Homework#4

Calculate each of the following for sequences ≤ 100kb and < 100kb

`faFilter -minSize=100000 dmel-all-chromosome-r6.64.fasta/dev/stdout/ | faSize /dev/stdin`

`faFilter -maxSize=100000 dmel-all-chromosome-r6.64.fasta/dev/stdout/ |faSize /dev/stdin`

1. Total number of nucleotides

   Sequences ≤ 100kb?

   Answer: 6178042

   Sequences > 100kb?

   Answer: 137547960

2. Total nuber of Ns

   In sequences ≤ 100kb?

   Answer: 662993

   In sequences > 100kb?

   Answer: 490385

3. Total number of sequences

   Total sequences ≤ 100kb ?

   Answer: 7 sequences

   Total sequences > 100kb?

   Answer: 1863 sequences

Obtain sequence length distribution

Store sequence length data in .txt file

Less than or equal to 100kb

`bioawk -c fastx ‘length (seq) <= 100000 {print length($seq)} dmel-all-chromosome-r6.64.fasta > sequence_lengths100kb_and_below.txt`

Greater than 100kb

`bioawk -c fastx length ($seq) >=100000 {print length ($seq)} dmel-all-chromosome-r6.64.fasta > sequencelengths100kb_and_below.txt`

Transform sequence length data to log scale form and present in histogram

Write gnuplot script using Vim text editor

Less than or equal to 100kb

`vim Seqlength100kb_and_below.sh`

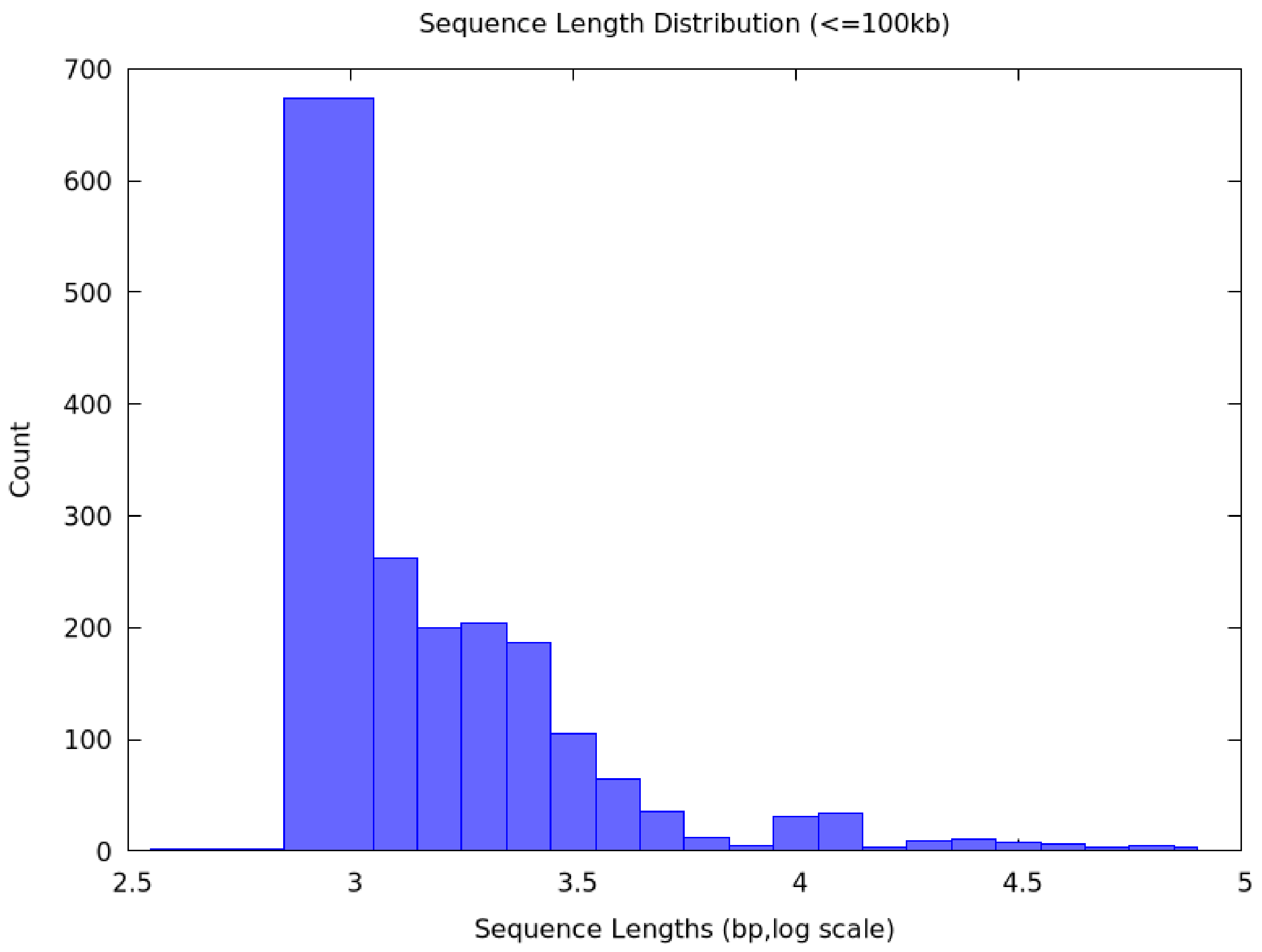
```
gnuplot -persist <<'EOF' title "Sequence Length Distribution (Log Bins)"
set xlabel "Sequence length (bp, log scale)"
set ylabel "Count"

set logscale x
binwidth = 0.1
bin(x) = 10**(binwidth*floor(log(x)/log(10)/binwidth))
set style fill solid 0.6
unset boxwidth
plot 'lengths.txt' using (bin($1)):(1) smooth freq with boxes lc rgb "blue"
notitle
EOF
```

