**Ejercicios - Clase 1**

1. **Averiguar si un archivo está correcto.**

Imagina que tenemos un árbol que está causando un error en un programa que utiliza newick como formato de entrada. Muchas veces podemos tener un árbol con un pequeño error por desatención en el momento de edición o por cualquier otro error en el programa que generó el archivo. Como conocemos la estructura del archivo, podemos usar comandos simples del shell para saber si un árbol dado está correcto. Observa el código:”

newick="(A,(B,C),(D,E));"  
open\_count=$(echo "$newick" | grep -o "(" | wc -l)  
close\_count=$(echo "$newick" | grep -o ")" | wc -l)

if [ "$open\_count" -eq "$close\_count" ]; then  
 echo "La estructura Newick parece correcta."  
else  
 echo "La estructura Newick es incorrecta."  
fi

1. Usa a man para entender que hace grep –o y wc –l ;
2. Haga la ejecucion de echo "$newick" | grep -o "(" | wc -l separadamente del pipe para saber que se esta pasando en cada parte;
3. Testa estos arboles:

(((Homo\_sapiens:0.21,(Mus\_musculus:0.30,Rattus\_norvegicus:0.28)95:0.15):0.05,(Canis\_lupus:0.40,Felis\_catus:0.38)85:0.14):0.03,(((Gallus\_gallus:0.45,Anas\_platyrhynchos:0.48)80:0.20,(Taeniopygia\_guttata:0.50,Meleagris\_gallopavo:0.52)85:0.22)75:0.18,(Anolis\_carolinensis:0.60,(Xenopus\_tropicalis:0.70,(Danio\_rerio:0.75,Oryzias\_latipes:0.78)90:0.35)85:0.28)70:0.25)80:0.20,((Drosophila\_melanogaster:0.85,Caenorhabditis\_elegans:0.90)95:0.50,(Apis\_mellifera:0.65,Nasonia\_vitripennis:0.68)90:0.45)85:0.40,((Saccharomyces\_cerevisiae:0.95,Schizosaccharomyces\_pombe:1.00)98:0.55,(Arabidopsis\_thaliana:0.80,Vitis\_vinifera:0.82)92:0.48)88:0.42);

(((Homo\_sapiens:0.21,(Mus\_musculus:0.30,Rattus\_norvegicus:0.28)95:0.15),(Canis\_lupus:0.40,Felis\_catus:0.38)85:0.14):0.03,(((Gallus\_gallus:0.45,Anas\_platyrhynchos:0.48)80:0.20),(Taeniopygia\_guttata:0.50,Meleagris\_gallopavo:0.52)85:0.22)75:0.18,(Anolis\_carolinensis:0.60,(Xenopus\_tropicalis:0.70,(Danio\_rerio:0.75,Oryzias\_latipes:0.78))90:0.35)85:0.28)70:0.25,((Drosophila\_melanogaster:0.85,Caenorhabditis\_elegans:0.90)95:0.50,((Apis\_mellifera:0.65,Nasonia\_vitripennis:0.68)90:0.45)85:0.40,(Saccharomyces\_cerevisiae:0.95,Schizosaccharomyces\_pombe:1.00)98:0.55,(Arabidopsis\_thaliana:0.80,Vitis\_vinifera:0.82)92:0.48)88:0.42);

1. **"Obtener información rápida sobre el contenido de un archivo Nexus."**

Considere el archivo nexus que segue:

