Actividad 2: Manipulación y formateo de archivos: Formato FASTQ y FASTA

Objetivo

El propósito de esta actividad es desarrollar un flujo de trabajo bioinformático integral, conocido como *pipeline*, para procesar datos biológicos. La intención es que el estudiante adquiera destrezas para interactuar con el sistema operativo mediante la línea de comandos y pueda desarrollar scripts en Shell para abordar diferentes desafíos bioinformáticos, centrándose en dos formatos de texto específicos: **FASTQ y FASTA.**

Detalles sobre la entrega

- La entrega se realizará utilizando este documento como plantilla; adicionando capturas de pantalla que ilustren el código empleado y su ejecución. Será esencial incluir el nombre de usuario completo (*prompt*) en las capturas de pantalla y se recomienda ajustar la resolución al máximo posible.
- Para cada comando empleado, deberá realizar una **breve explicación** donde indique su función y describa las opciones o parámetros utilizados. Para facilitar la lectura, emplee un color o formato de letra distinto al del enunciado propuesto.
- La entrega se realizará a través del Campus VIU, en un archivo **PDF** único descomprimido.

Parte I: Creación del directorio de trabajo.

Para inicializar este ejercicio, deberá crear y organizar un nuevo directorio de trabajo que contenga los elementos clave que se explorarán. La estructura de organización busca que cualquier persona no familiarizada con el proyecto pueda examinar los archivos en la computadora y entender en detalle lo realizado y por qué.

Utilizando comandos de Linux en la terminal, deberá crear la siguiente estructura de directorios para el proyecto, identificándolo de la siguiente manera:

User_proyect_year_publication donde:

- **User**: Corresponde al apellido del estudiante.
- **Project:** Es el nombre del proyecto hipotético en el que está trabajando.
- **Year**: Representa el año de la publicación o investigación en curso.
- **Publication**: Es el nombre de la revista donde planea publicar sus resultados.

Este directorio, en mi caso, ***Soler_humancells_2024_Nature*** será creado dentro del directorio ***Documents,*** que a su vez contendrá tres directorios ***data, code*** y ***submission***. Además, el directorio ***data*** tendrá a su vez dos subdirectorios llamados ***raw*** y **processed.**

Parte II: Obtención de datos

Ahora procederá a obtener el conjunto de datos con el cual trabajará. El archivo en cuestión puede encontrarse en la siguiente ruta del campus virtual y

Actividades/Portafolio de pruebas aplicativas/Prueba aplicativa 2/SRR1984406_1.fastq

Seguidamente, guarde este archivo en el directorio ***data/raw*** y responda a las preguntas indicadas, adjuntando los comandos empleados y capturas de pantalla.

- **P1.** ¿Cuántas secuencias podemos encontrar en el archivo? **(0,5 pts)**

```
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$ grep -c "^@SRR" SRR1984406_1.fastq
8246
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$
```

  Podemos encontrar 8246 secuencias.
  El comando grep con la opción -c nos permite contar los números de línea para un patrón dado, en este caso "^@SRR". El ^ delante del patrón se refiere a las líneas que empiecen por ese patrón. Dado que todas las secuencias en un archivo fastq tienen un encabezado con el identificador que empieza así, si contamos el número de líneas en las que aparece el encabezado conoceremos también el número de secuencias.

- **P2.** ¿Cuál es la longitud mínima y máxima de las secuencias incluidas en el archivo? ¿Cuántas secuencias tienen una longitud igual al valor de longitud mínima previamente computada? **(1,5 pts)**

```
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$ awk '/^@SRR/ {print $3}' SRR1984406_1.fastq > lo
ngitudes.txt
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$ sort -V longitudes.txt | head -1
length=75
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$ sort -V longitudes.txt | tail -1
length=135
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$ grep -c "length=75" SRR1984406_1.fastq
12
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$
```

  Visualizando el archivo se observa que en el encabezado de las secuencias hay una tercera columna donde aparece la longitud de la secuencia "length=". El comando awk permie buscar un patrón, en este caso ^@SRR, para que busque en las líneas que empiezan por ese patrón, y realizar una acción, {print $3}, para que imprima la columna 3 de esas líneas, y lo redirija a un nuevo archivo llamado longitudes.txt, con >.