Modify permissions to make writeable

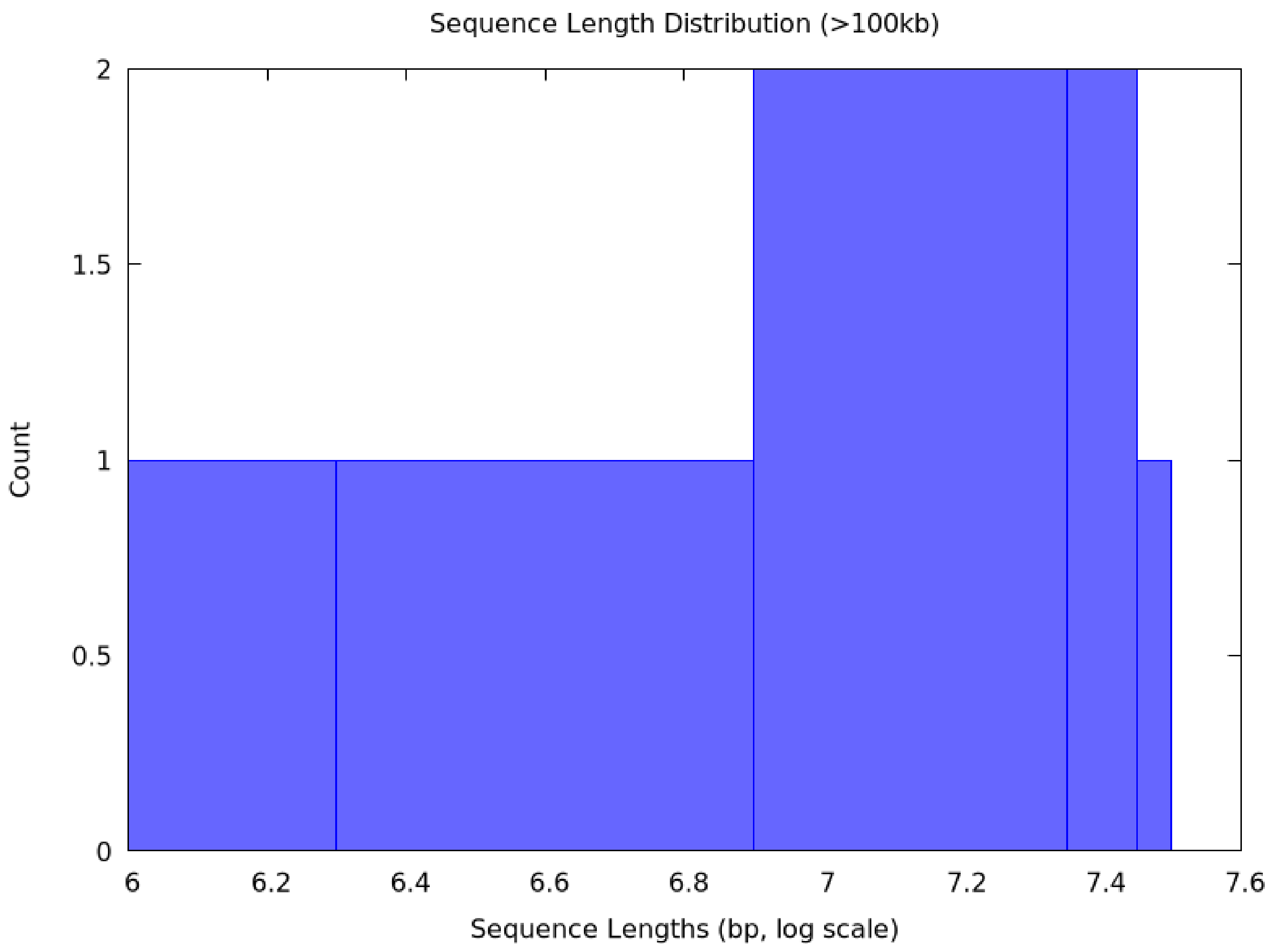
`chmod +x Seqlength100kb_and_below.sh`

Run command to obtain plot

`./Seqlength100kb_and_below.sh`



Repeat for sequences greater than 100kb



Obtain %GC of Sequences less than or equal to 100kb

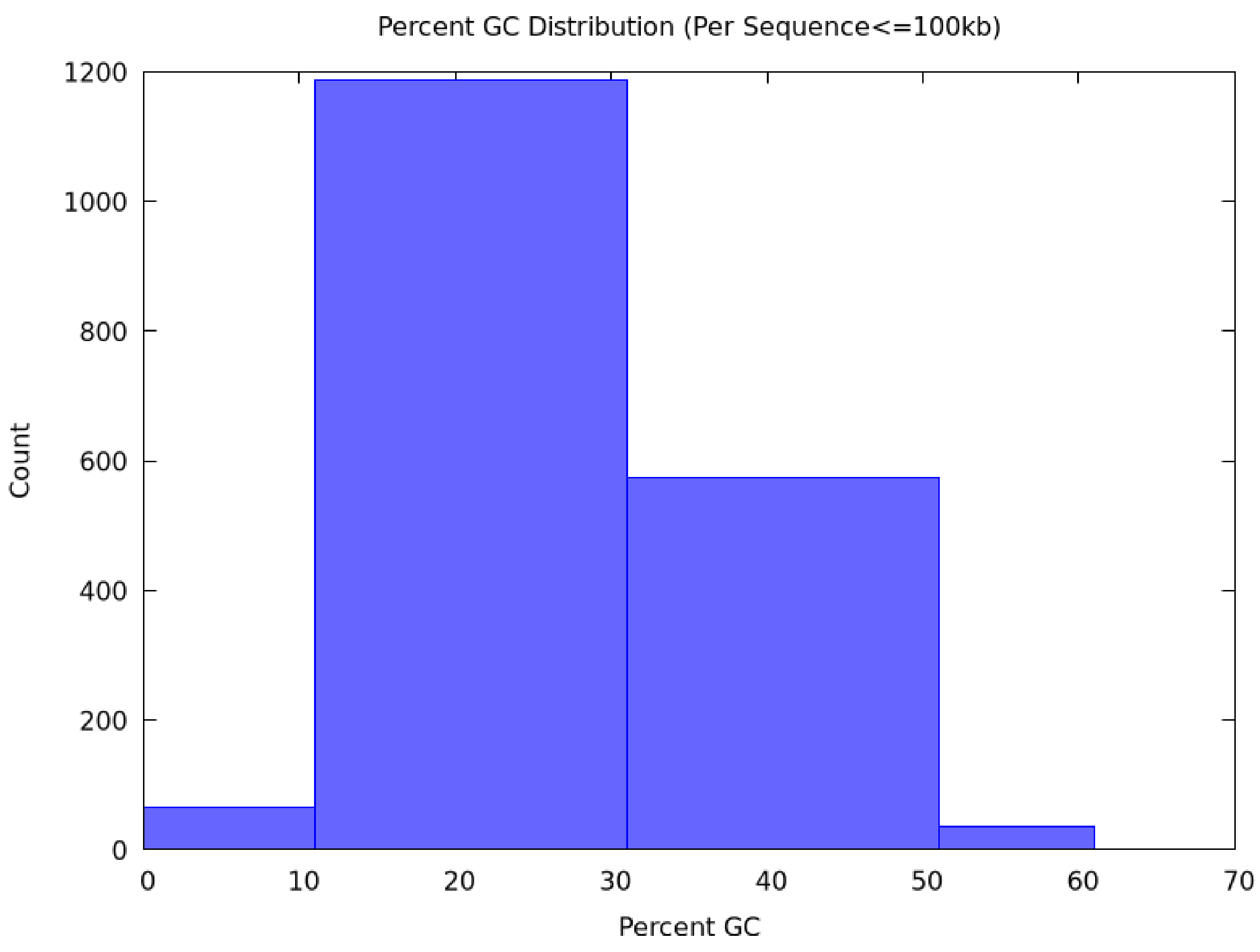
`bioawk -c fastx ‘length ($seq)≤100000 {print(gc($seq))*100}’ dmel-all-chromosome-r6.64.fasta > seqGC_100kb_and_below.txt`