#NEXUS  
Begin taxa;  
Dimensions ntax=10;  
Taxlabels  
 Homo\_sapiens  
 Mus\_musculus  
 Rattus\_norvegicus  
 Canis\_lupus  
 Felis\_catus  
 Gallus\_gallus  
 Danio\_rerio  
 Xenopus\_tropicalis  
 Drosophila\_melano  
 Caenorhabditis\_ele  
;  
End;  
Begin data;  
Dimensions ntax=10 nchar=30;  
Format datatype=dna missing=? gap=- matchchar=.;  
Matrix  
Homo\_sapiens ATGCTAGCTAGCTCGGATCTAGCTAGCTCG  
Mus\_musculus .T...........T.....T.........A  
Rattus\_norvegicus .T.........A..........C......G  
Canis\_lupus .TC.....C..G......G..........A  
Felis\_catus .T..........A.................  
Gallus\_gallus GT.......T.GC....T..T...T...G  
Danio\_rerio TTA........G......G..........A  
Xenopus\_tropicalis GT.......A.G.....T...........  
Drosophila\_melano CCG.......TG.GT.T...T...T...A  
Caenorhabditis\_ele TT..........G.....T...........  
;  
End;  
Begin trees;  
 Tree tree1 = ((Homo\_sapiens, Mus\_musculus), (((Canis\_lupus, Felis\_catus), Gallus\_gallus), (Danio\_rerio, Xenopus\_tropicalis)), (Drosophila\_melano, Caenorhabditis\_ele));  
 Tree tree2 = ((Homo\_sapiens, (Mus\_musculus, Rattus\_norvegicus)), (Canis\_lupus, (Felis\_catus, Gallus\_gallus)), (Danio\_rerio, (Xenopus\_tropicalis, Drosophila\_melano)), Caenorhabditis\_ele);  
 Tree tree3 = ((Homo\_sapiens, Mus\_musculus, Rattus\_norvegicus), (Canis\_lupus, Felis\_catus), ((Gallus\_gallus, Danio\_rerio), (Xenopus\_tropicalis, Drosophila\_melano, Caenorhabditis\_ele)));  
 Tree tree4 = (((Homo\_sapiens, Mus\_musculus), Rattus\_norvegicus), ((Canis\_lupus, Felis\_catus), Gallus\_gallus), (Danio\_rerio, (Xenopus\_tropicalis, Drosophila\_melano)), Caenorhabditis\_ele);  
 Tree tree5 = ((Homo\_sapiens, (Mus\_musculus, (Rattus\_norvegicus, (Canis\_lupus, Felis\_catus)))), ((Gallus\_gallus, Danio\_rerio), (Xenopus\_tropicalis, (Drosophila\_melano, Caenorhabditis\_ele))));  
End;

Utiliza los comandos del ejercicio anterior para averiguar rapidamente: el número de árboles del fichero nexus, cuantos taxons hay y el cumprimiento de las secuencias en el alineamiento.

1. **Transformando las lecturas SRA en 2 archivos separados**

¿Cuál es el patrón que delimita que una read sea R1 o R2?

zcat SRR24054051.fastq.gz | head -n 32

¿Cuántas reads hay?

zcat SRR24054051.fastq.gz | grep -c "^@"

Revisa el script split.py. No necesitas entender python3, solo mira con cuidado e intenta encontrar en qué línea el script detecta si es R1 o R2.

Usando el script split.py para dividir los archivos fastq.gz que descargamos de NCBI:

python3 script/split.py SRR24054051.fastq.gz SRR24054051\_R1.fastq.gz SRR24054051\_R2.fastq.gz

python3 script/split.py SRR24053926.fastq.gz SRR24053926\_R1.fastq.gz SRR24053926\_R2.fastq.gz

python3 script/split.py SRR24053947.fastq.gz SRR24053947\_R1.fastq.gz SRR24053947\_R2.fastq.gz

Ver el resultado.

Ejemplo para uno de los archivos:  
  
zcat SRR24054051\_R1.fastq | grep -c "^@"

zcat SRR24054051\_R2.fastq | grep -c "^@"

**¿Tiene sentido el número de lecturas al final?**

1. **¿Qué hace el siguiente comando?**

AWK es un lenguaje de programación diseñado para el procesamiento y análisis de datos textuales. Es particularmente potente para manipular y analizar archivos estructurados y datos tabulares. AWK se destaca por su capacidad para buscar patrones dentro de un archivo y realizar acciones sobre los segmentos de texto que coinciden con esos patrones.

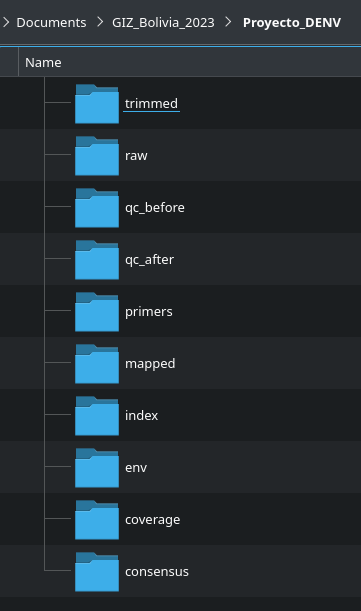
No siempre es necesario entender todo el código awk y hay muchos ejemplos útiles de archivos bioinformáticos en Internet que podemos utilizar para ayudarnos en nuestras tareas.

Ejecuta el comando abajo usando como entrada (archivo.fasta) las referencias de DENV que descargamos anteriormente.

awk '/^>/ {if (seq) print length(seq); seq=""; print; next} {seq = seq $0} END {print length(seq)}' archivo.fasta > result\_misterio.txt

1. **Creación de una estructura de directorios para nuestro proyecto de bioinformática**

Para nuestro proyecto de secuenciación, organizaremos los datos crudos, scripts, resultados y exportacion de entornos el la seguinte estructura:



Crie un script en shell que recibe un directorio de entrada como parametro y haga la creacion automarica de la estrucutura de directorios en la imagen.

