BINF 6110 – Project 2

Introduction

The emergence of high-throughput sequencing led to the production of large amounts of DNA and RNA reads from many different organisms (Reinert et al., 2015). As a result, researchers are faced with the task of identifying the genomic regions to which these reads belong (Reinert et al., 2015). The most common method used to accomplish this task is to align reads to a reference genome (Reinert et al., 2015). This method allows researchers to determine sequence variations in a species, which subsequently allows them to identify the mutations that confer both advantageous and disadvantageous traits in a population's gene pool (Reinert et al., 2015). Moreover, this method also enables researchers to deduce plausible phylogenetic relationships between different species (Valiente-Mullor et al., 2021).

Unfortunately, aligning reads to a reference genome presents several challenges. Firstly, it is computationally demanding and requires the use of very efficient algorithms (Reinert et al., 2015). The reason for this is that researchers must often map billions of reads to a very large reference genome (Trapnell & Salzberg, 2009). Furthermore, some genomes contain repeat regions whose lengths are much longer than the read lengths; generally, it is more difficult to align reads to a reference genome containing long and complex repeat patterns (Phan et al., 2015).

Secondly, researchers need to select an appropriate reference genome to which the reads will be aligned. Specifically, they often must decide between using a relatively complete genome from a distantly related species or a highly fragmented genome from the same species. On one hand, aligning reads to a genome from a distantly related species is preferred over conducting *de novo* assemblies, as it takes significantly less time and resources to complete (Prasad et al., 2022). As such, this method is preferred when there are no reference genomes for the species of interest. Furthermore, this method also allows researchers to infer several evolutionary traits from the species of interest, including levels of genetic diversity as well as adaptive genomic changes (Prasad et al., 2022). However, mapping efficiency tends to decrease as the phylogenetic distance between the reference genome and the reads increases (Prasad et al., 2022). On the other hand, using an incomplete reference genome from the same species often magnifies sequence differences between the reference genome and the reads (Valiente-Mullor et al., 2021). This is

problematic, as even simple sequence differences – such as single nucleotide polymorphisms – can notably affect downstream analyses (Valiente-Mullor et al., 2021).

This project aims to further investigate how the choice of reference genome as well as alignment software impacts alignment success. I will align 10 burbot reads to reference genomes belonging to burbot and cod, which is a related species. The burbot reference genome is highly fragmented, while the cod genome is relatively complete. Furthermore, I will use both the "bwa" and "bowtie2" alignment software to align the 10 burbot reads to both reference genomes. By comparing the percentage of reads mapped to each reference genome using the two software, I aim to find an appropriate alignment method that yields the highest alignment success rate.

Methods

List of Directory Paths

Below is a list of the Unix directories in which I did my analyses as well as the absolute paths to those directories. All the directories are viewable by the instructor and TA.

The directory "binf_6110" contains my .sh scripts and the directory "project_2".

```
cd /scratch/ajauwena/binf_6110
```

The directory "project_2" contains the directories in which the alignment results will be stored.

```
cd /scratch/ajauwena/binf_6110/project_2
```

The directories "burbot_alignment_bwa", "cod_alignment_bwa", "burbot_alignment_bowtie2", and "cod_alignment_bowtie2" contains the results of aligning the 10 burbot .fq read files to the burbot and cod reference genomes using both the bwa and bowtie2 software.

```
cd /scratch/ajauwena/binf_6110/project_2/burbot_alignment_bwa
cd /scratch/ajauwena/binf_6110/project_2/cod_alignment_bwa
cd /scratch/ajauwena/binf_6110/project_2/burbot_alignment_bowtie2
cd /scratch/ajauwena/binf_6110/project_2/cod_alignment_bowtie2
```

The directory “alignment_statistics” contains the .txt files that store various statistics obtained from aligning the 10 burbot .fq read files to the burbot and cod reference genomes using both the bwa and bowtie2 software.

```
cd /scratch/ajauwena/binf_6110/project_2/alignment_statistics
```

Aligning the 10 Burbot FASTQ Reads

Load the bwa and bowtie2 modules.

```
module load bwa
module load bowtie2
```

Make the directory for Project 2 and change into that directory.

```
mkdir /scratch/ajauwena/binf_6110/project_2
cd /scratch/ajauwena/binf_6110/project_2
```

Make the directories in which the alignment results will be stored.

```
mkdir burbot_alignment_bwa cod_alignment_bwa burbot_alignment_bowtie2
cod_alignment_bowtie2
```

Copy all the necessary files into the directory for Project 2.

```
rsync -av /scratch/emandevi/genomic_methods_w23/Project2/
/scratch/ajauwena/binf_6110/project_2
```

Change into the directory containing the 10 burbot .fq read files and unzip all the files in that directory.

```
cd /scratch/ajauwena/binf_6110/project_2/burbot_raw_data
gunzip *
```

Change back into the directory for Project 2 before beginning the alignment process.

```
cd /scratch/ajauwena/binf_6110
```

Execute the script “aligner.sh”, which aligns the 10 burbot .fq read files to the burbot and cod reference genomes using the bwa and bowtie2 software. The code for this script is provided in “Appendix 1” in the appendix section below.

```
sbatch aligner.sh /scratch/ajauwena/binf_6110/project_2/burbot_raw_data
/scratch/ajauwena/binf_6110/project_2/burbot_reference_genome
/scratch/ajauwena/binf_6110/project_2/cod_reference_genome
/scratch/ajauwena/binf_6110/project_2/burbot_alignment_bwa
/scratch/ajauwena/binf_6110/project_2/cod_alignment_bwa
/scratch/ajauwena/binf_6110/project_2/burbot_alignment_bowtie2
/scratch/ajauwena/binf_6110/project_2/cod_alignment_bowtie2
```

Calculating Alignment Statistics

Load the "samtools" module.

```
module load samtools
```

Make the directory in which the alignment statistics will be stored.

```
mkdir /scratch/ajauwena/binf_6110/project_2/alignment_statistics
```

Execute the script "alignment_statistics_calculator.sh", which outputs several alignment statistics of interest into .txt files. These statistics include: 1) the number of reads mapped, 2) the raw total sequences, 3) the error rate, 4) the average length, 5) the average quality, and 6) the alignment rate. The code for this script is provided in "Appendix 2" in the appendix section below. For reference, this script refers to alignment rate as "alignment_completeness".

```
sbatch alignment_statistics_calculator.sh
/scratch/ajauwena/binf_6110/project_2/burbot_alignment_bwa
/scratch/ajauwena/binf_6110/project_2/cod_alignment_bwa
/scratch/ajauwena/binf_6110/project_2/burbot_alignment_bowtie2
/scratch/ajauwena/binf_6110/project_2/cod_alignment_bowtie2
```

I used Python to convert the statistics obtained above into .csv files, which are all stored in one directory. The codes for the Python scripts are provided from "Appendix 3" to "Appendix 6" in the appendix section below. For reference, the Python scripts refer to alignment rate as "alignment_completeness".