Transform %GC content data from .txt file into histogram

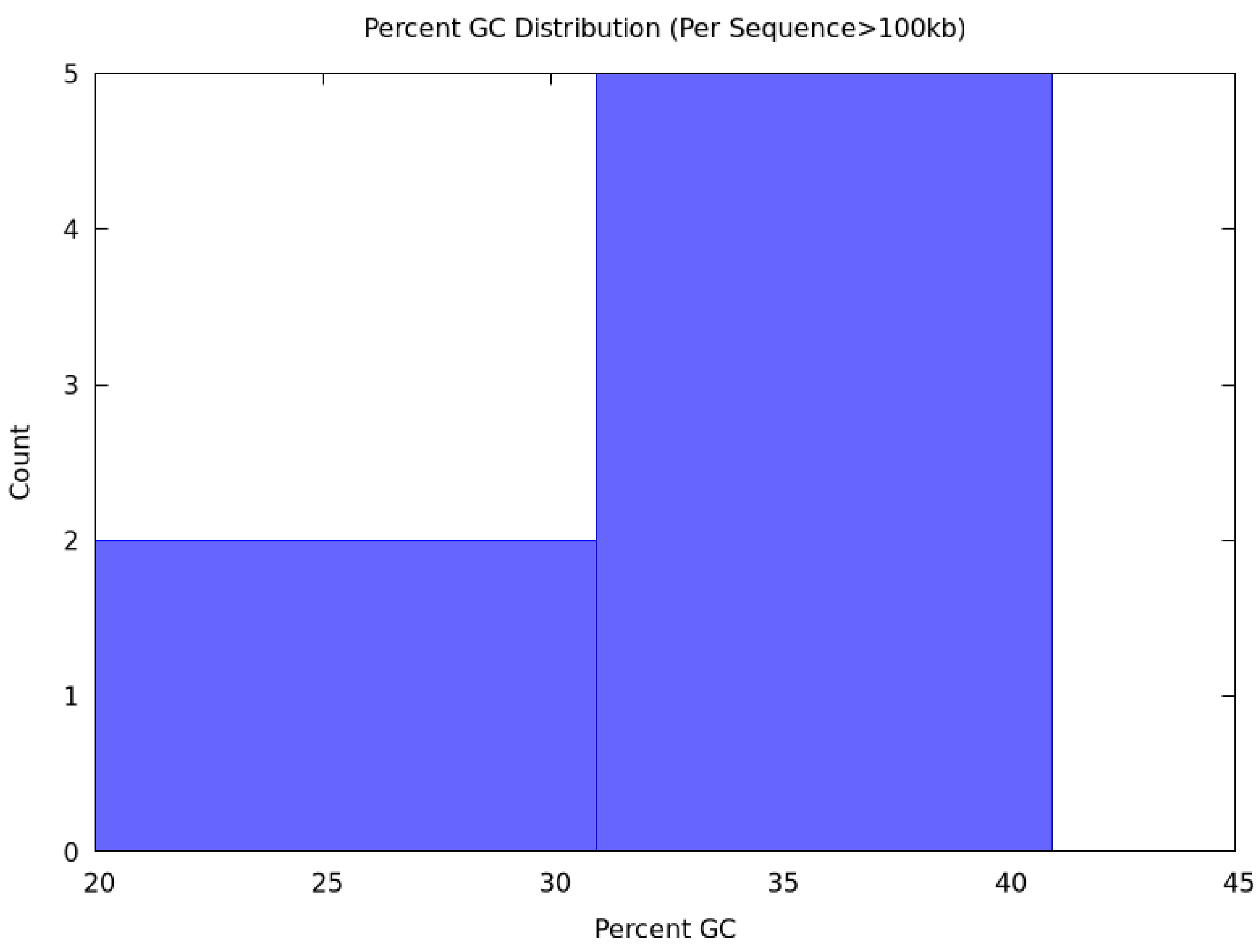
Initialize gnuplot script using Vim text editor