La ejecución del script debe ser como en el ejemplo:

./crear\_direc.sh “directorio-padre” “nombre-del-proyecto”  
./crear\_direc.sh “/home/murilo” “DENV2023”

1. **Creacion del entorno virtual para nuestros proximos pasos**

Utilizar a mamba para crear un entorno virtual con el nombre “NGS”. Crear adicionando a dos channels bioconda y conda-forge (-c biconda -c conda forge) y con los paquetes:

* fastqc
* trim-galore
* samtools=1.18
* ivar
* bwa

Nota que hay una restriction de version para samtools.

1. **Exportar las configuraciones de entornos virtuales**

Utiliza mamba env export para crear un archivo .yaml que contenga todos los paquetes y librerías para el entorno ngs en el directorio env (resultante de la estructura de directorios del ejercicio 4).

¿Sólo están los paquetes que le pedimos a mamba que instalara? Si es así, ¿cuáles pueden ser los otros programas?

1. **Utilizando Variables y Parámetros en un Script para FastQC**

En este ejercicio, aprenderemos a escribir un script Bash que utiliza variables y parámetros para ejecutar la herramienta FastQC de forma flexible. Idealmente, haremos esto para otros programas.

En lugar de escribir comandos específicos cada vez, usaremos variables y parámetros para hacer el proceso más eficiente y personalizable.

**Creación del Script:**

* Crea un archivo de script llamado run\_fastqc.sh.
* Define dos variables: archivo\_entrada para el nombre del archivo de secuencia y directorio\_salida para el directorio donde se guardarán los resultados de FastQC.
* Permite que estas variables se pasen como parámetros al script.
* Codigo:

#!/bin/bash  
  
#activar al entorno con fastqc instalado  
conda activate ngs  
  
#Obtener parámetros de entrada  
archivo\_r1=$1  
archivo\_r2=$2  
directorio\_salida=$3  
  
# Verificar si se proporcionaron suficientes argumentos  
if [ "$#" -ne 3 ]; then   
 echo "Uso: $0 <R1> <R2> <directorio\_de\_salida>"   
 exit 1   
fi

# Verificar si el directorio de salida existe, si no, crearlo

if [ ! -d "$directorio\_salida" ]; then  
 echo "El directorio de salida '$directorio\_salida' no existe. Creando directorio..."  
 mkdir -p "$directorio\_salida"  
fi

# Ejecutar FastQC con variables y parámetros   
fastqc --threads 8 --outdir $directorio\_salida $archivo\_r1 $archivo\_r2

**Ejecución del Script:**

* Debes ejecutar el script proporcionando el nombre del archivo de entrada y el directorio de salida como argumentos. Ejemplo:

./run\_fastqc.sh mi\_secuencias\_r1.fastq.gz mi\_secuencias\_r2.fastq.gz resultados\_fastqc

Recuerda que “resultado\_fastqc” hay que ser una ruta. Podemos utilizar la estructura criada en el ejercicio 5.

1. **Modificando a PATH para la instalacion de softwares**

A veces nos encontramos con software nuevo e interesante que no está disponible en los repositorios de nuestros sistemas operativos. Cuando esto ocurre, normalmente tenemos que descargar el software, guardarlo en un directorio y añadir este directorio al PATH.

Como ejercicio, vamos a instalar la última versión del estimador de filogenia IQTREE.

* Descarga iqtree-2.2.2.6-Linux.tar.gz de: http://www.iqtree.org/
* Muévelo a un directorio de programas de tu máquina. Puedes crear una carpeta llamada software en home. Ejemplo: 'mkdir /home/murilo/softwares'
* Navega a este directorio con cd. Usa el comando para descomprimir el archivo:

tar -xvf iqtree-2.2.2.6-Linux.tar.gz

* Observa que se ha creado una carpeta iqtree-2.2.2.6-Linux/bin que contiene el archivo iqtree2.
* En tu directorio personal hay un archivo oculto llamado '.bashrc'. Ábrelo en un editor de texto normal y añade la siguiente línea al final del archivo, como se muestra en el ejemplo:

export PATH=$PATH:</mi/nuevo/directorio/con/software>

, donde </mi/nuevo/directorio/con/software> en mi caso era

/home/murilo/Documents/software/iqtree-2.2.2.6-Linux/bin/

* El comando final será algo como:

export PATH=$PATH:/home/murilo/Documents/software/iqtree-2.2.2.6-Linux/bin/

* ¡UTILICE SIEMPRE RUTAS ABSOLUTAS EN EL PATH!
* Escriba iqtree2 en el terminal.
* ¿Qué ocurre?
* Cierra el terminal, ábrelo de nuevo y escribe iqtree2.