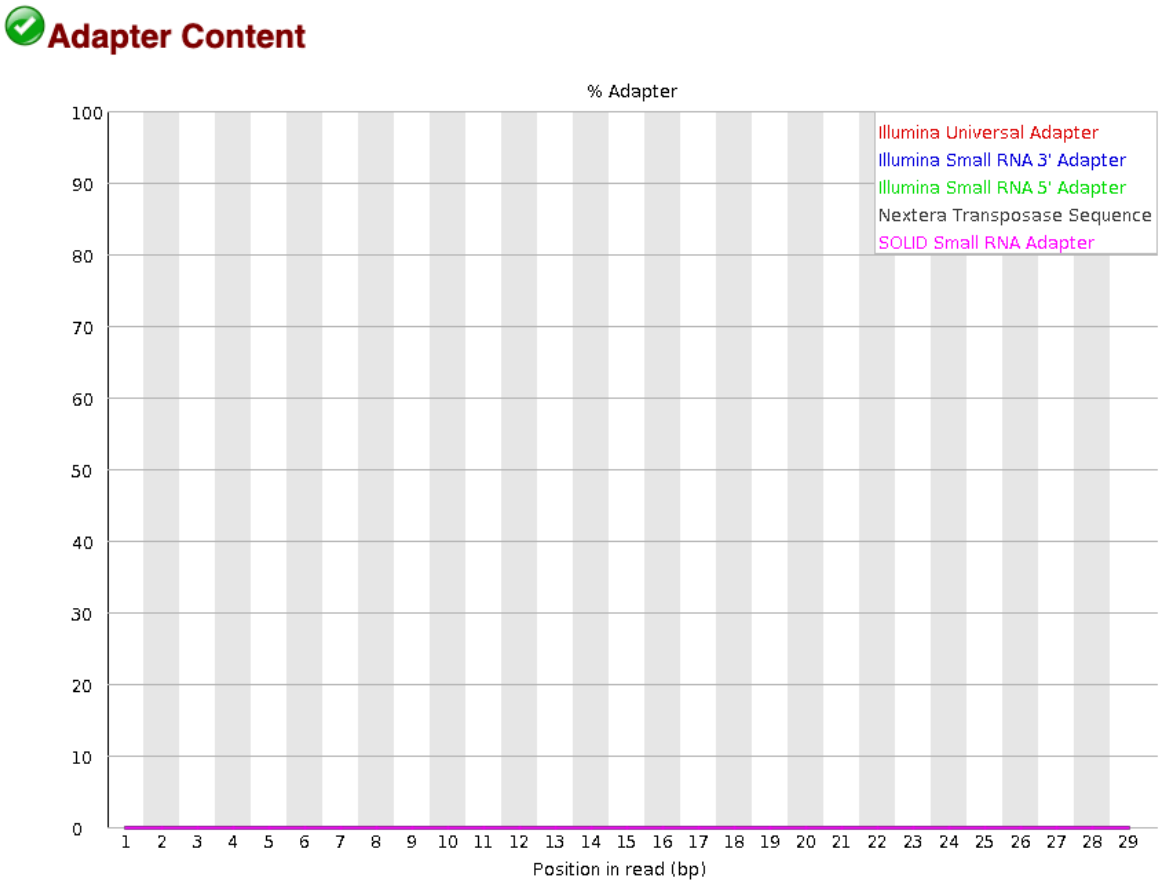
Bad Quality

Overrepresented sequences

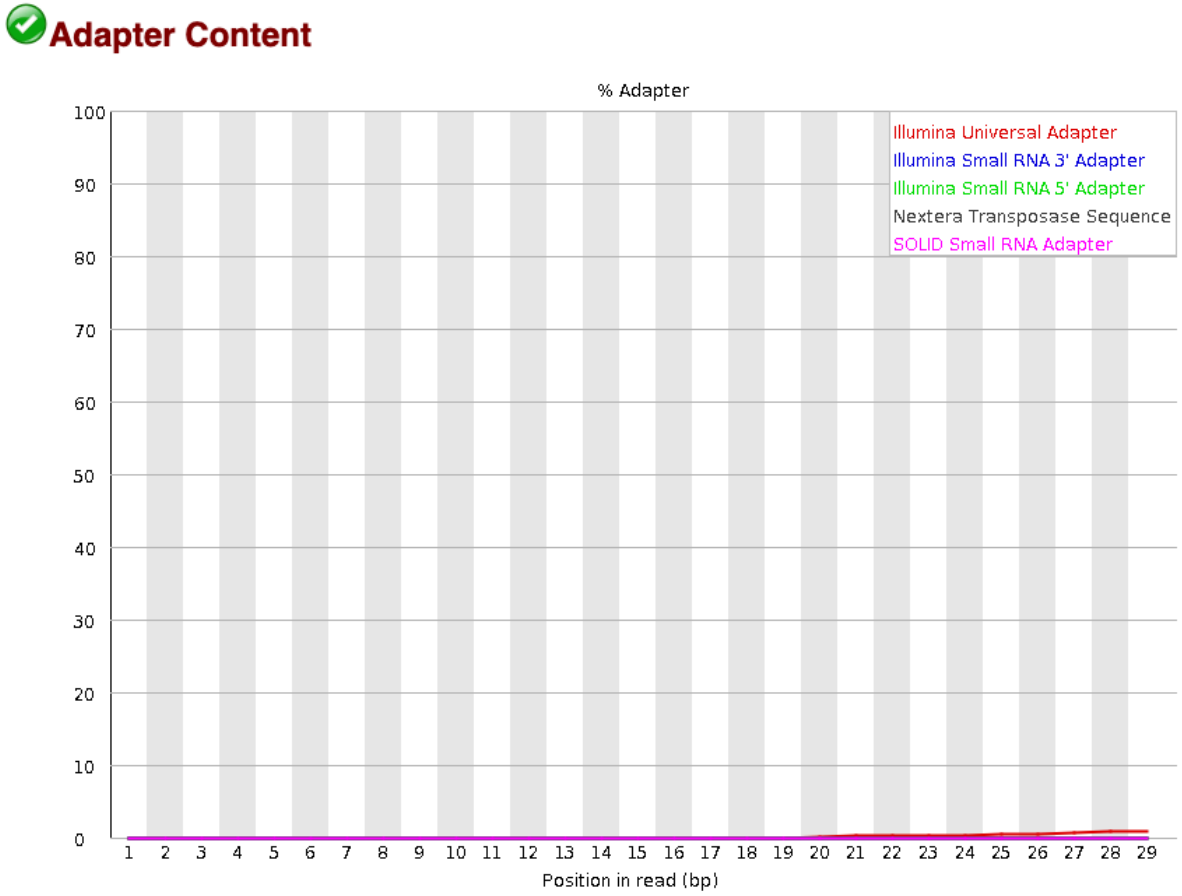
Sequence	Count	Percentage	Possible Source
AGAGTTTTATCGCTTCCATGACGCAGAAGTTAACACTTTC	2065	0.5224039181558763	No Hit
GATTGGCGTATCCAACCTGCAGAGTTTTATCGCTTCCATG	2047	0.5178502762542754	No Hit
ATTGGCGTATCCAACCTGCAGAGTTTTATCGCTTCCATGA	2014	0.5095019327680071	No Hit
CGATAAAAATGATTGGCGTATCCAACCTGCAGAGTTTTAT	1913	0.4839509420979134	No Hit
GTATCCAACCTGCAGAGTTTTATCGCTTCCATGACGCAGA	1879	0.4753496185060006 6	No Hit
AAAAATGATTGGCGTATCCAACCTGCAGAGTTTTATCGCT	1846	0.4670012750197325	No Hit
TGATTGGCGTATCCAACCTGCAGAGTTTTATCGCTTCCAT	1841	0.4657363744915099 5	No Hit
AACCTGCAGAGTTTTATCGCTTCCATGACGCAGAAGTTAA	1836	0.4644714739632875 3	No Hit
GATAAAAATGATTGGCGTATCCAACCTGCAGAGTTTTATC	1831	0.463206573435065 1	No Hit
AAATGATTGGCGTATCCAACCTGCAGAGTTTTATCGCTTC	1779	0.4500516079415514 7	No Hit
ATGATTGGCGTATCCAACCTGCAGAGTTTTATCGCTTCCA	1779	0.4500516079415514 7	No Hit
AATGATTGGCGTATCCAACCTGCAGAGTTTTATCGCTTCC	1760	0.445244985934306 1	No Hit
AAAATGATTGGCGTATCCAACCTGCAGAGTTTTATCGCTT	1729	0.437402602659326 9	No Hit
CGTATCCAACCTGCAGAGTTTTATCGCTTCCATGACGCAG	1713	0.433354920969014 96	No Hit
ATCCAACCTGCAGAGTTTTATCGCTTCCATGACGCAGAAG	1708	0.432090020440792 53	No Hit
CAGAGTTTTATCGCTTCCATGACGCAGAAGTTAACACTTT	1684	0.426018497905324 76	No Hit
TGCAGAGTTTTATCGCTTCCATGACGCAGAAGTTAACACT	1668	0.42197081621501 28	No Hit
CAACCTGCAGAGTTTTATCGCTTCCATGACGCAGAAGTTA	1668	0.42197081621501 28	No Hit
TATCCAACCTGCAGAGTTTTATCGCTTCCATGACGCAGAA	1630	0.412357572200522 1	No Hit
GTCATGGAAGCGATAAAACTCTGCAGGTTGGATACGCCAA	1620	0.409827771144077 26	No Hit