`vim GC_Percent100kb_and_below.sh`

```
#!/bin/bash
gnuplot -persist <<'EOF'
set title "Percent GC Distribution (Per Sequence<=100kb)"
set terminal pngcairo size 800, 600
set output "GC content 100kb or less.png"
set xlabel "Percent GC"
set ylabel "Count"
binwidth = 20
set style fill solid 0.6
bin(x) = binwidth *floor((x-1)/binwidth) +1
plot "seqGC_100kb_and_below.txt" using (bin($1)):(1) smooth freq with boxes lc rgb
"blue" notitle
EOF
```



Repeat for sequences greater than 100kb



Obtain cumulative size from largest to smallest for sequences ≤ 100kb and >100kb respectively, and store each in renamed.txt file

`awk '{ count[$1]++ ; total++;} END {cum = 0 ;for (len in count) {cum += count[len]; print len, cum/total;}}' cdf100kb_and_below.txt | sort -rn > sort_cdfdat100kb_less.txt`

Use data stored in .txt file to obtain CDF plot for each respective size partition

Initialize gnuplot script using Vim

`vim Cdf_plot100kb_and_below.sh`

```
#!/bin/bash
gnuplot -persist << 'EOF'
set terminal pngcairo size 800,600
set output "CDF100kb_or_Less.png"

set title "CDF Sequence Lengths <=100kb"
set xlabel "Sequence Length (bases)"
set ylabel "Cumulative Fraction"
set xrange [*:*] reverse
set grid
set style fill solid 0.8
set key top left
```

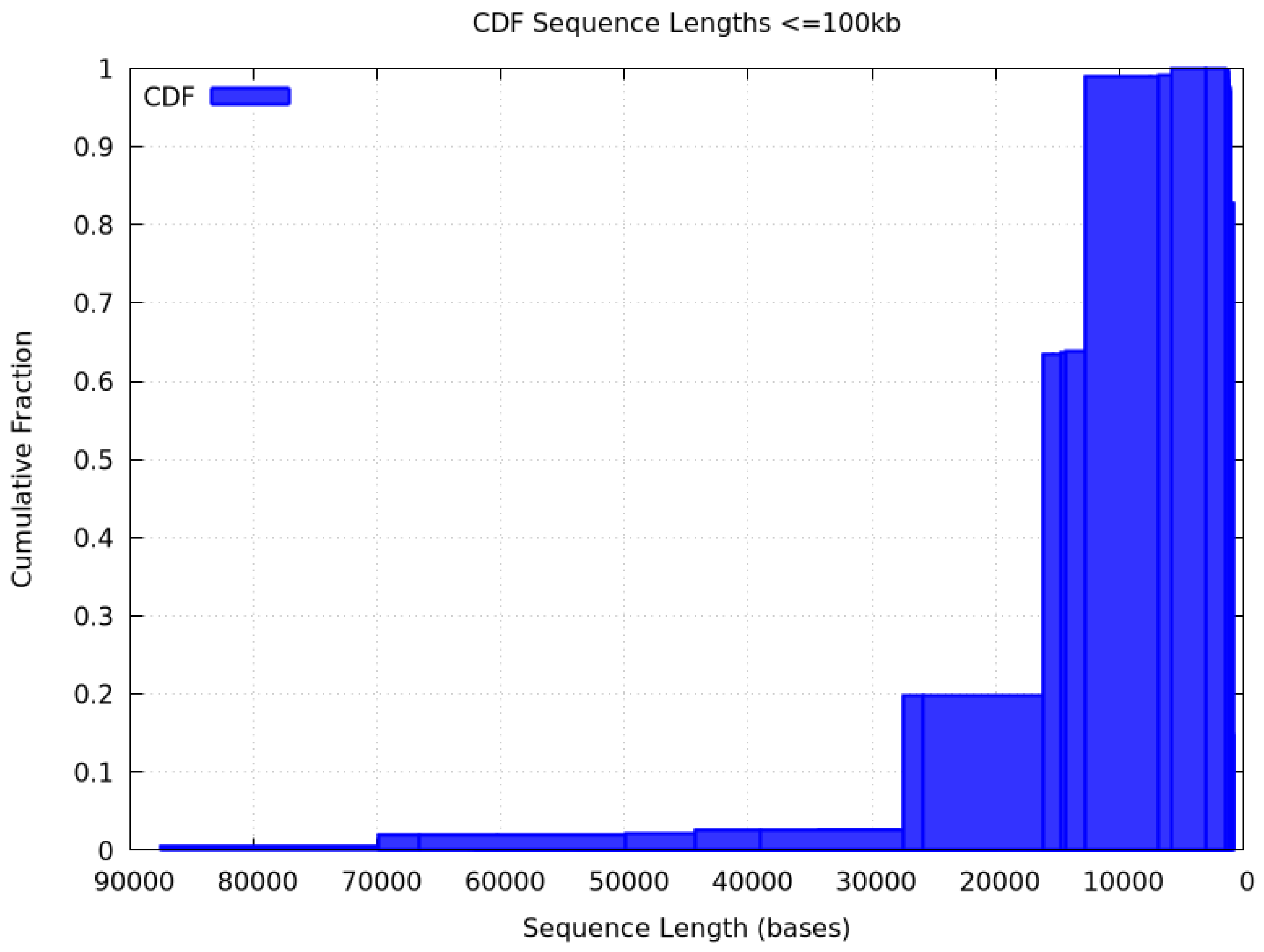
```
plot "cdf100kb_and_below.txt" using 1:2 with boxes lw 2 lc rgb "blue" title "CDF"
EOF
```

