

1 | Entorno de Trabajo

1.1 | Línea de Comandos

La línea de comandos es una interfaz de texto que permite interactuar con el sistema operativo mediante comandos. Es fundamental al procesar datos bioinformáticos, ya que muchas herramientas carecen de interfaz gráfica y el uso de la terminal facilita la automatización y ejecución de tareas en entornos HPC o en la nube.

Un comando es una instrucción que se escribe en la terminal y se ejecuta al presionar Enter. La estructura típica de un comando es:

```
comando [subcomando] [opciones] <argumentos>
```

sh

Los <argumentos> suelen ser archivos de entrada y salida, mientras que las opciones modifican el comportamiento del comando. Estas pueden ser largas (`--opcion`) o cortas (`-o`), y a veces requieren argumentos. Finalmente, algunos comandos poseen [subcomandos].



Consejo

Para obtener ayuda sobre un comando, utiliza `--help` o `-h`. Por ejemplo: `git --help`.

Los corchetes (`[]`) indican argumentos opcionales, mientras que los obligatorios se escriben con corchetes angulares (`< >`) o sin ellos.

1.2 | Interacción con archivos en Linux

Para ejecutar herramientas desde la terminal, es necesario saber cómo encontrar y manipular archivos. A continuación, se presentan algunos de los comandos más básicos para interactuar con archivos en Linux.

ls - Listar archivos y directorios

```
ls                # Muestra los archivos en el directorio actual
ls -l            # Muestra detalles de los archivos
ls -a            # Incluye archivos ocultos en la lista
```

sh

pwd - Imprimir el directorio actual

```
pwd
```

sh

cd - Cambiar de directorio

```
cd data          # Entra a la carpeta indicada
cd ..            # Sube un nivel en el árbol de directorios
cd ../..         # Sube dos niveles
cd ~             # Regresa al directorio de inicio
```

sh

cp - Copiar archivos o directorios

```
cp config.yaml analisis/    # Copia el archivo a la ubicación destino
cp -r results results_bak   # Copia una carpeta y su contenido
```

sh

mv - Mover o renombrar archivos

```
mv sample01.fastq data/     # Mueve el archivo al destino
mv config.test.txt config.txt # Cambia el nombre del archivo
```

sh

rm - Eliminar archivos o directorios

```
rm test.txt          # Elimina el archivo
rm -r tmp            # Elimina una carpeta y su contenido
```

sh

mkdir – Crear directorios

```
mkdir data # Crea un directorio en la ubicación actual
mkdir -p analysis/quality # Crea directorios anidados
```

sh**head** / **tail** / **less** – Visualizar el contenido de archivos

```
less sequences.fasta # Permite desplazarse por el contenido
head -n 10 sequences.fasta # Muestra las primeras 10 líneas
tail -n 5 sequences.fasta # Muestra las últimas 5 líneas
```

sh**1.3 | Tipos de archivos relevantes al procesar datos bioinformáticos**

Tipo	Extensión	Contenido
FASTA	.fasta .fa .fna .fsa .faa	Secuencias biológicas
FASTQ	.fastq .fq	Secuencias biológicas con calidad
SAM	.sam	Alineamiento de secuencias contra una referencia
BAM	.bam .ubam	Alineamiento de secuencias contra una referencia (comprimido)
POD5	.pod5	Datos crudos de Oxford Nanopore
CSV / TSV	.csv .tsv	Datos tabulares separados por comas (CSV) o tabulaciones (TSV)

**Consejo**

Es común encontrar comprimir los archivos para ahorrar espacio de almacenamiento. Los archivos comprimidos tienen la extensión adicional `.gz`. Para descomprimirlos, debes utilizar el comando `gunzip`.

```
gunzip sequences.fastq.gz # Descomprime el archivo
head sequences.fastq # Visualiza las primeras líneas del archivo descomprimido
```

sh**1.4 | Gestión de entorno de trabajo con mamba**

Mamba es un gestor de paquetes que facilita la instalación y gestión de paquetes de Python y R, y también herramientas bioinformáticas. Estos paquetes se instalan en “ambientes”, que aíslan las dependencias de proyectos y ofrecen un entorno reproducible. Los ambientes se pueden activar y desactivar según sea necesario. Los paquetes están disponibles en “canales”, donde destacan [conda-forge](#) (paquetes de Python y R) y [bioconda](#) (herramientas bioinformáticas).