AACTTCTGCGTCATGGAAGCGATAAAACTCTGCAGGTTGG	1616	0.40881585072149 93	No Hit
GCAGAGTTTTATCGCTTCCATGACGCAGAAGTTAACACTT	1580	0.399708566918297 54	No Hit
TGGCGTATCCAACCTGCAGAGTTTTATCGCTTCCATGACG	1569	0.39692578575620 82	No Hit
GGCGTATCCAACCTGCAGAGTTTTATCGCTTCCATGACGC	1542	0.390095322903806 83	No Hit
ATAAAAATGATTGGCGTATCCAACCTGCAGAGTTTTATCG	1481	0.374663536459492 85	No Hit
ACCTGCAGAGTTTTATCGCTTCCATGACGCAGAAGTTAAC	1479	0.374157576248203 84	No Hit
ATGGAAGCGATAAAACTCTGCAGGTTGGATACGCCAATCA	1452	0.36732711339580 26	No Hit
GATAAAACTCTGCAGGTTGGATACGCCAATCATTTTTATC	1420	0.359231750015178 76	No Hit
CGTCATGGAAGCGATAAAACTCTGCAGGTTGGATACGCCA	1412	0.35720790917002 29	No Hit
ACTTCTGCGTCATGGAAGCGATAAAACTCTGCAGGTTGGA	1368	0.346067784521665 24	No Hit
TAACTTCTGCGTCATGGAAGCGATAAAACTCTGCAGGTTG	1363	0.344811883993442 76	No Hit
CATGGAAGCGATAAAACTCTGCAGGTTGGATACGCCAATC	1333	0.33722248082410 8	No Hit
CGATAAAACTCTGCAGGTTGGATACGCCAATCATTTTTAT	1304	0.329886057760417 74	No Hit
TAAAAATGATTGGCGTATCCAACCTGCAGAGTTTTATCGC	1277	0.323055594908016 44	No Hit
GCGTATCCAACCTGCAGAGTTTTATCGCTTCCATGACGCA	1262	0.319260893323349 06	No Hit
TGCGTCATGGAAGCGATAAAACTCTGCAGGTTGGATACGC	1233	0.31192447025965 88	No Hit
GGAAGCGATAAAACTCTGCAGGTTGGATACGCCAATCATT	1182	0.29902248487178 97	No Hit
AAGCGATAAAACTCTGCAGGTTGGATACGCCAATCATTTT	1136	0.28738540001214 31	No Hit
ACTCTGCAGGTTGGATACGCCAATCATTTTTATCGAAGCG	1133	0.286626459695209 56	No Hit
AAACTCTGCAGGTTGGATACGCCAATCATTTTTATCGAAG	1131	0.28612049948392 06	No Hit
AAAACTCTGCAGGTTGGATACGCCAATCATTTTTATCGAA	1129	0.28561453927263 16	No Hit
AGCGATAAAACTCTGCAGGTTGGATACGCCAATCATTTTT	1113	0.28156685758231 97	No Hit
ATAAAACTCTGCAGGTTGGATACGCCAATCATTTTTATCG	1111	0.28106089737103 74	No Hit