  El comando sort nos permite con la opción -V ordenar el archivo teniendo en cuenta que trabaja con texto y números, por lo que ordenará las longitudes de las secuencias de menor a mayor. Con una tubería (|) se transfiere el resultado y mediante head -1 imprime la primera línea y tail -1 imprime la última línea, por lo se obtendrá el valor mínimo y máximo de longitud de secuencia, respectivamente.
  La longitud mínima de las secuencias es 75 y la longitud máxima 135.

El comando grep -c permite contar el número de líneas en las que aparece el patrón “length=75”, que es la longitud mínima, que es equivalente al número de secuencias que tienen esa longitud.
Hay 12 secuencias con una longitud igual al valor de longitud mínima.

- **P3.** ¿Cuántas veces aparece el patrón “ATGATG” en las secuencias del archivo descargado? **(0,5 pts)**

```
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$ grep -o "ATGATG" SRR1984406_1.f
astq | wc -w
835
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$
```

El comando grep con la opción -o solo muestra las partes de una línea que coinciden con un patrón dado, en este caso “ATGATG”. A través de una tubería (|) se transfiere el resultado al comando wc, que con la opción -w permite contar palabras, por tanto, contará todas las veces que aparece “ATGATG” en las secuencias del archivo.

- **P4.** Con el propósito de transformar el archivo de formato FASTQ a FASTA, deberá seleccionar las líneas correspondientes que representan el encabezado (header) y la secuencia de cada lectura. El resultado se almacenará en un archivo llamado *all_sequences.fasta* dentro del directorio *data/processed*. Incluya una captura de pantalla con el código empleado para realizar esta conversión de formato y visualice las 4 primeras líneas del archivo *all_sequences.fasta* en el directorio determinado. **(1,5 pts)**

```
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$ sed -n 1~4s'/^@/>/p;2~4p' SRR1984406_1.fastq > /
home/msevillanogonzalez/Documentos/Sevillano_klebsiellaresistencias_2024_Science/data/processed/all_sequences.fa
sta
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 raw]$ cd ..
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 data]$ cd processed/
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ cat all_sequences.fasta | head -4
>SRR1984406.1 1 length=135
GACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCACATCTGACTTCCAGTATCCATCCGAAGTTCTCCATTCAATAGTGAGGAATCTGACGACTGCCATCTGAACGTGT
GGAATCAACGGAGCCACATCTGA
>SRR1984406.2 2 length=134
TTTGGGAATTTCCTGTATCCATCCGAAGTTCTCCATTCAATAGTGAGGAATCTGACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCACATCTGACTAGTGCCCAGCAT
GAGCGACTCCACCGCCATTGGG
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$
```

El comando sed -n indica que no se impriman todas las líneas por defecto. Con 1~4s nos permite sustituir en la primera línea de cada 4 el @ al inicio de la línea, indicado con ^, por un “>”, que es como comienza el encabezado de las secuencias en un archivo fasta. Esto seguido de la opción p lo va a imprimir y se concatena con otra acción a través del “;”, 2~4p que permite la impresión de la segunda línea de cada 4 líneas, es decir, cada secuencia.
Todo este resultado se redirige a otro archivo con “>” indicando la ruta donde se encontrará y el nombre al final de la ruta.

El comando cat permite visualizar todo el contenido del archivo, que mediante la tubería (|) permite transferirlo al comando head -4 que permite imprimir las 4 primeras líneas.

- **P5.** Posterior a la conversión, seleccione las primeras 5 lecturas del archivo all_sequences.fasta (con su cabecera y su secuencia asociada) y cree archivos individuales denominados secuencia1.fasta, secuencia2.fasta, y así sucesivamente. En cada uno de ellos, deberá modificar la cabecera, por Secuencia 1, Secuencia 2 y así, sucesivamente (no emplee ningún editor de texto manual para realizarlo).