Results

Visualizing Alignment Success Using R

Now that all the data have been obtained and processed, I can finally compare the alignment success between the two reference genomes as well as the two alignment software. I used R for this section.

Load the appropriate packages.

```
library("ggplot2")
library("dplyr")
```

Set the working directory to the desired directory. Then, check if the working directory is indeed the desired directory.

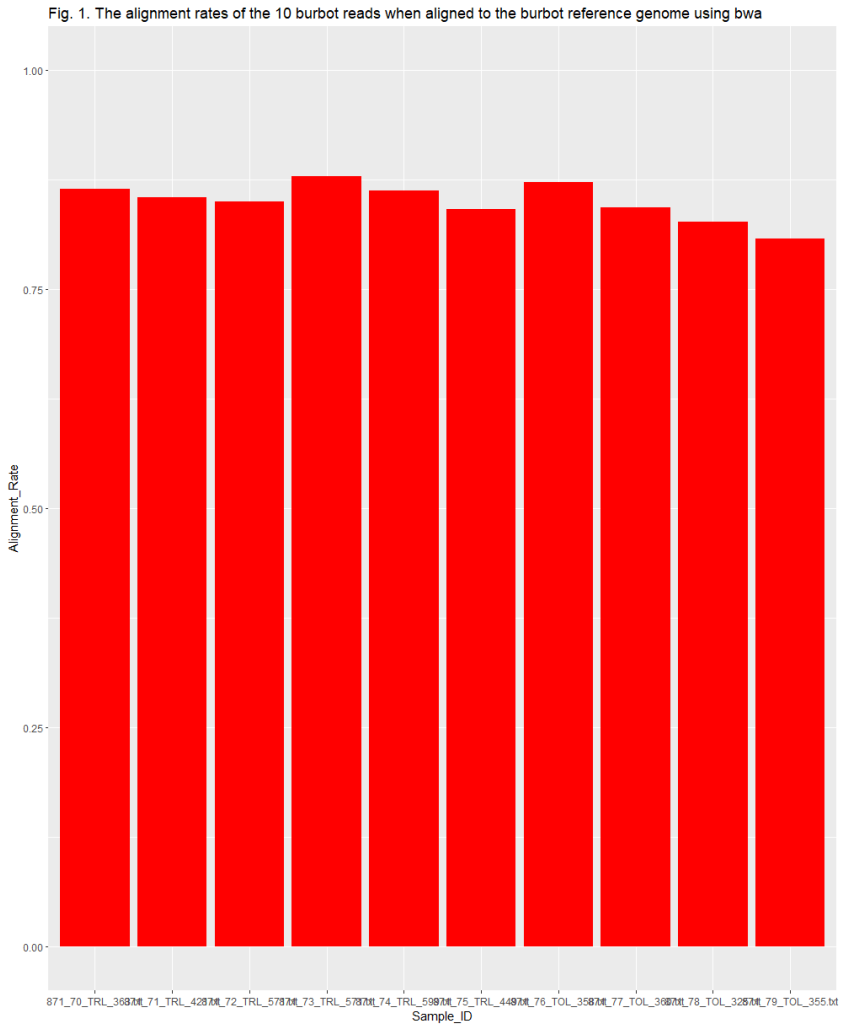
```
setwd("C:/Users/ajauw/OneDrive/Documents/m_a/w/education/master's/mbinf/w23/binf_6110/projects/p2/r_code")
getwd()
```

Read the .csv files from the directory as data frames.

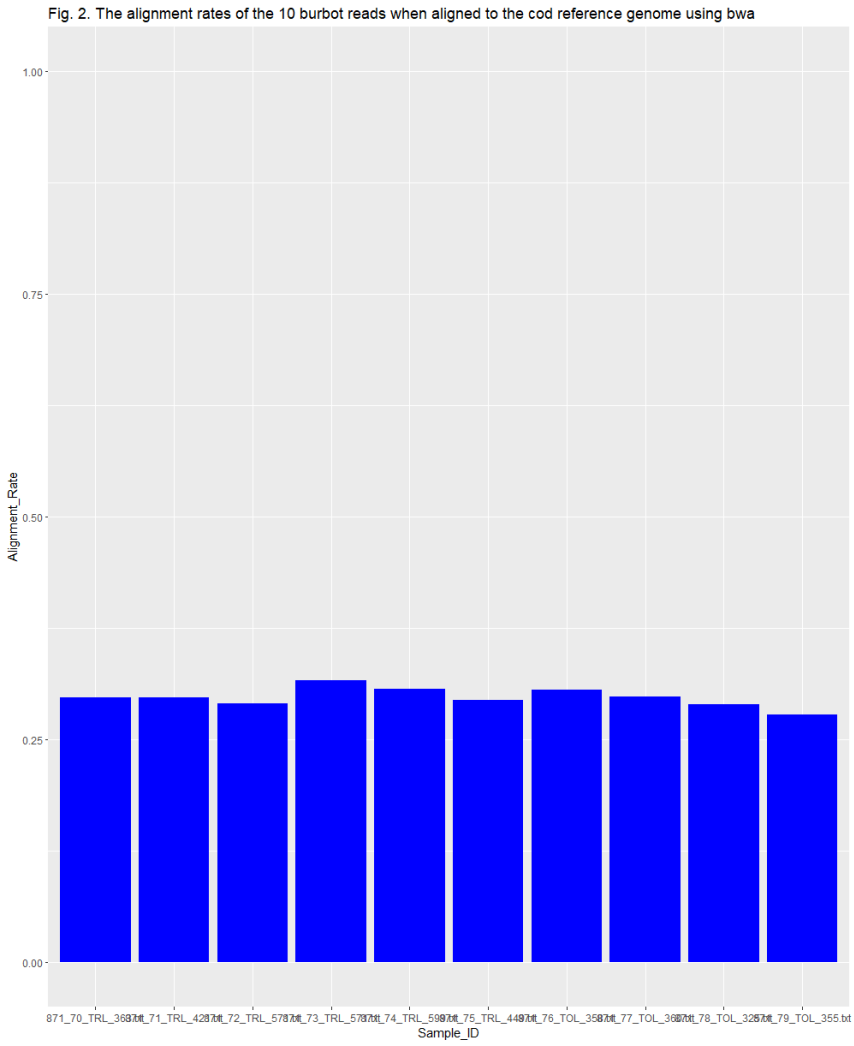
```
df_burbot_bwa <- read.csv('df_burbot_statistics_bwa.csv')
df_cod_bwa <- read.csv('df_cod_statistics_bwa.csv')
df_burbot_bowtie2 <- read.csv('df_burbot_statistics_bowtie2.csv')
df_cod_bowtie2 <- read.csv('df_cod_statistics_bowtie2.csv')
```