Modify permisions to make writeable

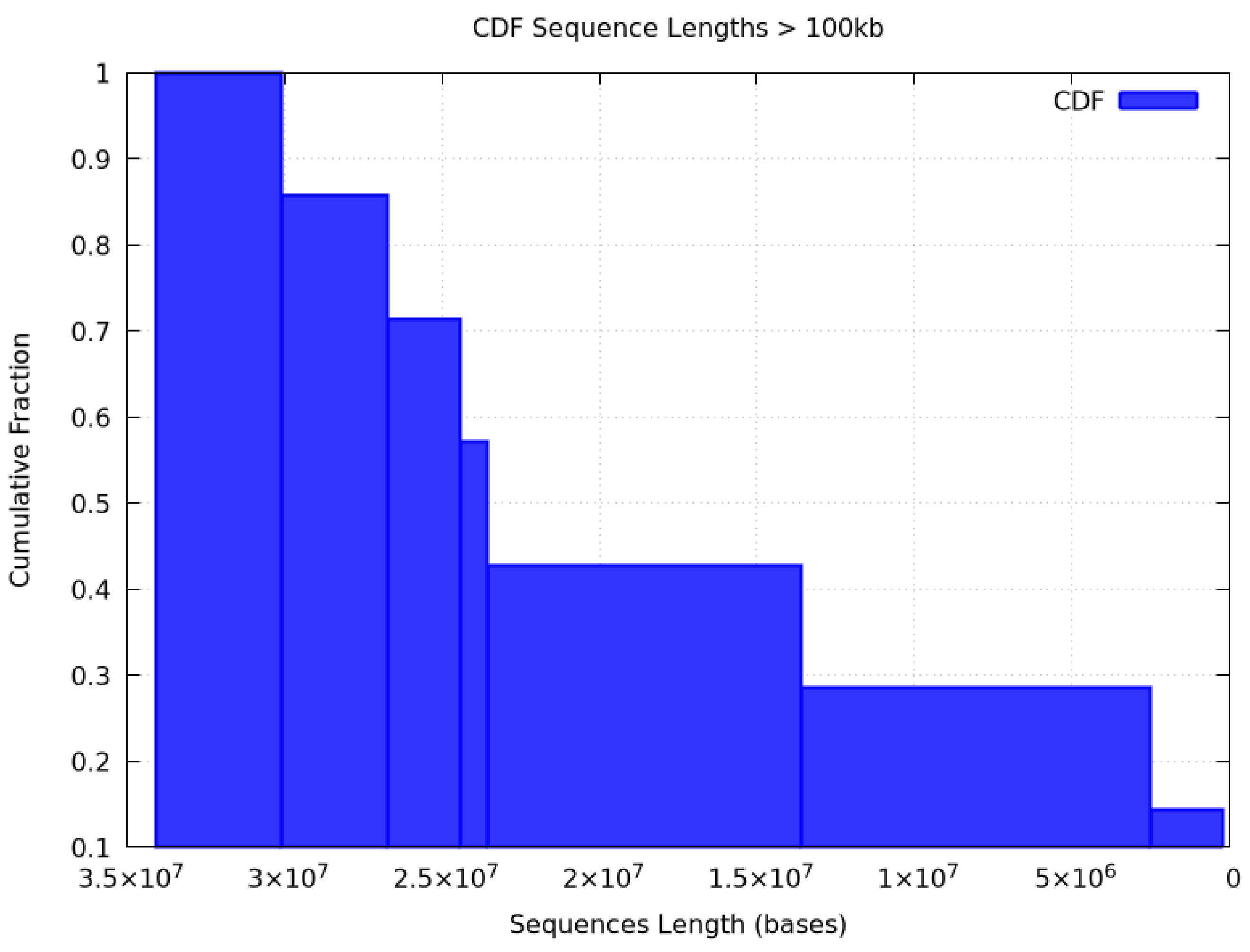
`chmod +x Cdf_plot100kb_and_below.sh`

Run command to obtain plot

`./Cdf_plot100kb_and_below.sh`



Repeat for sequences greater than 100kb



Sequence assembly

Obtain PacBio hifi D.melanogaster ISO1 genome sequence

```
wget m64069_221101_123519.hifi_reads.fastq.gz
```

Perform MD5sum

```
md5sum m64069-221101_123519.hifi_reads.fastq.gz
```

a7fe7514fb122bcdd3ca5204572c4be3 m64069_221101_123519.hifi_reads.fastq.gz

Rename for downstream applications (optional)

```
mv m64096 221101 123519.hifi_reads.fastq.gz ISO1_hifi.fastq.gz
```

Remove possible adapter sequences using hifiadatperfilt

hifiadapterfilt

install bioconda :: hifiadapterfilt

General formula for hifiadapterfilt commands

-p=prefix of target file -l=min length of sequences to target for removal -t=threads -o= output directory and resulting filename

Create FASTA database file for commonly used PacBio Adapter sequences

```
cat << 'EOF' > SMRTbell_adapters.fasta
>SMRTbell_Adapter_1
CTCACTCTCACACTACTCCAGTCGTCGCTCTATCTTCG
>SMRTbell_Adapter_2
CTCACTCTCACACTACTCCAGTCGTCGCTCTATCTTCGTAGACTTGGGGTAGCGGCCGCTCATAT
EOF
```

Store in blastdb file

`makeblastdb -in SMRTbell_adapters.fasta -dbtype nucl -out adapters`

Run hifiasm

```
bash /data/homezvol1/rsims1/miniforge3/envs/ee282/bin/hifiadapterfilt.sh \
  -p ISO1 \
  -l 25 \
  -m 98 \
  -t 16 \
  -o /tmp/rsims1/ISO1_filtered_L25
```

Output:

1. ISO1.filt.fastq.gz
2. ISO1.stats
3. ISO1.blocklist

Preliminary Stats

Reads Before filtering

`zcat ISO1.fastq.gz | grep -c "^@"` 1961704

Reads post filtering

`zcat ISO1.filt.fastq.gz | grep -c "^@"` 1961704

Bases before filtering

`zcat ISO1.fastq.gz | awk 'NR % 4 == 2 {bases += length($0)} END {print bases}'`

22505423345

Bases post filtering

`zcat ISO1.filt.fastq.gz | awk 'NR % 4 == 2 {sum += length($0)} END {print sum}'`

22505423345

Genome coverage

Purpose

Coverage in this context refers to the multiple iterative reads a genome undergoes after its initial sequencing.

It is a form of proofreading wherein greater coverage typically ensures greater accuracy and fidelity in scaffold or contigs that comprise whole genome assemblies.