  Como ejemplo, cada uno de ellos contendrá lo siguiente:

  >Secuencia 1
  GACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCACATCTGACTTCCAGTATCCATCCGAAGTTCTC
  CATTCAATAGTGAGGAATCTGACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCACATCTG

  Incluya una captura de pantalla con el código empleado y su ejecución, visualice el contenido de los archivos creados y con el comando *ls* muestre la creación de estos en el directorio *data/processed* **(1,25 pts)**

  El comando sed con la opción -n indica que no debe imprimir todas las líneas por defecto si no las que se especifican, en el primer caso, extrae las dos primeras líneas del archivo ("1,2p"). Este resultado se redirige con una tubería como entrada del siguiente comando, sed con la opción de sustituir la primera línea (1s) completa (.*) por >Secuencia1 y redirija la salida (>) a un nuevo archivo llamado secuencia1.fasta

  Como se pide que se creen archivos individuales para cada secuencia con su cabecera asociada de las primeras 5 lecturas del archivo, se repite el mismo comando, pero seleccionado de dos en dos las líneas siguientes y sustituyendo la cabecera del archivo por el nombre de la secuencia correspondiente.

```
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ sed -n "1,2p" all_sequences.fasta | sed '1s/.*/>Secuencia 1/'> sec
uencia1.fasta
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ cat secuencia1.fasta
>Secuencia 1
GACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCACATCTGACTTCCAGTATCCATCCGAAGTTCTCCATTCAATAGTGAGGAATCTGACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCACATCTGA
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ sed -n "3,4p" all_sequences.fasta | sed '1s/.*/>Secuencia 2/'> sec
uencia2.fasta
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ cat secuencia2.fasta
>Secuencia 2
TTTGGGAATTTCCTGTATCCATCCGAAGTTCTCCATTCAATAGTGAGGAATCTGACGACTGCCATCTGAACGTGTGGAATCAACGGAGCCACATCTGACTAGTGCCCAGCATGAGCGACTCCACCGCCATTGGG
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ sed -n "5,6p" all_sequences.fasta | sed '1s/.*/>Secuencia 3/'> sec
uencia3.fasta
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ cat secuencia3.fasta
>Secuencia 3
ATGCTGGGCACTAGTCAAATGTGGCTCCGTTGATTCCACACGTTCAGATGGCAATCGTCAGATTCCTCACTATTGAATGGAGAACTTCGGATGGATACTGGGTATCCGCGTTTTCCAGTATCCATCCGAAGTCT
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ sed -n "7,8p" all_sequences.fasta | sed '1s/.*/>Secuencia 4/'> sec
uencia4.fasta
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ cat secuencia4.fasta
>Secuencia 4
CCTCTGTAGTCAGCCCAATGGCGGTGGAGTCGCTCATGCTGGGCACTAGTCAGATGTGGCTCCGTTGATTCCACACGTTCAGATGGCAGTCGTCAGATTCCTCACTATTGAATGGGGATC
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ sed -n "9,10p" all_sequences.fasta | sed '1s/.*/>Secuencia 5/'> se
cuencia5.fasta
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ cat secuencia5.fasta
>Secuencia 5
CCTCTGTAGTCAGCCCAATGGCGGTGGAGTCGCTCATTCTGGGCACTAGTCAGGTGTGGCTCCGTTGATTCCACACGTTCAGATGGCAGTCGTCAGATTCCTCACTATTGAATGGGGATC
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$
```

```
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ ls
all_sequences.fasta  secuencia1.fasta  secuencia2.fasta  secuencia3.fasta  secuencia4.fasta  secuencia5.fasta
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$
```

- **P6**. Ahora deberá crear un script llamado *analyze_sequences.sh* en el directorio *code.* Este script deberá leer cada uno de los archivos, específicamente su secuencia, y reportar por la salida estándar estadísticas detalladas de las secuencias:

- Longitud de cada secuencia.
- Identificación del nucleótido inicial y final de cada secuencia.
- Porcentaje de contenido de GC para cada secuencia. Redondee el resultado a dos decimales.
- Determine con una estructura condicional si en alguna de las secuencias se encuentra el patrón “GGGG”. Si no lo encuentra reporte un mensaje de negación al usuario y si lo encuentra, reporte las secuencias donde lo ha encontrado y sustituya este patrón por la palabra “Mutation”.