Plot bar graphs to visualize the alignment rates of the 10 burbot reads when aligned to the burbot and cod reference genomes using bwa and bowtie2.

```
# Using the burbot reference genome and bwa.
ggplot() +
      geom_bar(data = df_burbot_bwa, aes(x = File_Name, y =
      Alignment_Completeness), stat = "identity", fill = "red") +
      ylim(0, 1) +
      labs(x = "Sample_ID", y = "Alignment_Rate", title = "Fig. 1. The
      alignment rates of the 10 burbot reads when aligned to the burbot
      reference genome using bwa")
```

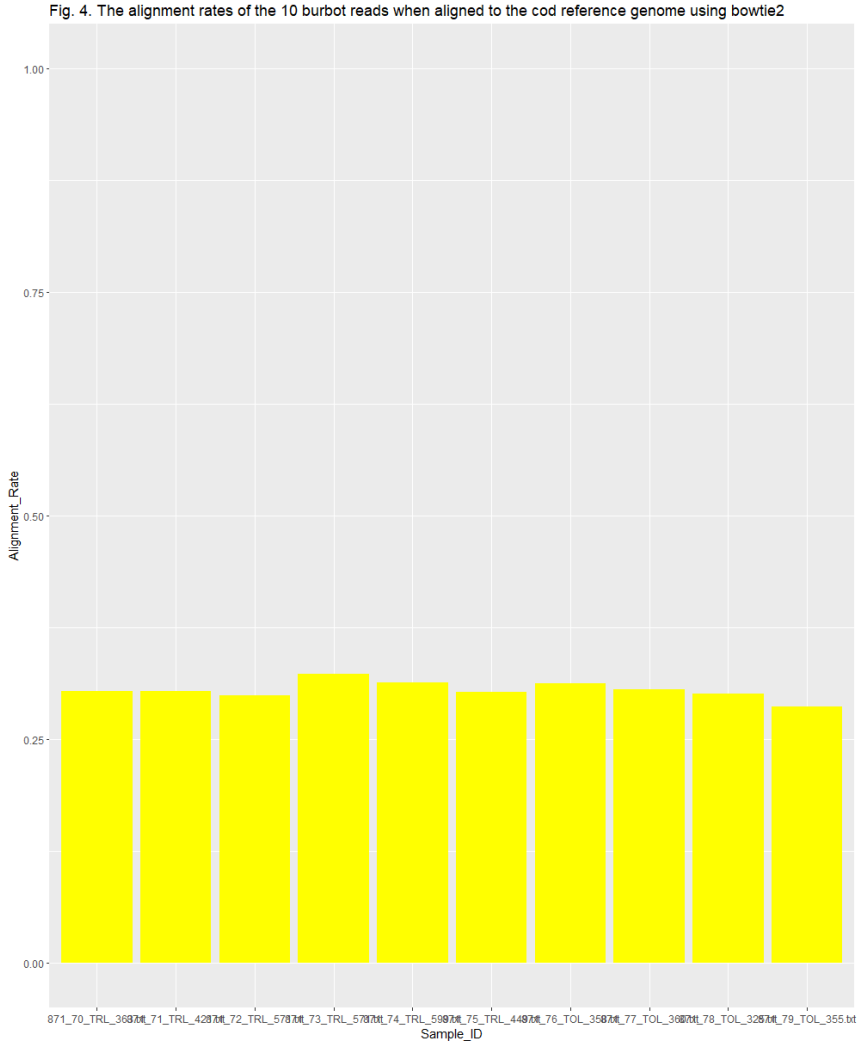


```
# Using the cod reference genome and bwa.
ggplot() +
      geom_bar(data = df_cod_bwa, aes(x = File_Name, y =
      Alignment_Completeness), stat = "identity", fill = "blue") +
      ylim(0, 1) +
      labs(x = "Sample_ID", y = "Alignment_Rate", title = "Fig. 2. The
      alignment rates of the 10 burbot reads when aligned to the cod
      reference genome using bwa")
```



```
# Using the burbot reference genome and bowtie2.
ggplot() +
      geom_bar(data = df_burbot_bowtie2, aes(x = File_Name, y =
      Alignment_Completeness), stat = "identity", fill = "green") +
      ylim(0, 1) +
      labs(x = "Sample_ID", y = "Alignment_Rate", title = "Fig. 3. The
      alignment rates of the 10 burbot reads when aligned to the burbot
      reference genome using bowtie2")
```

```
# Using the cod reference genome and bowtie2.
ggplot() +
      geom_bar(data = df_cod_bowtie2, aes(x = File_Name, y =
      Alignment_Completeness), stat = "identity", fill = "yellow") +
      ylim(0, 1) +
      labs(x = "Sample_ID", y = "Alignment_Rate", title = "Fig. 4. The
      alignment rates of the 10 burbot reads when aligned to the cod
      reference genome using bowtie2")
```