AACTCTGCAGGTTGGATACGCCAATCATTTTTATCGAAGC	1083	0.27397745441298 5	No Hit
CTGCAGAGTTTTATCGCTTCCATGACGCAGAAGTTAACAC	1055	0.26689401145493 92	No Hit
TTCTGCGTCATGGAAGCGATAAAACTCTGCAGGTTGGATA	947	0.23957216004533 402	No Hit
TGGAAGCGATAAAACTCTGCAGGTTGGATACGCCAATCAT	946	0.23931917993968 954	No Hit
TAAAACTCTGCAGGTTGGATACGCCAATCATTTTTATCGA	912	0.23071785634777 68	No Hit
GAAGCGATAAAACTCTGCAGGTTGGATACGCCAATCATTT	888	0.22464633381230 9	No Hit
GCGTCATGGAAGCGATAAAACTCTGCAGGTTGGATACGCC	805	0.203648985043816 15	No Hit
GCGATAAAACTCTGCAGGTTGGATACGCCAATCATTTTTA	785	0.198589382930926 32	No Hit
TTGGCGTATCCAACCTGCAGAGTTTTATCGCTTCCATGAC	784	0.198336402825281 8	No Hit
CTTCTGCGTCATGGAAGCGATAAAACTCTGCAGGTTGGAT	762	0.19277084050110 3	No Hit
TCCAACCTGCAGAGTTTTATCGCTTCCATGACGCAGAAGT	752	0.190241039444658 06	No Hit
CCAACCTGCAGAGTTTTATCGCTTCCATGACGCAGAAGTT	744	0.188217198599502 12	No Hit
TCATGGAAGCGATAAAACTCTGCAGGTTGGATACGCCAAT	665	0.168231770253587 26	No Hit
TCTGCGTCATGGAAGCGATAAAACTCTGCAGGTTGGATAC	627	0.158618526239096 56	No Hit
CCTGCAGAGTTTTATCGCTTCCATGACGCAGAAGTTAACA	613	0.155076804760073 66	No Hit
CGGTTCAGCAGGAATGCCGAGATCGGAAGAGCGGTTCAGC	599	0.151535083281050 78	Illumina Paired End PCR Primer 2 (96% over 25bp)
TCTGCAGGTTGGATACGCCAATCATTTTTATCGAAGCGCG	585	0.14799336180202 79	No Hit
CGCTTAAAGCTACCAGTTATATGGCTGGGGGGTTTTTTTT	552	0.13964501831575 965	No Hit
CTCTGCAGGTTGGATACGCCAATCATTTTTATCGAAGCGC	532	0.13458541620286 98	No Hit
CTGCGTCATGGAAGCGATAAAACTCTGCAGGTTGGATACG	515	0.13028475440691 342	No Hit
CTGCAGGTTGGATACGCCAATCATTTTTATCGAAGCGCGC	505	0.12775495335046 852	No Hit
GCTTAAAGCTACCAGTTATATGGCTGGGGGGTTTTTTTTG	411	0.10397482341988 626	No Hit