Adicione una captura de pantalla que muestre la estructura del script, su ejecución y los resultados obtenidos **(2,5 pts)**

```bash
#!usr/bin/bash

ruta_archivos=/home/msevillanogonzalez/Documentos/Sevillano_klebsiellaresistencias_2024_Science/data/processed

# iterar sobre cada uno de los archivos e imprimir con el comando sed la segunda linea de cada archivo
for numero in {1..5}
do
        archivos="${ruta_archivos}/secuencia${numero}.fasta"
        secuencia=$(sed -n '2p' "$archivos")

# Obtener la longitud de la secuencia con el # antes de la variable
        length=${#secuencia}

# Obtener los nucleótidos iniciales y finales mediante extracción de subcadenas

        nucleotido_incial=${secuencia:0:1}
        nucleotido_final=${secuencia: -1}

# Calcular el porcentaje de GC contando cuantas veces aparece GC en la secuencia

        contenido_GC=$(echo "$secuencia" | grep -no '[GC]' | wc -l)
        porcentaje_GC=$(echo "scale=2;($contenido_GC/$length)*100" | bc)

echo "Archivo: secuencia${numero}.fasta"
echo "La longitud de la secuencia es de: $length nucleótidos"
echo "Nucleótido inicial: $nucleotido_incial"
echo "Nucleótido final: $nucleotido_final"
echo "Porcentaje de GC: $porcentaje_GC %"

# Buscar el patrón "GGGG" en cada secuencia y si se encuentra sustituirlo por "Mutation"

        if echo "$secuencia" | grep  "GGGG"
        then
                echo "$secuencia" | sed 's/GGGG/Mutation/g'
        else
                echo "No se encuentra el patrón "GGGG" en la secuencia${numero}.fasta"
        fi

echo ""

done
```

El bucle *for* permite iterar sobre una secuencia de números en un rango del 1 al 5, y *numero* es la variable que va a tomar esos valores en cada iteración. El comando *do* es el inicio de un bloque de código que se inicia en cada iteración.

La variable *archivos*, permite obtener cada uno de los 5 archivos, pues ${ruta_archivos} tiene predefinida la ruta donde se encuentra los archivos, y secuencia${numero}.fasta es el nombre del archivo con el número actual de cada iteración.

Con el comando sed – n se extrae la segunda fila del archivo actual de esa iteración (con la variable *archivos*), para acceder a la secuencia, que se almacena en la variable *secuencia*.