Calculate the average alignment rates of the 10 burbot reads when aligned to the burbot and cod reference genomes using bwa and bowtie2.

```
average_alignment_rate_burbot_bwa <-
mean(df_burbot_bwa$Alignment_Completeness)
average_alignment_rate_cod_bwa <- mean(df_cod_bwa$Alignment_Completeness)
average_alignment_rate_burbot_bowtie2 <-
mean(df_burbot_bowtie2$Alignment_Completeness)
average_alignment_rate_cod_bowtie2 <-
mean(df_cod_bowtie2$Alignment_Completeness)
```

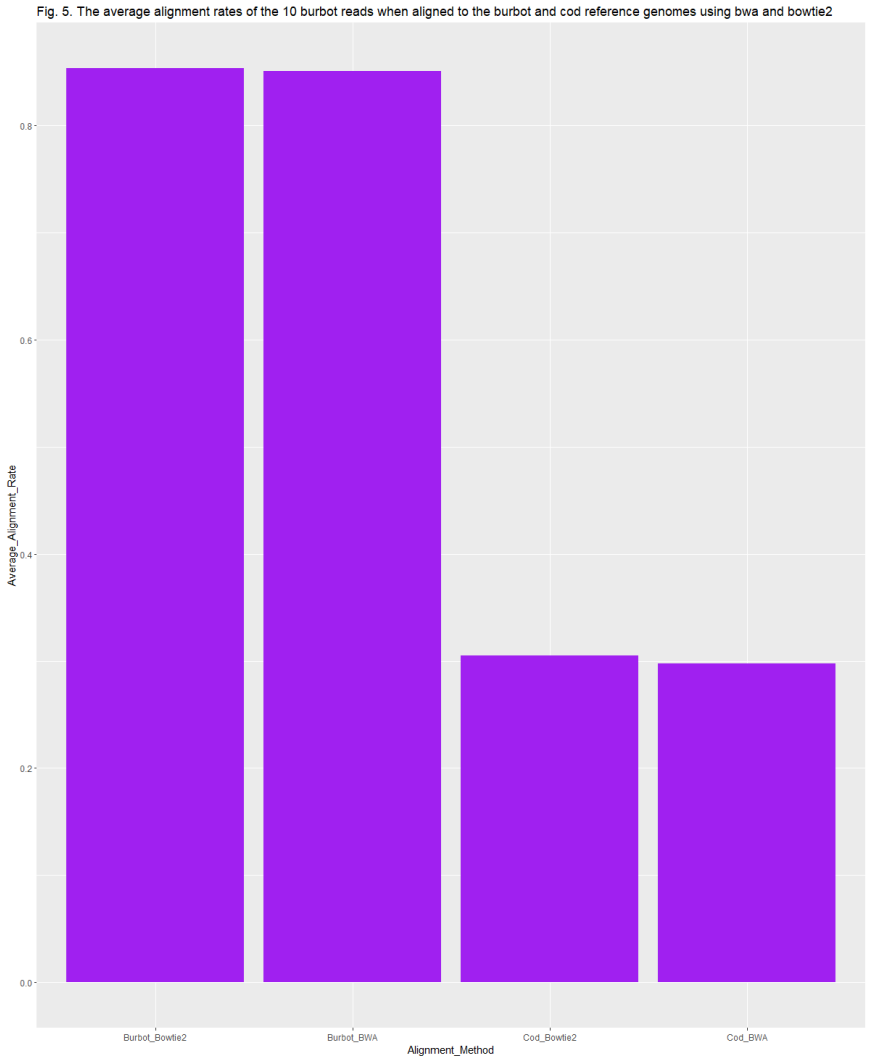
Create a new data frame containing the average alignment rates of the 10 burbot reads when aligned to the burbot and cod reference genomes using bwa and bowtie2.

```
# Create the columns of the data frame.
Alignment_Method_1 <- c("Burbot_BWA", "Cod_BWA", "Burbot_Bowtie2",
"Cod_Bowtie2")
Average_Alignment_Rate <- c(average_alignment_rate_burbot_bwa,
average_alignment_rate_cod_bwa, average_alignment_rate_burbot_bowtie2,
average_alignment_rate_cod_bowtie2)

# Create the data frame.
df_average_alignment_rates <- data.frame(Alignment_Method_1,
Average_Alignment_Rate)
```

Plot a bar graph to visualize the average alignment rates of the 10 burbot reads when aligned to the burbot and cod reference genomes using bwa and bowtie2.

```
ggplot() +
      geom_bar(data = df_average_alignment_rates, aes(x = Alignment_Method_1,
      y = Average_Alignment_Rate), stat = "identity", fill = "purple") +
      labs(x = "Alignment_Method", y = "Average_Alignment_Rate", title =
      "Fig. 5. The average alignment rates of the 10 burbot reads when
      aligned to the burbot and cod reference genomes using bwa and bowtie2")
```




```
# The mean alignment rates for the burbot reference genome are around 80% for
both bwa and bowtie. Conversely, the mean alignment rates for the cod
reference genome are around 30% for both bwa and bowtie.
```

Conduct a t-test to see whether there is a significant difference between the mean alignment success between the two reference genomes.

```
# Comparing the burbot vs. cod reference genome when using bwa.
t.test(df_burbot_bwa[, 8], df_cod_bwa[, 8])
```

```
> t.test(df_burbot_bwa[, 8], df_cod_bwa[, 8])

        Welch Two Sample t-test

data:  df_burbot_bwa[, 8] and df_cod_bwa[, 8]
t = 73, df = 13.216, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.5365645 0.5692355
sample estimates:
mean of x mean of y
   0.8505    0.2976
```

```
# Comparing the burbot vs. cod reference genome when using bowtie2.
t.test(df_burbot_bowtie2[, 8], df_cod_bowtie2[, 8])
```

```
> t.test(df_burbot_bowtie2[, 8], df_cod_bowtie2[, 8])

        Welch Two Sample t-test

data:  df_burbot_bowtie2[, 8] and df_cod_bowtie2[, 8]
t = 77.428, df = 12.975, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.5324153 0.5629847
sample estimates:
mean of x mean of y
   0.8531    0.3054
```

```
# There are significant differences in mean alignment success rates when
using the burbot versus the cod reference genome, as seen by the very small
p-values.
```