The ISO1 genome consists of ~144mb. While our initial ISO1 file consists of 22 gigabases. This ~14x coverage indicates the initial ISO1 file I downloaded had undergone some proccesing, adapter addition, and downsampling.

ChatGPT provided a formula for further downsampling of this hifi read that would result in 10x coverage:

Run 10x coverage script

`vim ISO1_filt_10x.sh`

```
#!/bin/bash

INPUT="ISO1.filt.fastq.gz"
GENOME_SIZE=140000000  # ~140 Mb
TARGET_COVERAGE=10
BASES_TO_KEEP=$((GENOME_SIZE * TARGET_COVERAGE))
TOTAL_BASES=$(zcat $INPUT | awk 'NR % 4 == 2 {sum += length($0)} END {print sum}')
FRACTION=$(echo "$BASES_TO_KEEP / $TOTAL_BASES" | bc -l)
seqtk sample -s100 $INPUT $FRACTION > ISO1.filt.10x.fastq
EOF
```

`chmod +x ISO1_filt_10x.sh`

`./ISO1_filt_10x.sh`

Output:

ISO1.filt.10x.fastq

Perform Assembly on filtered 10x ISO1 Genome Using hifiasm

Install hifiasm

`mamba install hifiasm`

General Hifiasm format

hifiasm -o <output_prefix> -t <number_of_threads> <input_reads.fastq.gz>

Write hifiasm script in Vim `Vim BS_hifiasm_job.slurm`

```
#!/bin/bash
#SBATCH --job-name=ISO1_hifiasm
#SBATCH --output=ISO1_hifiasm.%j.out
#SBATCH --error=ISO1_hifiasm.%j.err
#SBATCH --partition=standard
#SBATCH --time=24:00:00
#SBATCH --cpus-per-task=32
#SBATCH --mem=120G
#SBATCH --ntasks=1
Module purge
module load hifiasm/0.21.0-r686

cp /data/homezvol1/rsims1/ISO1.filt.10x.fastq $TMPDIR/
cd $TMPDIR

hifiasm -o ISO1_assembly -t 32 ISO1.filt.10x.fastq > ISO1_assembly.log 2>&1

cp -r * /data/homezvol1/rsims1/ISO1_hifiasm_output/
```

Run hifiasm as sbatch job in slurm

`sbatch BS_hifiasm_job.slurm`

Check SLURM Job

$seff 43191002

Output:

`cd/data/homezvol1/rsims1/ISO1_hifiasm_output`

There will be multiple assembly output files, however ISO1_assembly.bp.p_ctg.gfa contains primary contig assembly and will be used for further downstream analysis

There are also haplotigs: ISO1_assembly.bp.hap1.p_ctg.gfa ISO1_assembly.bp.hap2.p_ctg.gfa

Convert from .gfa to .fasta file

`awk '/^S/{print ">"$2"\n"$3}' ISO1_assembly.bp.p_ctg.gfa > ISO1_assembly.bp.p_ctg.fa`

ISO1_assembly.bp.p_ctg.fa

`module load samtools`

Convert to fai (FASTA index file) using samtools

`samtools faidx ISO1_assembly.bp.p_ctg.fa`

ISO1_assembly.bp.p_ctg.fa.fai

Obtain assembly stats from.fai file

Count bases `awk '{sum += $2} END {print "Total bases:", sum}' ISO1_assembly.bp.p_ctg.fa.fai` Total bases: 128810372

Count contigs `awk 'END {print "Number of contigs:", NR}' ISO1_assembly.bp.p_ctg.fa.fai` Number of contigs: 411

Longest contig `awk 'max<$2 {max=$2} END {print "Longest contig:", max}' ISO1_assembly.bp.p_ctg.fa.fai`

Longest contig: 5237741

Calculate N50 of assembly

```
awk '{
    len[NR]=$2; total+=$2
} END {
    asort(len)
    half=total/2; sum=0
    for (i=NR; i>0; i--) {
        sum+=len[i]
        if (sum>=half) {
            print "N50:", len[i]
            break
        }
    }
}' ISO1_assembly.bp.p_ctg.fa.fai
```

N50: 1248665

Contiguity Plot

Compare fidelity of assemblies

1. r6_ISO1_MT Scaffold
2. r6_ISO1_MT Contigs
3. ISO1_assembly Contigs

```

```

#!/bin/bash set -euo pipefail

mkdir -p tmp output/figures

cut -f2 r6_ISO1_MT.fna.bgz.fai
| sort -rn
| awk 'BEGIN { print "Assembly\tLength\nFB_Scaff\t0" } { print "FB_Scaff\t" $1 }' \

> tmp/r6scaff.txt

zcat r6_ISO1_MT.fna.bgz
| faSplitByN /dev/stdin /dev/stdout 10
| bioawk -c fastx '{ print length($seq) }'
| sort -rn
| awk 'BEGIN { print "Assembly\tLength\nFB_Ctg\t0" } { print "FB_Ctg\t" $1 }' \

> tmp/r6ctg.txt

cut -f2 ISO1_assembly.bp.p_ctg.fa.fai
| sort -rn
| awk 'BEGIN { print "Assembly\tLength\nBS_ISO1_Ctg\t0" } { print "BS_ISO1_Ctg\t" $1 }' \