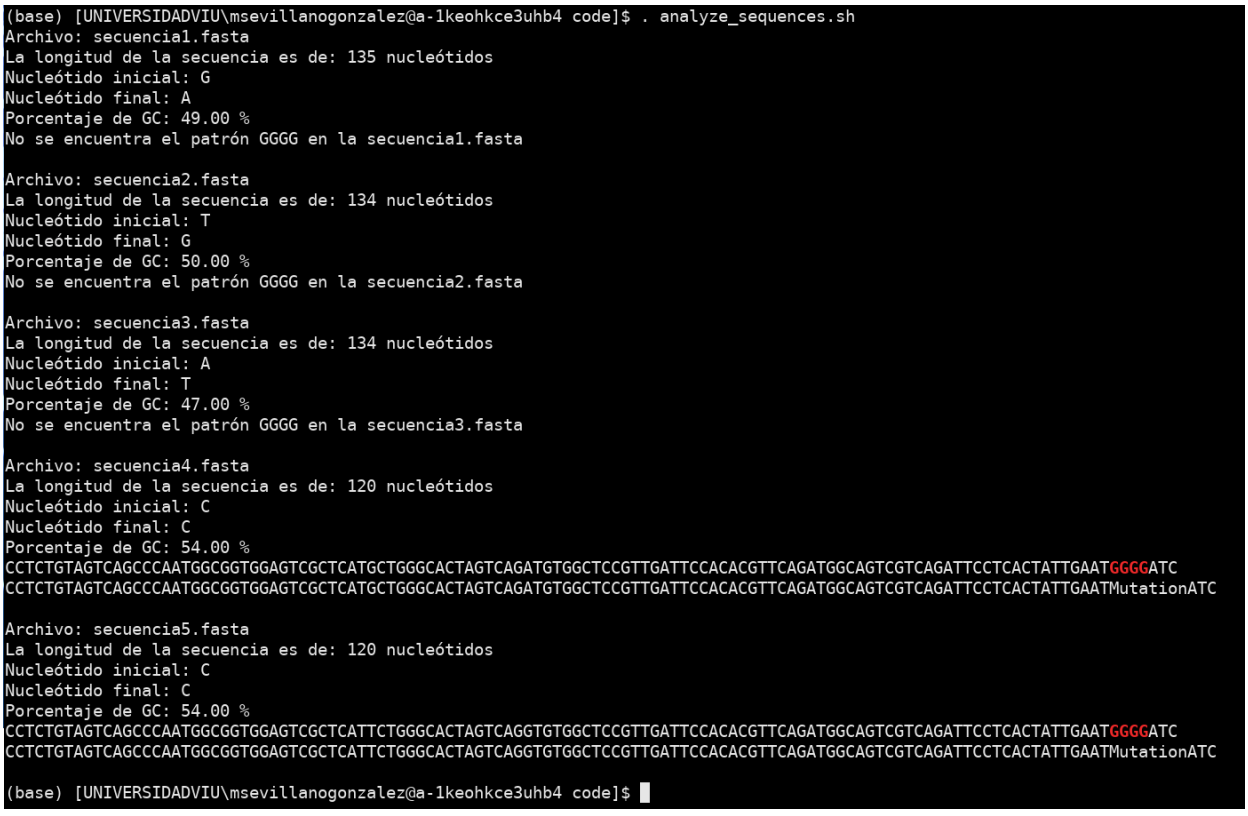
{#secuencia}, el # antes de la variable *secuencia*, nos permite contar el número de caracteres en la variable secuencia, y se guarda en otra variable llamada *length.*

La variable *nucleótido_inicial* extrae el primer nucleótido de cada secuencia, que está en posición 0.

La variable nucleótido_final extrae el último nucleótido de cada secuencia, que está en posición -1.

Para saber el contenido de GC, el comando grep con la opción -no permite extraer solo el patrón buscado, [GC], que se transfiere al comando wc -l, que cuenta el número de coincidencias con ese patrón.
bc es la calculadora, que permite realizar operaciones con decimales. Y con scale=2 le indicamos que el resultado tenga dos decimales.

El bucle *if permite* buscar el patrón "GGGG" en la secuencia con el comando grep. Con *then* le decimos que si encuentra ese patrón sustituya con el comando sed el patrón "GGGG" por la palabra Mutation.
Con *else* le decimos que si no encuentra el patrón que imprima con el comando echo que no encuentra el patrón en el archivo usado en esa interacción.
Con *fi* cerramos el bucle *if,* y con *donde* cerramos todo el bucle *for*.



- **P7**. Implemente otro script llamado *Metamorphosis.sh* en el directorio *code.* Este script deberá leer cada uno de los cinco archivos e invertir el orden de las secuencias de nucleótidos en cada registro fasta. Después, deberá convertir todas las secuencias a minúsculas y escribir las secuencias procesadas en un nuevo archivo fasta con un nombre que indique que se ha invertido y convertido a minúscula. Adicione una captura de pantalla que muestre la estructura del script, su ejecución y el contenido de los resultados obtenidos. **(2 pts)**

```
#!usr/bin/bash

ruta_archivos=/home/msevillanogonzalez/Documentos/Sevillano_klebsiellaresistencias_2024_Science/data/processed
ruta_salida=/home/msevillanogonzalez/Documentos/Sevillano_klebsiellaresistencias_2024_Science/data/processed

# Lee cada uno de los 5 archivos e invierte el orden de las secuencias y conviértelas en minúsculas
for numero in {1..5}
do
        archivos="${ruta_archivos}/secuencia${numero}.fasta"
        secuencia=$(sed -n '2p' "$archivos" | rev)  #invierte la secuencia
        minusculas=$(echo "$secuencia" | tr '[:upper:]' '[:lower:]') #convierte la secuencia a minúscula

        {
                awk 'NR==1' "$archivos"
                echo "$minusculas"
        } > "${ruta_salida}/secuencia${numero}_invertida_minuscula.fasta"

        cat "${ruta_salida}/secuencia${numero}_invertida_minuscula.fasta"

echo " "

done
~
~
```