Adapter Content

Good Quality



Bad Quality



Sidenote - Loops in Bash

What’s a loop?

What’s a loop?

A loop is a piece of code that repeatedly executes a command for a certain condition until that condition is no longer true.

What’s a loop?

A loop is a piece of code that repeatedly executes a command for a certain condition until that condition is no longer true.

```
for i in *.fastq.gz;
   do echo $i;
done
```

What's a loop?

A loop is a piece of code that repeatedly executes a command for a certain condition until that condition is no longer true.

```
for i in *.fastq.gz;
   do echo $i;
done
```

condition

What's a loop?

A loop is a piece of code that repeatedly executes a command for a certain condition until that condition is no longer true.

```
for i in *.fastq.gz;
   do echo $i;
done
```

condition

command

What's a loop?

A loop is a piece of code that repeatedly executes a command for a certain condition until that condition is no longer true.

```
for i in *.fastq.gz;
   do echo $i;
done
```

condition

command

stop

How do you write a loop in bash?

How do you write a loop in bash?

```
for i in *.fastq.gz;
   do echo $i;
done
```

How do you write a loop in bash?

starting a loop

```
for i in *.fastq.gz;
   do echo $i;
done
```

How do you write a loop in bash?

starting a loop

variable name

```
for i in *.fastq.gz;
   do echo $i;
done
```

How do you write a loop in bash?

starting a loop

variable name

meeting this condition

```
for i in *.fastq.gz;
    do echo $i;
done
```

How do you write a loop in bash?