Conduct a t-test to see whether there is a significant difference between the mean alignment success between the two software.

```
# Comparing bwa vs. bowtie2 when using the burbot reference genome.
```

```
t.test(df_burbot_bwa[, 8], df_burbot_bowtie2[, 8])
```

```
> t.test(df_burbot_bwa[, 8], df_burbot_bowtie2[, 8])

        Welch Two Sample t-test

data:  df_burbot_bwa[, 8] and df_burbot_bowtie2[, 8]
t = -0.27951, df = 17.931, p-value = 0.7831
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.02214808  0.01694808
sample estimates:
mean of x mean of y
   0.8505    0.8531
```

```
# Comparing bwa vs. bowtie2 when using the cod reference genome.
t.test(df_cod_bwa[, 8], df_cod_bowtie2[, 8])
```

```
> t.test(df_cod_bwa[, 8], df_cod_bowtie2[, 8])

        Welch Two Sample t-test

data:  df_cod_bwa[, 8] and df_cod_bowtie2[, 8]
t = -1.7072, df = 17.84, p-value = 0.1051
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.017405245  0.001805245
sample estimates:
mean of x mean of y
   0.2976    0.3054
```

```
# There are no significant differences in mean alignment success rates when
using bwa versus bowtie2, as seen by the relatively large p-values.
```

Discussion

Regardless of the software of choice, aligning the 10 burbot reads to the burbot reference genome produced higher alignment success rates compared to aligning them to the cod reference genome (see *Fig. 5*). These results suggest that using a reference genome from the species of interest or from a very closely related species yield more accurate alignments, even if the reference genome is highly fragmented. The t-test results also support this conclusion, as they

show significant differences in mean alignment success rates when using the burbot versus the cod reference genome. Additionally, the differences in mean alignment success rates remain significant regardless of the choice of software. These t-test results further highlight the importance of using a reference genome from the species of interest during alignment analyses to ensure accurate alignment results. Further research, then, can investigate the degree to which the reference genome should be complete to still produce accurate alignment results against reads from the same species. Alternatively, future studies can also investigate whether using reference genomes from an even more closely related species will yield favorable alignment results.

Despite the poor alignment success rates, using a reference genome from a related species may be beneficial in certain cases. The first case is when investigating genomic regions that are conserved between species; in this case, the reads that align to the reference genome may give insight into which genes are conserved between species sharing a common ancestor. Subsequently, one can further investigate whether these genes grant an evolutionary advantage. The second is when the species of interest lacks a reference genome. As mentioned previously, aligning reads to a genome from a distantly related species takes significantly less time and resources compared to *de novo* assemblies (Prasad et al., 2022).

Interestingly, there are no significant differences in mean alignment success rates when using bwa versus bowtie2, suggesting that the choice of alignment software seems to not have a significant impact on alignment accuracy. Nevertheless, I am not confident in making this conclusion, seeing that this project only investigated two different software. As such, future studies should explore how the performance of other alignment software perform in aligning reads to reference genomes, especially when compared to bwa and bowtie2.

References

Phan, V., Gao, S., Tran, Q., & Vo, N. S. (2015). How genome complexity can explain the difficulty of aligning reads to genomes. *BMC Bioinformatics, 16*(S3). **https://doi.org/10.1186/1471-2105-16-S17-S3**

Prasad, A., Lorenzen, E. D., & Westbury, M V. (2022). Evaluating the role of reference-genome phylogenetic distance on evolutionary inference. *Molecular Ecology Resources, 22*, 45-55. **https://doi.org/10.1111/1755-0998.13457**

Reinert, K., Langmead, B., Weese, D., & Evers, D. J. (2015). Alignment of Next-Generation Sequencing Reads. *Annual Reviews of Genomics and Human Genetics, 16,* 133-151. **https://doi.org/10.1146/annurev-genom-090413-025358**

Trapnell, C., & Salzberg, S. L. (2009). How to map billions of short reads onto genomes. *Nature Biotechnology, 27*, 455-457. **https://doi.org/10.1038/nbt0509-455**

Valiente-Mullor, C., Beamud, B., Ansari, I., Francés-Cuesta, C., García-González, N., Mejía, L., Ruiz-Hueso, P., & González-Candelas, F. (2021). One is not enough: On the effects of reference genome for the mapping and subsequent analyses of short-reads. *PLOS Computational Biology, 17*(1). **https://doi.org/10.1371/journal.pcbi.1008678**