El bucle *for* permite iterar sobre una secuencia de números en un rango del 1 al 5, y *numero* es la variable que va a tomar esos valores en cada iteración. El comando *do* es el inicio de un bloque de código que se inicia en cada iteración.

La variable *archivos*, permite obtener cada uno de los 5 archivos, pues ${ruta_archivos} tiene predefinida la ruta donde se encuentra los archivos, y secuencia${numero}.fasta es el nombre del archivo con el número actual de cada iteración.

La variable *secuencia* almacena la secuencia invertida de cada archivo, ya que el comando sed – n permite imprimir la segunda fila del archivo actual de esa iteración (con la variable *archivos*) y con la tubería lo dirige al comando rev, que permite invertir la línea.

La variable *minúscula*, almacena las secuencias en minúsculas. Con echo imprime la variable *secuencia* de la iteración actual y lo dirige con la tubería al comando tr (translate), que permite cambiar caracteres, como las mayúsculas [:upper:] por minúsculas [:lower:].

El comando awk permite seleccionar la primera línea de los archivos, con echo imprimir las secuencias de cada interacción en minúsculas e invertidas y después se redirige la salida para guardar cada secuencia y su cabecera en un archivo nuevo, con una ruta de salida previamente definida. El comando cat permite la visualización de los archivos por pantalla.

Con done cerramos el bucle.

```
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 code]$ . Metamorphosis.sh
>Secuencia 1
agtctacaccgaggcaactaaggtgtgcaagtctaccgtcagcagtctaaggagtgataacttacctcttgaagcctacctatgaccttcagtct
acaccgaggcaactaaggtgtgcaagtctaccgtcagcag

>Secuencia 2
gggttaccgccacctcagcgagtacgacccgtgatcagtctacaccgaggcaactaaggtgtgcaagtctaccgtcagcagtctaaggagtgata
acttacctcttgaagcctacctatgtcctttaagggttt

>Secuencia 3
tctgaagcctacctatgacctttgcgcctatgggtcataggtaggcttcaagaggtaagttatcactccttagactgctaacggtagacttgca
caccttagttgcctcggtgtaaactgatcacgggtcgta

>Secuencia 4
ctaggggtaagttatcactccttagactgctgacggtagacttgcacaccttagttgcctcggtgtagactgatcacgggtcgtactcgctgagg
tggcggtaacccgactgatgtctcc

>Secuencia 5
ctaggggtaagttatcactccttagactgctgacggtagacttgcacaccttagttgcctcggtgtggactgatcacgggtcttactcgctgagg
tggcggtaacccgactgatgtctcc

(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 code]$
```

```
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$ ls
all_sequences.fasta                  secuencia3_invertida_minuscula.fasta
secuencia1.fasta                     secuencia4.fasta
secuencia1_invertida_minuscula.fasta  secuencia4_invertida_minuscula.fasta
secuencia2.fasta                     secuencia5.fasta
secuencia2_invertida_minuscula.fasta  secuencia5_invertida_minuscula.fasta
secuencia3.fasta
(base) [UNIVERSIDADVIU\msevillanogonzalez@a-1keohkce3uhb4 processed]$
```

- **P8.** Para concluir el proyecto, genere una copia de esta actividad cumplimentada en formato PDF y trasládela al directorio submission. Además, incluya una captura de pantalla con la estructura de directorios actual usando el comando *tree* desde el terminal **(0,25 pts).**