starting a loop

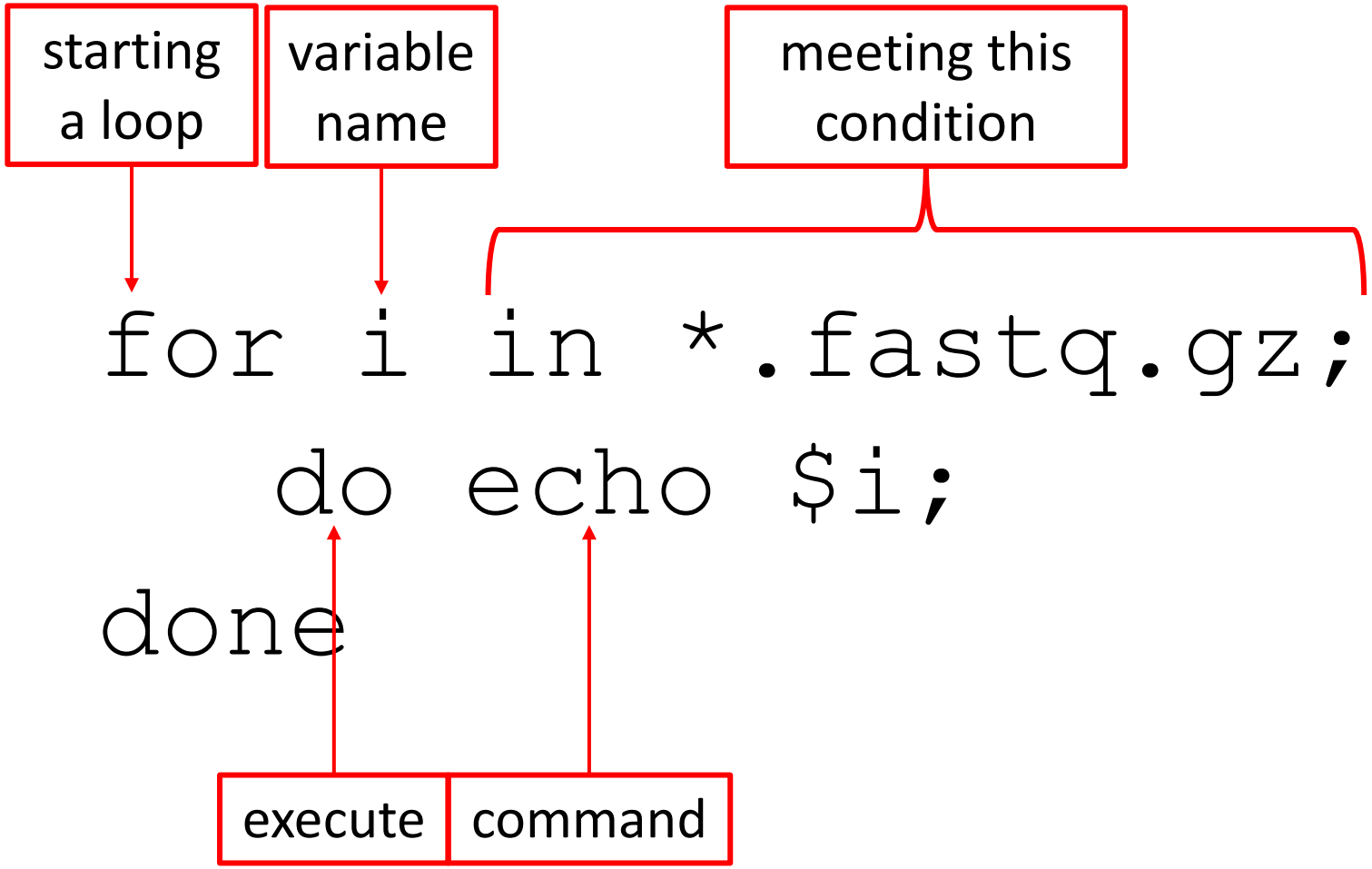
variable name

meeting this condition

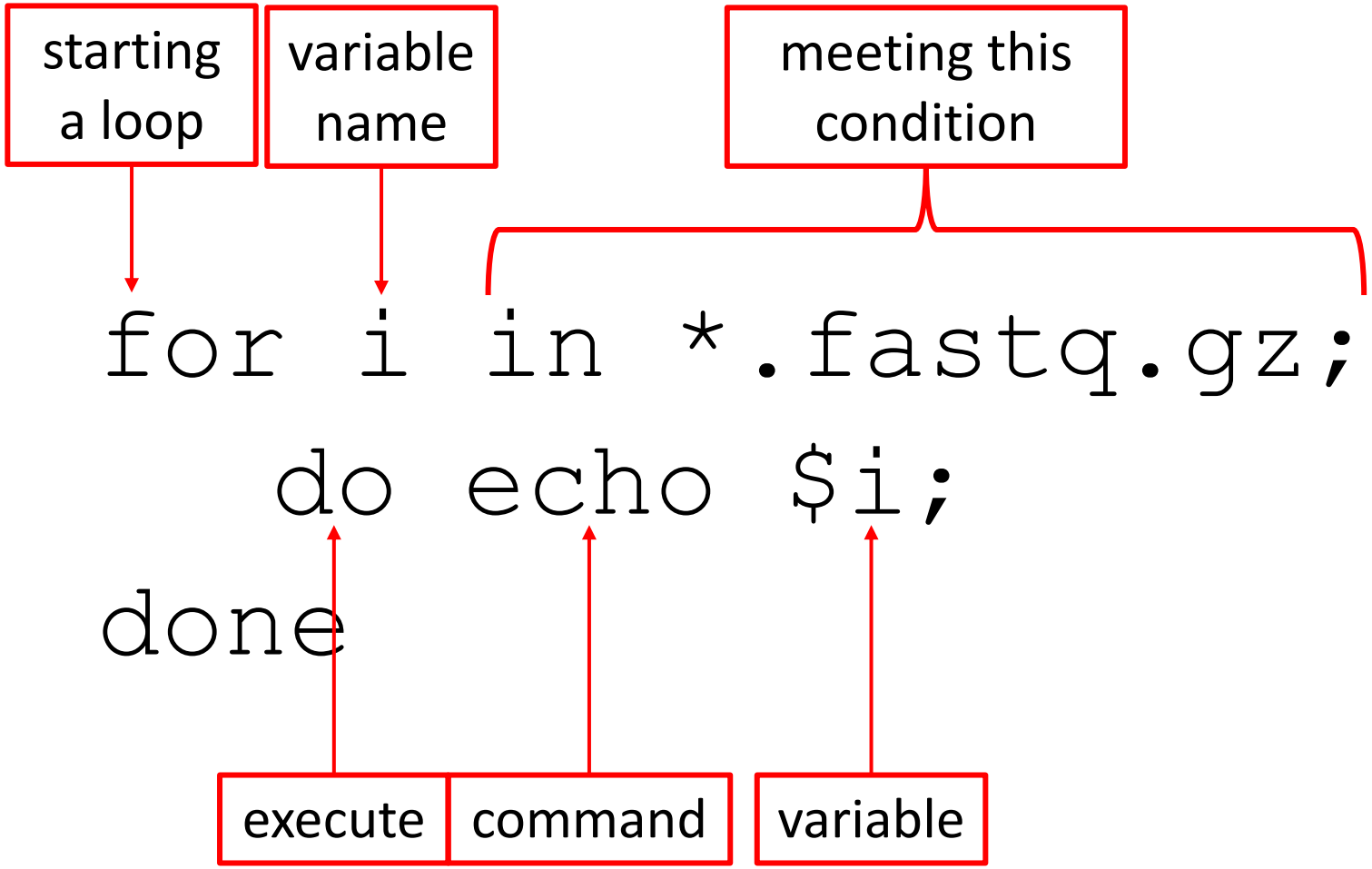
```
for i in *.fastq.gz;
   do echo $i;
done
```