Algunas de las tareas más comunes que se pueden realizar con Mamba son:

Listar, crear y eliminar ambientes

```
mamba env list # Lista los ambientes disponibles
mamba create -n qc # Crea un ambiente llamado 'qc'
mamba env remove -n analysis # Elimina el ambiente 'analysis'
```

sh**Activar y desactivar ambientes**

```
mamba activate qc # Activa el ambiente 'qc'
mamba deactivate # Desactiva el ambiente activo
```

sh**Gestionar paquetes**

```
mamba list # Lista los paquetes instalados en el ambiente activo
mamba install python # Instala Python en el ambiente activo si no está presente
mamba install bioconda::samtools # Instala el paquete 'samtools' desde el canal 'bioconda'
mamba install bioconda::nanoq=0.9.0 # Instala una versión específica del paquete 'nanoq'
mamba update nanoq # Actualiza el paquete 'nanoq'
```

sh

```
mamba remove samtools # Desinstala el paquete 'samtools'
```

Exportar e importar ambientes

```
mamba env export -n qc > qc.yaml # Exporta el ambiente 'qc' a un archivo YAML
mamba env create -f qc.yaml # Crea un ambiente a partir del archivo YAML
```

[sh](#)

2 | Basecalling

Basecalling es el proceso por el cual se convierte la señal eléctrica captada por los dispositivos de Oxford Nanopore en secuencias de nucleótidos. Esta conversión requiere de algoritmos computacionales avanzados, como las redes neuronales LSTM y los transformers utilizados por las opciones de basecalling actuales. Este proceso es crucial porque define la calidad de los datos, lo que afectará directamente los análisis posteriores.

Oxford Nanopore ha desarrollado múltiples basecallers haciendo uso de diversos avances tecnológicos. El basecaller actual es **Dorado**, el cual se puede utilizar mediante línea de comandos, o a través de MinKNOW.

2.1 | Precisión del basecalling

Dorado cuenta con múltiples modelos de basecalling, cada uno con diferentes equilibrios entre precisión y requerimientos computacionales. Actualmente, los modelos disponibles son tres:

Modelo	Precisión Simplex	Requerimientos computacionales
fast (fast)	95.5%	
high accuracy (hac)	99.25%	7.5 veces lo requerido por fast
super accuracy (sup)	99.75%	8.5 veces lo requerido por hac

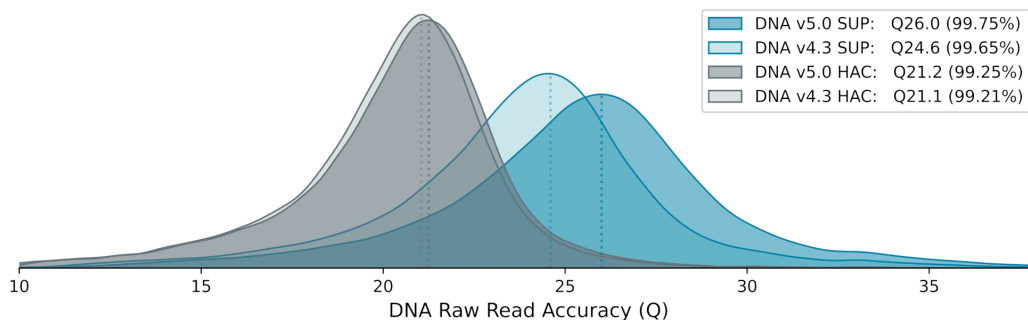


Figura 1. Precisión actual de los modelos de basecalling de Oxford Nanopore, usando el Kit V14 y celdas R10.4.1 de PromethION para secuenciación de genoma humano. Fuente: <https://nanoporetech.com/es/platform/accuracy>.

Usar el modelo **sup** es la opción recomendada si se busca obtener la máxima precisión posible. Sin embargo, este modelo tiene un costo computacional tan elevado que se vuelve impráctico de usar sin hardware dedicado (GPU).