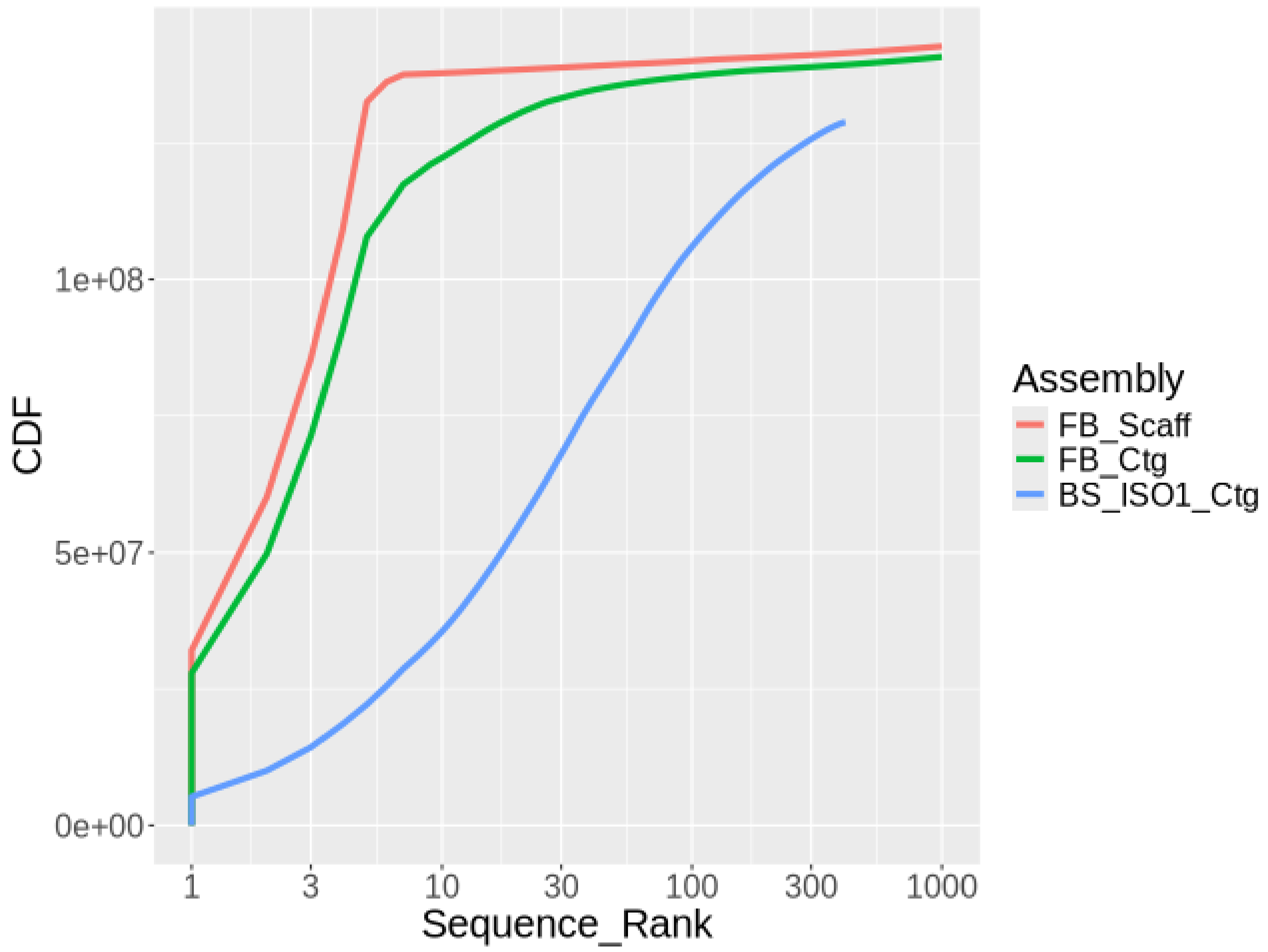
> tmp/BS_ISO1ctg.txt

Obtain CDF Contiguity plot

./ISO1_hifiasm_output/output/figures/CDF.png

Rename (optional)

`mv CDF.png BS_ISO1_vs_r6_ISO1_MT.png`



BUSCO

Check completeness/fidelity of assembly against established consensus assembly diptera odb10

`conda install bioconda::busco` ISO1_assembly vs diptera_odb10

`busco -i ISO1_assembly.bp.p_ctg.fa -l diptera_odb10 -o busco_out -m genome`

Locate Output

`cd /data/homezvol1/rsims1/ISO1_hifiasm_output/busco_out`

Output list:

$ls logs run_diptera_odb10
short_summary.specific.diptera_odb10.busco_out.json short_summary.specific.diptera_odb10.busco_out.txt

Access "run diptera_odb10"

`$cd /data/homezvol1/rsims1/ISO1_hifiasm_output/busco_out/run_diptera_odb10`

List

short_summary.txt

full_table.tsv

Obtain run statistics from short_summary.txt

`cat short_summary.txt`

***** Results: *****

C:89.6%[S:89.2%,D:0.4%],F:0.2%,M:10 3%,n:3285,E:11.0%

2942 Complete BUSCOs (C)
(of which 323 contain internal stop codons)

2930 Complete and single-copy BUSCOs (S)

12 Complete and duplicated BUSCOs (D)

5 Fragmented BUSCOs (F)

338 Missing BUSCOs (M)

3285 Total BUSCO groups searched

Assembly Statistics:

411 Number of scaffolds

411 Number of contigs

128810372 Total Base length

0.000% Percent gaps

1 Mbp Scaffold N50

1 Mbp Contigs N50

Move short_summary.txt and full_table.tsv to RStudio For plot generation

Repeat BUSCO with r6-ISO1_MT assembly

r6_ISO1_MT vs diptera_odb10

`busco -i ISO1_MT.fna.bgz -l diptera_odb10 -o busco_out_1 -m genome`

Designate unique output "Busco_out1"

Assign unique identifiers to corresponding short_summary.txt and full_table.tsv files

`mv short_summary.txt short_summary1.txt`

`mv full_table.tsv full_table1.tsv`

`cat short_summary1.tsv`

***** Results: *****

C:99.9%[S:99.7%,D:0.3%],F:0.0%,M:0.1%,n:3285,E:10.7%

3283 Complete BUSCOs (C) (of which 351 contain internal stop codons)

3274 Complete and single-copy BUSCOs (S)

9 Complete and duplicated BUSCOs (D)

0 Fragmented BUSCOs (F)

2 Missing BUSCOs (M)

3285 Total BUSCO groups searched

Assembly Statistics:

1870 Number of scaffolds

2442 Number of contigs

143726002 Total length

0.802% Percent gaps

25 Mbp Scaffold N50

21 Mbp Contigs N50

In R Studio

Begin new project for BUSCO plot

Load necessary BUSCO output statistics files

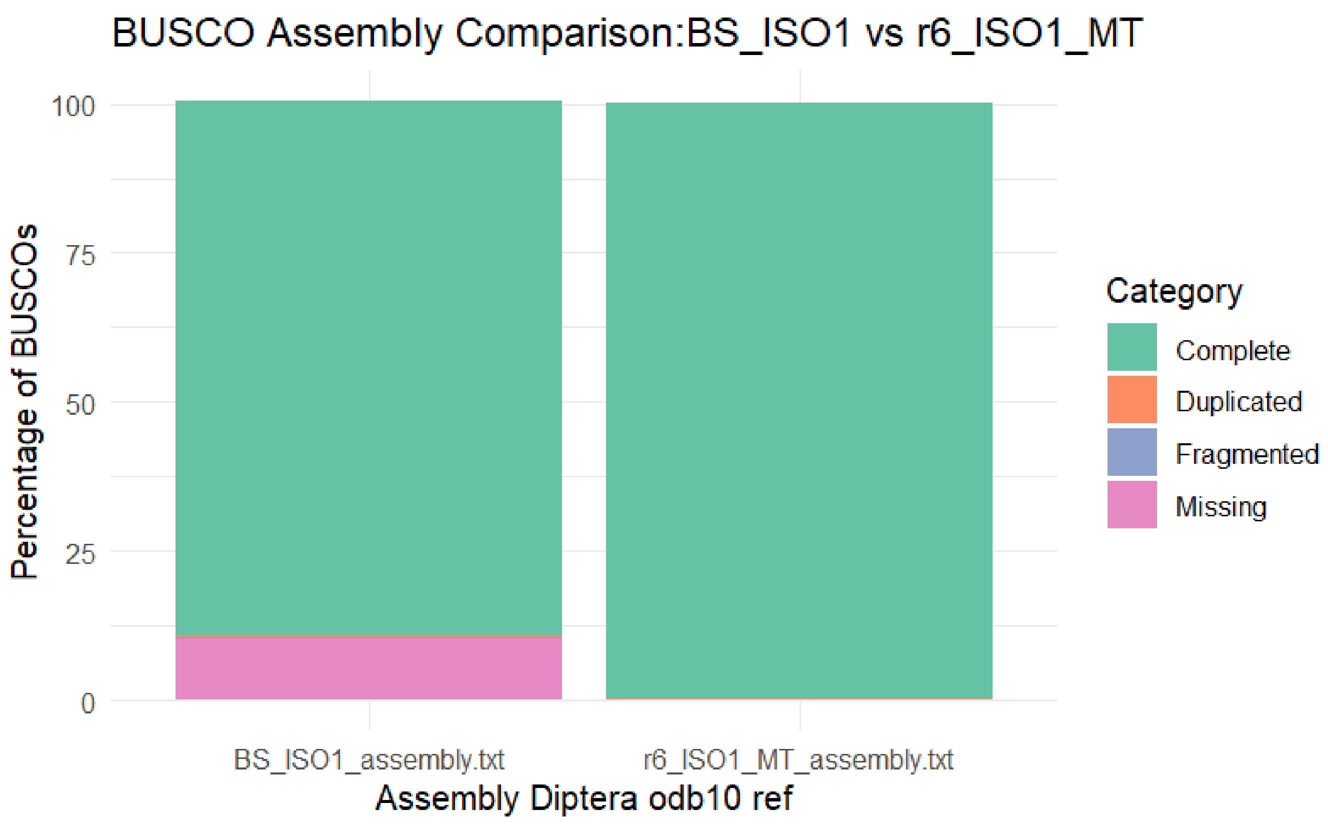
For ISO1_assembly vs diptera_odb_10: short_summary.txt, full_table.tsv

For r6_ISO1_MT vs diptera_odb_10:

short_summary1.txt, full_table1.tsv

Load necessary libraries

```
library(dplyr)


library(ggplot2)

library(tidyr)

### Function to parse BUSCO short_summary file

parse_busco_summary <- function(file) {
  lines <- readLines(file)




### Find the summary line (starts with "C:")

  summary_line <- lines[grepl("^\\s*C:", lines)]

  if (length(summary_line) == 0) stop("No summary line found in ", file)

### Extract percentages using regex


  complete <- as.numeric(str_match(summary_line, "C:([0-9.]+)%")[,2])

  single   <- as.numeric(str_match(summary_line, "S:([0-9.]+)%")[,2])

  duplicated <- as.numeric(str_match
  (summary_line, "D:([0-9.]+)%")[,2])

  fragmented <- as.numeric(str_match(summary_line, "F:([0-9.]+)%")[,2])

  missing <- as.numeric(str_match(summary_line, "M:([0-9.]+)%")[,2])

  tibble(
    Assembly = basename(file),
    Category = c("Complete", "Duplicated", "Fragmented", "Missing"),
    Percent = c(complete, duplicated, fragmented, missing)
  )
}

### Apply to the two summaries
### Replace with actual filenames

file.rename(from = "short_summary.txt", to = "BS_ISO1_assembly.txt")

file.rename(from = "short_summary1.txt", to = "r6_ISO1_MT_assembly.txt")

df1 <- parse_busco_summary("BS_ISO1_assembly.txt")

df2 <- parse_busco_summar("r6_ISO1_MT_assembly.txt")

busco_df <- bind_rows(df1, df2)

### Plot stacked bar chart

ggplot(busco_df, aes(x = Assembly, y = Percent, fill = Category)) +

  geom_bar(stat = "identity", position = "stack") +

  labs(
    title = "BUSCO Assembly Comparison:BS_ISO1 vs r6_ISO1_MT",

    x = "Assembly: Diptera odb10 ref",
    y = "Percentage of BUSCOs"
  )
   +
  scale_fill_brewer(palette = "Set2") +
  theme_minimal()
```


BUSCO Assembly Comparison:BS_ISO1 vs r6_ISO1_MT
Percentage of BUSCOs
100
75
50
25
0
BS_ISO1_assembly.txt
r6_ISO1_MT_assembly.txt
Assembly Diptera odb10 ref
Category
Complete
Duplicated
Fragmented
Missing