execute

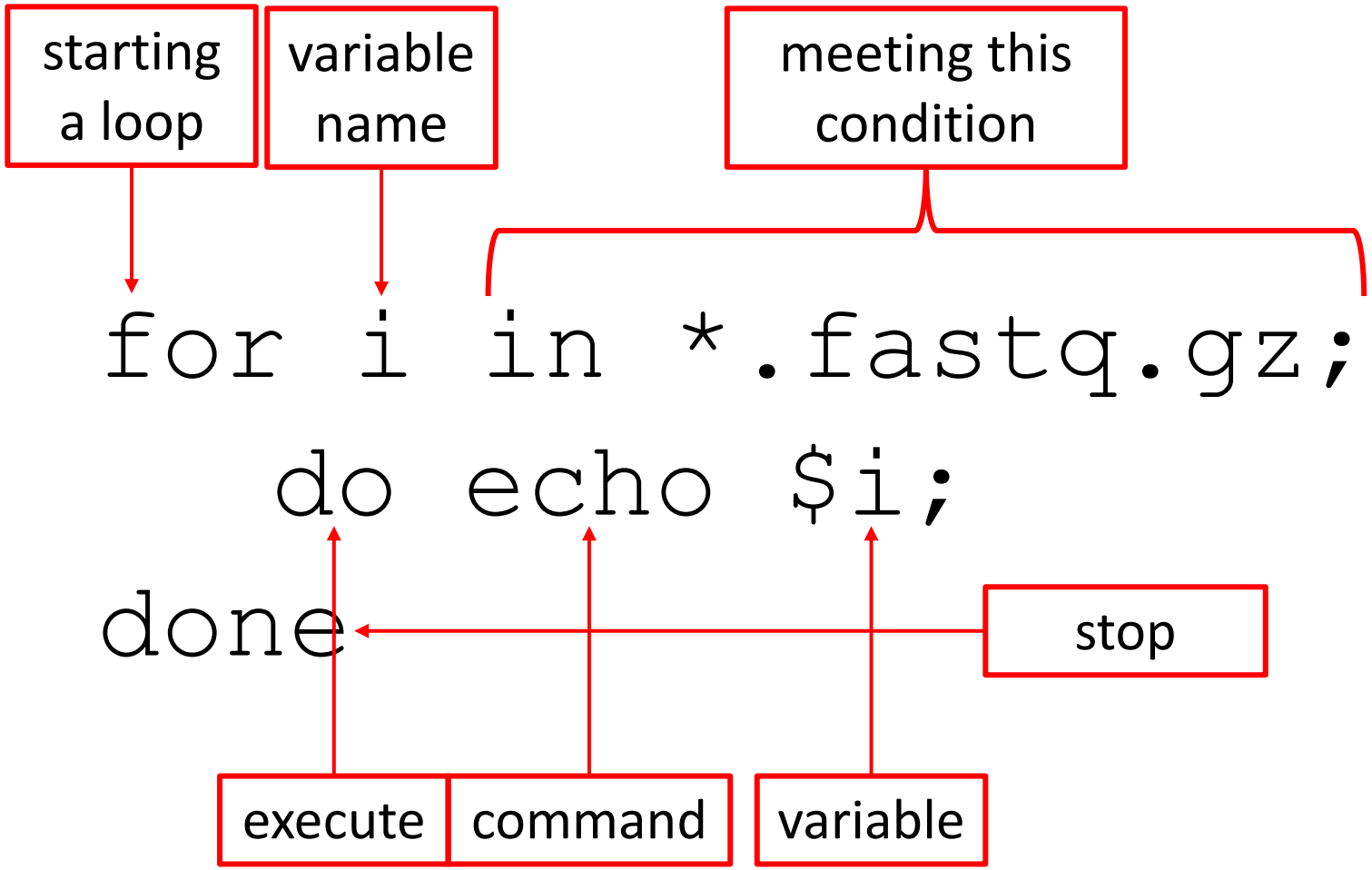
How do you write a loop in bash?