Appendix

Appendix 1: The Unix script "aligner.sh"

```
#!/bin/sh

#SBATCH --account=def-lukens
#SBATCH --time=0-10:00:00 ## days-hours:minutes:seconds
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4 # number of threads
#SBATCH --mem=16000 # requested memory (in MB)
#SBATCH --mail-type=END

# To execute this script, type the following command in the terminal:
      # sbatch aligner.sh
      /scratch/ajauwena/binf_6110/project_2/burbot_raw_data
      /scratch/ajauwena/binf_6110/project_2/burbot_reference_genome
      /scratch/ajauwena/binf_6110/project_2/cod_reference_genome
      /scratch/ajauwena/binf_6110/project_2/burbot_alignment_bwa
      /scratch/ajauwena/binf_6110/project_2/cod_alignment_bwa
      /scratch/ajauwena/binf_6110/project_2/burbot_alignment_bowtie2
      /scratch/ajauwena/binf_6110/project_2/cod_alignment_bowtie2

# Set a variable storing the working directory.
wd=/scratch/ajauwena/binf_6110/project_2

# Set variables storing the burbot and cod reference genomes (i.e., the .fna
files).
burbot_ref_gen=$2/GCA_900302385.1_ASM90030238v1_genomic.fna
cod_ref_gen=$3/GCF_902167405.1_gadMor3.0_genomic.fna

# Build the bwa index for the burbot reference genome.
bwa index -a bwtsw ${burbot_ref_gen}

# Build the bwa index for the cod reference genome.
bwa index -a bwtsw ${cod_ref_gen}

# Build the bowtie2 index for the burbot reference genome.
bowtie2-build ${burbot_ref_gen} $2/GCA_900302385.1_ASM90030238v1_genomic
```

```
# Build the bowtie2 index for the cod reference genome.
bowtie2-build ${cod_ref_gen} $3/GCF_902167405.1_gadMor3.0_genomic

# Loop through each raw data file.
for file in $1/*
do

	# Extract the file's base name without the .fq.
	base_name_fq=$(echo ${file} | rev | cut -d '/' -f1 | rev)
	base_name=$(echo ${base_name_fq} | cut -d '.' -f1)

	# Align the file to the burbot reference genome using bwa, then output
	the results to a .sam file in the appropriate directory.
	bwa mem -t 4 ${burbot_ref_gen} ${file} > $4/${base_name}.sam

	# Align the file to the cod reference genome using bwa, then output the
	results to a .sam file in the appropriate directory.
	bwa mem -t 4 ${cod_ref_gen} ${file} > $5/${base_name}.sam

	# Align the file to the burbot reference genome using bowtie2, then
	output the results to a .sam file in the appropriate directory.
	bowtie2 -x $2/GCA_900302385.1_ASM90030238v1_genomic -U ${file} -S
	$6/${base_name}.sam --very-sensitive-local

	# Align the file to the cod reference genome using bowtie2, then output
	the results to a .sam file in the appropriate directory.
	bowtie2 -x $3/GCF_902167405.1_gadMor3.0_genomic -U ${file} -S
	$7/${base_name}.sam --very-sensitive-local

done
```

Appendix 2: The Unix script "alignment_statistics_calculator.sh"

```
#!/bin/sh

#SBATCH --account=def-lukens
#SBATCH --time=0-10:00:00 ## days-hours:minutes:seconds
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4 # number of threads
#SBATCH --mem=16000 # requested memory (in MB)
#SBATCH --mail-type=END

# To execute this script, type the following command in the terminal:
      # sbatch alignment_statistics_calculator.sh
      /scratch/ajauwena/binf_6110/project_2/burbot_alignment_bwa
      /scratch/ajauwena/binf_6110/project_2/cod_alignment_bwa
      /scratch/ajauwena/binf_6110/project_2/burbot_alignment_bowtie2
      /scratch/ajauwena/binf_6110/project_2/cod_alignment_bowtie2

# Set a variable storing the working directory.
wd=/scratch/ajauwena/binf_6110/project_2

# Loop through each .sam file in the directory "burbot_alignment_bwa".
for file in $1/*
do

      # Extract the file's base name without the .fq.
      base_name=$(echo ${file} | cut -d '/' -f 7 | cut -d '.' -f 1)

      # Obtain the number of reads mapped.
      reads_mapped=$(samtools stats ${file} | grep 'reads mapped:' | awk
      '{print $4}')

      # Obtain the number of raw total sequences.
      raw_total_sequences=$(samtools stats ${file} | grep 'raw total
      sequences:' | awk '{print $5}')

      # Obtain the error rate.
      error_rate=$(samtools stats ${file} | grep 'error rate:' | awk '{print
      $4}')
```

```
        # Obtain the average length.
        average_length=$(samtools stats ${file} | grep 'average length:' | awk
        '{print $4}')

        # Obtain the average quality.
        average_quality=$(samtools stats ${file} | grep 'average quality:' |
        awk '{print $4}')

        # Calculate the alignment completeness by dividing the reads mapped by
        the raw total sequences.
        alignment_completeness=$(echo "scale=3;
        ${reads_mapped}/${raw_total_sequences}" | bc)

        # Output all the results into a .txt file in the appropriate directory.
        echo "Reads_Mapped: ${reads_mapped} Raw_Total_Sequences:
        ${raw_total_sequences} Error_Rate: ${error_rate} Average_Length:
        ${average_length} Average_Quality: ${average_quality}
        Alignment_Completeness: ${alignment_completeness}" >
        ${wd}/alignment_statistics/burbot_statistics_bwa/${base_name}.txt

done

# Loop through each .sam file in the directory "cod_alignment_bwa".
for file in $2/*
do

        # Extract the file's base name without the .fq.
        base_name=$(echo ${file} | cut -d '/' -f 7 | cut -d '.' -f 1)

        # Obtain the number of reads mapped.
        reads_mapped=$(samtools stats ${file} | grep 'reads mapped:' | awk
        '{print $4}')

        # Obtain the number of raw total sequences.
        raw_total_sequences=$(samtools stats ${file} | grep 'raw total
        sequences:' | awk '{print $5}')
```

```
        # Obtain the error rate.
        error_rate=$(samtools stats ${file} | grep 'error rate:' | awk '{print
        $4}')

        # Obtain the average length.
        average_length=$(samtools stats ${file} | grep 'average length:' | awk
        '{print $4}')

        # Obtain the average quality.
        average_quality=$(samtools stats ${file} | grep 'average quality:' |
        awk '{print $4}')

        # Calculate the alignment completeness by dividing the reads mapped by
        the raw total sequences.
        alignment_completeness=$(echo "scale=3;
        ${reads_mapped}/${raw_total_sequences}" | bc)

        # Output all the results into a .txt file in the appropriate directory.
        echo "Reads_Mapped: ${reads_mapped} Raw_Total_Sequences:
        ${raw_total_sequences} Error_Rate: ${error_rate} Average_Length:
        ${average_length} Average_Quality: ${average_quality}
        Alignment_Completeness: ${alignment_completeness}" >
        ${wd}/alignment_statistics/cod_statistics_bwa/${base_name}.txt

done

# Loop through each .sam file in the directory "burbot_alignment_bowtie2".
for file in $3/*
do

        # Extract the file's base name without the .fq.
        base_name=$(echo ${file} | cut -d '/' -f 7 | cut -d '.' -f 1)

        # Obtain the number of reads mapped.
        reads_mapped=$(samtools stats ${file} | grep 'reads mapped:' | awk
        '{print $4}')

        # Obtain the number of raw total sequences.
```

```
        raw_total_sequences=$(samtools stats ${file} | grep 'raw total
        sequences:' | awk '{print $5}')

        # Obtain the error rate.
        error_rate=$(samtools stats ${file} | grep 'error rate:' | awk '{print
        $4}')

        # Obtain the average length.
        average_length=$(samtools stats ${file} | grep 'average length:' | awk
        '{print $4}')

        # Obtain the average quality.
        average_quality=$(samtools stats ${file} | grep 'average quality:' |
        awk '{print $4}')

        # Calculate the alignment completeness by dividing the reads mapped by
        the raw total sequences.
        alignment_completeness=$(echo "scale=3;
        ${reads_mapped}/${raw_total_sequences}" | bc)

        # Output all the results into a .txt file in the appropriate directory.
        echo "Reads_Mapped: ${reads_mapped} Raw_Total_Sequences:
        ${raw_total_sequences} Error_Rate: ${error_rate} Average_Length:
        ${average_length} Average_Quality: ${average_quality}
        Alignment_Completeness: ${alignment_completeness}" >
        ${wd}/alignment_statistics/burbot_statistics_bowtie2/${base_name}.txt

done

# Loop through each .sam file in the directory "cod_alignment_bowtie2".
for file in $4/*
do

        # Extract the file's base name without the .fq.
        base_name=$(echo ${file} | cut -d '/' -f 7 | cut -d '.' -f 1)

        # Obtain the number of reads mapped.
```

```
        reads_mapped=$(samtools stats ${file} | grep 'reads mapped:' | awk
        '{print $4}')

        # Obtain the number of raw total sequences.
        raw_total_sequences=$(samtools stats ${file} | grep 'raw total
        sequences:' | awk '{print $5}')

        # Obtain the error rate.
        error_rate=$(samtools stats ${file} | grep 'error rate:' | awk '{print
        $4}')

        # Obtain the average length.
        average_length=$(samtools stats ${file} | grep 'average length:' | awk
        '{print $4}')

        # Obtain the average quality.
        average_quality=$(samtools stats ${file} | grep 'average quality:' |
        awk '{print $4}')

        # Calculate the alignment completeness by dividing the reads mapped by
        the raw total sequences.
        alignment_completeness=$(echo "scale=3;
        ${reads_mapped}/${raw_total_sequences}" | bc)

        # Output all the results into a .txt file in the appropriate directory.
        echo "Reads_Mapped: ${reads_mapped} Raw_Total_Sequences:
        ${raw_total_sequences} Error_Rate: ${error_rate} Average_Length:
        ${average_length} Average_Quality: ${average_quality}
        Alignment_Completeness: ${alignment_completeness}" >
        ${wd}/alignment_statistics/cod_statistics_bowtie2/${base_name}.txt

done
```