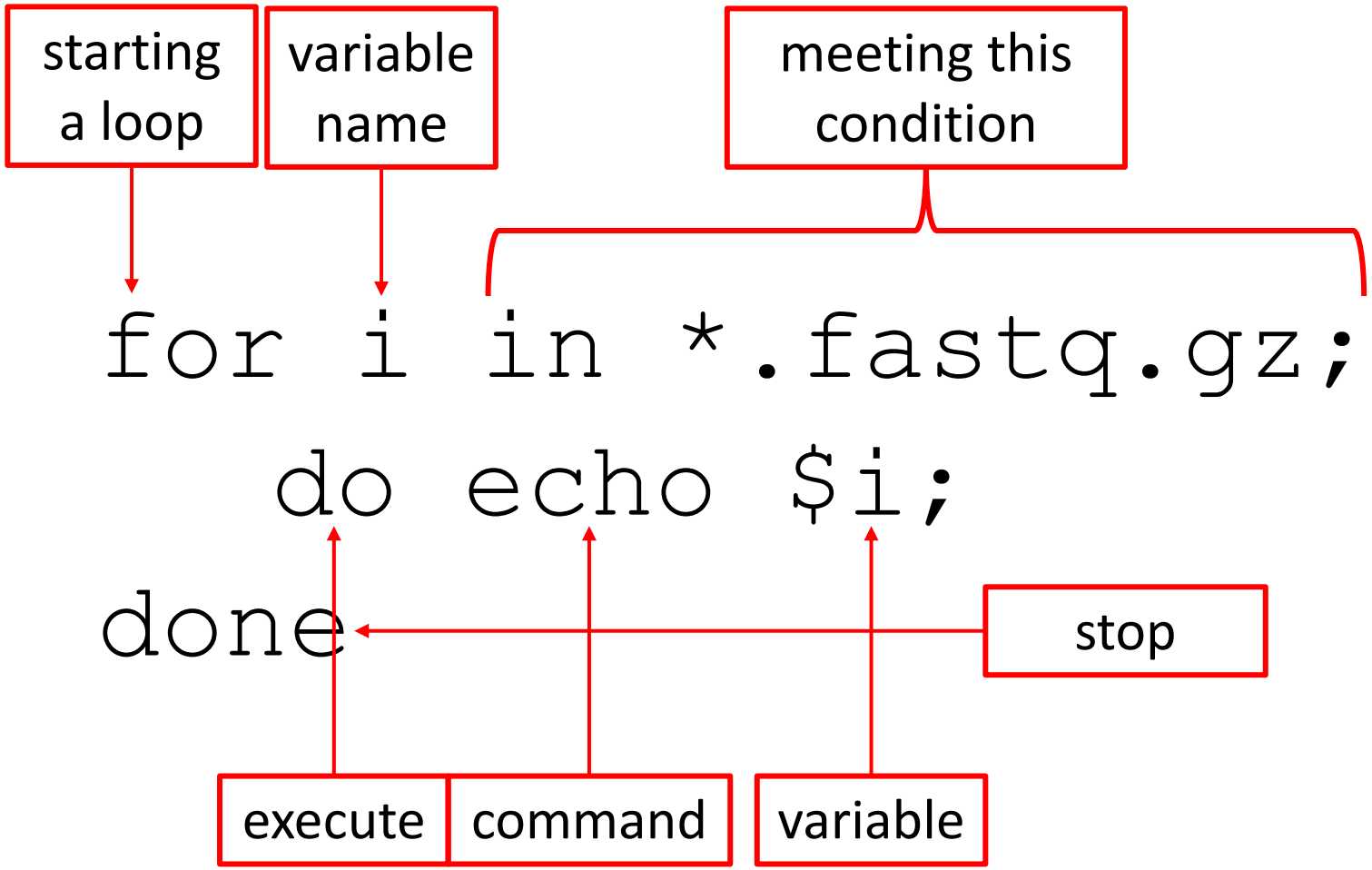
How do you write a loop in bash?



How do you write a loop in bash?



How do you write a loop in bash?



Now go try this loop on the server

Run FastQC

1. Go to the RNA-seq data directory
2. Make a directory to put the FastQC reports into, `fastqc`
3. Run fastqc on the samples

```
for i in *.fastq.gz; do fastqc $i -o fastqc/;
done
```

Trim Bad Quality Sequences

What is trimming and why do it?

What is trimming and why do it?

- Trimming removes sequencing adapters, bad quality sequences, and/or other biased sequence information

What is trimming and why do it?

- Trimming removes sequencing adapters, bad quality sequences, and/or other biased sequence information
- Why is that important?
  - Helps prevent incorrect base calls by removing poor quality information
  - Increases speed and accuracy of alignment by removing artificial sequences and low quality sequences

What is trimming and why do it?

- Trimming removes sequencing adapters, bad quality sequences, and/or other biased sequence information
- Why is that important?
  - Helps prevent incorrect base calls by removing poor quality information
  - Increases speed and accuracy of alignment by removing artificial sequences and low quality sequences

- Trimming does two complementary things:
  1. Removes any sequence information that comes from library preparation or sequencing
  2. Removes low quality bases / low quality reads

Trim Sequences

1. Go back up to the rnaseq directory
2. Make a folder to put the analysis results in, `analysis`
3. Make a folder inside the analysis folder to put the trimmed reads in, `analysis/01_trim`

Trim Sequences

```
for i in *R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output ../analysis/01_trim/
          --retain_unpaired
          -q 30
          $i
          ${i/R1/R2};
done
```

Trim Sequences

```
for i in *R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output ../analysis/01_trim/
          --retain_unpaired
          -q 30
          $i
          ${i/R1/R2};
done
```

loop condition

Trim Sequences

```
for i in *R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output ../analysis/01_trim/
          --retain_unpaired
          -q 30
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

Trim Sequences

```
for i in *R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output ../analysis/01_trim/
          --retain_unpaired
          -q 30
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

Trim Sequences

```
for i in *R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output ../analysis/01_trim/
          --retain_unpaired
          -q 30
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

run FastQC again after trimming

Trim Sequences

```
for i in *R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output ../analysis/01_trim/
          --retain_unpaired
          -q 30
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

run FastQC again after trimming

trim Illumina adapters

Trim Sequences

```
for i in *R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output ../analysis/01_trim/
          --retain_unpaired
          -q 30
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

run FastQC again after trimming

trim Illumina adapters

output goes here

Trim Sequences

```
for i in *R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output ../analysis/01_trim/
          --retain_unpaired
          -q 30
          $i
          ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

run FastQC again after trimming

trim Illumina adapters

output goes here

keep reads where one mate fails trimming but the other doesn't

Trim Sequences

```
for i in *R1.fastq.gz;
     do trim_galore
          --paired
          --fastqc
          --illumina
          --output ../analysis/01_trim/
          --retain_unpaired
          -q 30
          $i
          ${i/R1/R2};
done
```

- loop condition
- call the program
- reads are paired-end
- run FastQC again after trimming
- trim Illumina adapters
- output goes here
- keep reads where one mate fails trimming but the other doesn't
- Keep bases at this quality or above

Trim Sequences

```
for i in *R1.fastq.gz;
      do trim_galore
         --paired
         --fastqc
         --illumina
         --output ../analysis/01_trim/
         --retain_unpaired
         -q 30
         $i
         ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

run FastQC again after trimming

trim Illumina adapters

output goes here

keep reads where one mate fails trimming but the other doesn't

Keep bases at this quality or above

By default bases quality less than 20 will be trimmed and if the read falls below 20 bp, it will be discarded; we set the minimum quality to be 30

Trim Sequences

```
for i in *R1.fastq.gz;
      do trim_galore
         --paired
         --fastqc
         --illumina
         --output ../analysis/01_trim/
         --retain_unpaired
         -q 30
         $i
         ${i/R1/R2};
done
```

loop condition

call the program

reads are paired-end

run FastQC again after trimming

trim Illumina adapters

output goes here

keep reads where one mate fails trimming but the other doesn't

Keep bases at this quality or above

read files

By default bases quality less than 20 will be trimmed and if the read falls below 20 bp, it will be discarded; we set the minimum quality to be 30

Trim Command

```
for i in *R1.fastq.gz; do trim_galore
--paired --fastqc --illumina --output
../analysis/01_trim/
--retain_unpaired -q 30 $i ${i/R1/R2}; done
```