Appendix 3: The Python script "alignment_statistics_csv_burbot_bwa.py"

Before running this Python script, scp all the .txt files from Unix into the directory "burbot_statistics_bwa" in my local computer.

```
import os
import pandas as pd

# Create an empty list "list_burbot_statistics_bwa".
list_burbot_statistics_bwa = []

# Set a variable storing the absolute path to the directory
"burbot_statistics_bwa" in my local computer.
path_burbot_statistics_bwa =
r"C:\Users\ajauw\OneDrive\Documents\m_a\w\education\master's\mbinf\w23\binf_6
110\projects\p2\alignment_statistics_txt\burbot_statistics_bwa"

# Loop through all the .txt files in the directory "burbot_statistics_bwa".
for txt_file in os.listdir(path_burbot_statistics_bwa):

      # Open the file.
      with open(os.path.join(path_burbot_statistics_bwa, txt_file)) as file:

            # Obtain the name of the file.
            name = txt_file

            # Read the line in the file (each file only contains one line).
            line = file.readlines()

            # Store the elements in each line as a list, split by the
            whitespace character.
            element = line[0].split(' ')

            # Obtain the number of reads mapped.
            reads_mapped = float(element[1])

            # Obtain the raw total sequences.
            raw_total_sequences = float(element[3])
```

```
            # Obtain the error rate.
            error_rate = float(element[5])

            # Obtain the average length.
            average_length = float(element[7])

            # Obtain the average quality.
            average_quality = float(element[9])

            # Obtain the alignment completeness.
            alignment_completeness = float(element[11])

            # Append the name and all the obtained statistics to the list
            "list_burbot_statistics_bwa".
            list_burbot_statistics_bwa.append([name, reads_mapped,
            raw_total_sequences, error_rate, average_length, average_quality,
            alignment_completeness])
            # "list_burbot_statistics_bwa" is a list of lists.

# Create a data frame "df_burbot_statistics_bwa" from the list
"list_burbot_statistics_bwa".
df_burbot_statistics_bwa = pd.DataFrame(list_burbot_statistics_bwa,
columns=['File_Name', 'Reads_Mapped', 'Raw_Total_Sequences', 'Error_Rate',
'Average_Length', 'Average_Quality', 'Alignment_Completeness'])

# Output the data frame "df_burbot_statistics_bwa" as a .csv file.
df_burbot_statistics_bwa.to_csv('df_burbot_statistics_bwa.csv')
```

Before running this Python script, scp all the .txt files from Unix into the directory "cod_statistics_bwa" in my local computer.

```
import os
import pandas as pd

# Create an empty list "list_cod_statistics_bwa".
list_cod_statistics_bwa = []

# Set a variable storing the absolute path to the directory
"cod_statistics_bwa" in my local computer.
path_cod_statistics_bwa =
r"C:\Users\ajauw\OneDrive\Documents\m_a\w\education\master's\mbinf\w23\binf_6
110\projects\p2\alignment_statistics_txt\cod_statistics_bwa"

# Loop through all the .txt files in the directory "cod_statistics_bwa".
for txt_file in os.listdir(path_cod_statistics_bwa):

      # Open the file.
      with open(os.path.join(path_cod_statistics_bwa, txt_file)) as file:

            # Obtain the name of the file.
            name = txt_file

            # Read the line in the file (each file only contains one line).
            line = file.readlines()

            # Store the elements in each line as a list, split by the
            whitespace character.
            element = line[0].split(' ')

            # Obtain the number of reads mapped.
            reads_mapped = float(element[1])

            # Obtain the raw total sequences.
            raw_total_sequences = float(element[3])
```

```
            # Obtain the error rate.
            error_rate = float(element[5])

            # Obtain the average length.
            average_length = float(element[7])

            # Obtain the average quality.
            average_quality = float(element[9])

            # Obtain the alignment completeness.
            alignment_completeness = float(element[11])

            # Append the name and all the obtained statistics to the list
            "list_cod_statistics_bwa".
            list_cod_statistics_bwa.append([name, reads_mapped,
            raw_total_sequences, error_rate, average_length, average_quality,
            alignment_completeness])
            # "list_cod_statistics_bwa" is a list of lists.

# Create a data frame "df_cod_statistics_bwa" from the list
"list_cod_statistics_bwa".
df_cod_statistics_bwa = pd.DataFrame(list_cod_statistics_bwa,
columns=['File_Name', 'Reads_Mapped', 'Raw_Total_Sequences', 'Error_Rate',
'Average_Length', 'Average_Quality', 'Alignment_Completeness'])

# Output the data frame "df_cod_statistics_bwa" as a .csv file.
df_cod_statistics_bwa.to_csv('df_cod_statistics_bwa.csv')
```

Appendix 5: The Python script "alignment_statistics_csv_burbot_bowtie2.py"

Before running this Python script, scp all the .txt files from Unix into the directory "burbot_statistics_bowtie2" in my local computer.

```
import os
import pandas as pd

# Create an empty list "list_burbot_statistics_bowtie2".
list_burbot_statistics_bowtie2 = []

# Set a variable storing the absolute path to the directory
"burbot_statistics_bowtie2" in my local computer.
path_burbot_statistics_bowtie2 =
r"C:\Users\ajauw\OneDrive\Documents\m_a\w\education\master's\mbinf\w23\binf_6
110\projects\p2\alignment_statistics_txt\burbot_statistics_bowtie2"

# Loop through all the .txt files in the directory
"burbot_statistics_bowtie2".
for txt_file in os.listdir(path_burbot_statistics_bowtie2):

      # Open the file.
      with open(os.path.join(path_burbot_statistics_bowtie2, txt_file)) as
      file:

            # Obtain the name of the file.
            name = txt_file

            # Read the line in the file (each file only contains one line).
            line = file.readlines()

            # Store the elements in each line as a list, split by the
            whitespace character.
            element = line[0].split(' ')

            # Obtain the number of reads mapped.
            reads_mapped = float(element[1])
```

```
            # Obtain the raw total sequences.
            raw_total_sequences = float(element[3])

            # Obtain the error rate.
            error_rate = float(element[5])

            # Obtain the average length.
            average_length = float(element[7])

            # Obtain the average quality.
            average_quality = float(element[9])

            # Obtain the alignment completeness.
            alignment_completeness = float(element[11])

            # Append the name and all the obtained statistics to the list
            "list_burbot_statistics_bowtie2".
            list_burbot_statistics_bowtie2.append([name, reads_mapped,
            raw_total_sequences, error_rate, average_length, average_quality,
            alignment_completeness])
            # "list_burbot_statistics_bowtie2" is a list of lists.

# Create a data frame "df_burbot_statistics_bowtie2" from the list
"list_burbot_statistics_bowtie2".
df_burbot_statistics_bowtie2 = pd.DataFrame(list_burbot_statistics_bowtie2,
columns=['File_Name', 'Reads_Mapped', 'Raw_Total_Sequences', 'Error_Rate',
'Average_Length', 'Average_Quality', 'Alignment_Completeness'])

# Output the data frame "df_burbot_statistics_bowtie2" as a .csv file.
df_burbot_statistics_bowtie2.to_csv('df_burbot_statistics_bowtie2.csv')
```

Appendix 6: The Python script "alignment_statistics_csv_cod_bowtie2.py"

Before running this Python script, scp all the .txt files from Unix into the directory "cod_statistics_bowtie2" in my local computer.

```
import os
import pandas as pd

# Create an empty list "list_cod_statistics_bowtie2".
list_cod_statistics_bowtie2 = []

# Set a variable storing the absolute path to the directory
"cod_statistics_bowtie2" in my local computer.
path_cod_statistics_bowtie2 =
r"C:\Users\ajauw\OneDrive\Documents\m_a\w\education\master's\mbinf\w23\binf_6
110\projects\p2\alignment_statistics_txt\cod_statistics_bowtie2"

# Loop through all the .txt files in the directory "cod_statistics_bowtie2".
for txt_file in os.listdir(path_cod_statistics_bowtie2):

      # Open the file.
      with open(os.path.join(path_cod_statistics_bowtie2, txt_file)) as file:

            # Obtain the name of the file.
            name = txt_file

            # Read the line in the file (each file only contains one line).
            line = file.readlines()

            # Store the elements in each line as a list, split by the
            whitespace character.
            element = line[0].split(' ')

            # Obtain the number of reads mapped.
            reads_mapped = float(element[1])

            # Obtain the raw total sequences.
            raw_total_sequences = float(element[3])
```

```
            # Obtain the error rate.
            error_rate = float(element[5])

            # Obtain the average length.
            average_length = float(element[7])

            # Obtain the average quality.
            average_quality = float(element[9])

            # Obtain the alignment completeness.
            alignment_completeness = float(element[11])

            # Append the name and all the obtained statistics to the list
            "list_cod_statistics_bowtie2".
            list_cod_statistics_bowtie2.append([name, reads_mapped,
            raw_total_sequences, error_rate, average_length, average_quality,
            alignment_completeness])
            # "list_cod_statistics_bowtie2" is a list of lists.

# Create a data frame "df_cod_statistics_bowtie2" from the list
"list_cod_statistics_bowtie2".
df_cod_statistics_bowtie2 = pd.DataFrame(list_cod_statistics_bowtie2,
columns=['File_Name', 'Reads_Mapped', 'Raw_Total_Sequences', 'Error_Rate',
'Average_Length', 'Average_Quality', 'Alignment_Completeness'])

# Output the data frame "df_cod_statistics_bowtie2" as a .csv file.
df_cod_statistics_bowtie2.to_csv('df_cod_statistics_bowtie2.csv')
```