2.2 | Uso básico de Dorado

[nanoporetech/dorado](https://github.com/nanoporetech/dorado)

Dorado posee múltiples subcomandos con distintas funcionalidades, que incluyen basecalling, demultiplexación, descarga de modelos y otras.

2.2.1 | Basecalling

Para basecalling, se utiliza el subcomando **dorado basecaller**. La forma más básica de ejecutar este comando necesita el modelo a utilizar en el basecalling y el directorio con los archivos POD5. Por ejemplo, si tenemos el directorio **pod5/** y queremos utilizar el modelo **sup**, se ejecuta el siguiente comando:

```
dorado basecaller sup pod5/ > reads.ubam
```

sh

Adicionalmente, existen múltiples opciones adicionales que se pueden visualizar en la ayuda del comando.

2.2.2 | Demultiplexación

Para demultiplexar los datos, se utiliza el subcomando `dorado demux`. Este comando requiere el archivo UBAM con los reads generado en el basecalling y el kit de barcoding. Por ejemplo, si tenemos el archivo `reads.ubam` y el kit SQK-NBD114-24:

```
dorado demux --output-dir basecalled_reads --kit-name SQK-NBD114-24 --emit-fastq reads.ubam
```

sh

En este ejemplo, `--output-dir` indica el directorio donde se guardarán los archivos demultiplexados, y `--emit-fastq` se utiliza para que los archivos generados estén en formato FASTQ (por defecto, se generan en BAM).

3 | Control y Filtros de Calidad

El control de calidad es una fase fundamental en el análisis de datos de secuenciación, especialmente en tecnologías de tercera generación debido a su mayor tasa de error. Entre las tareas más comunes de esta etapa se incluyen la eliminación de secuencias de baja calidad, la remoción de adaptadores, la filtración de secuencias cortas y la descontaminación.

La calidad de las lecturas se codifica en los archivos FASTQ generados por los dispositivos de secuenciación. En el caso de Oxford Nanopore, estos archivos, resultado del proceso de basecalling, contienen tanto la información de la secuencia como la calidad asociada a cada base. Un ejemplo de lectura en formato FASTQ se presenta a continuación:

```
@NB551068:9:HK5NLBGXX:1:11101:12901:1044 1:N:0:ATCACG
GATCGGAAGAGCACACGTCTGAACTCCAGTCACATCGTCTGAGGCTGCTGAACGCTCTCCGATCTTCTGCTTGA
+
IIIIHHHHHHHHGGGGGGGGGGFFFFEEEEEEEDDDDDCCCCCBBBBBBBBBAAAAAA@#####
```

Listado 1. Ejemplo de lectura en formato FASTQ. La primera línea contiene el identificador de la secuencia, la segunda línea la secuencia de nucleótidos, la tercera línea un carácter + y la cuarta línea la calidad de la secuencia.

Los valores de calidad Q-Score están codificados en formato ASCII, donde cada carácter corresponde a un valor numérico. A partir de este valor, es posible calcular la probabilidad de error asociada a la base utilizando la fórmula $P = 10^{-\frac{Q}{10}}$, donde Q representa el Q-Score. En la siguiente tabla se presentan las equivalencias entre los códigos ASCII y los Q-Scores:

Símbolo	Código ASCII	Q-Score	Precisión
!	33	0	0%
"	34	1	20%
#	35	2	36.9%
\$	36	3	50%
%	37	4	60%
&	38	5	68.4%
"	39	6	75%
(40	7	80%
)	41	8	84.1%
*	42	9	87.5%
+	43	10	90%
,	44	11	92.1%
-	45	12	93.7%
.	46	13	95%

Símbolo	Código ASCII	Q-Score	Precisión
/	47	14	96%
0	48	15	96.8%
1	49	16	97.5%
2	50	17	98%
3	51	18	98.4%
4	52	19	98.7%
5	53	20	99%
6	54	21	99.2%
7	55	22	99.4%
8	56	23	99.5%
9	57	24	99.6%
:	58	25	99.7%
;	59	26	99.8%
<	60	27	99.8%

Símbolo	Código ASCII	Q-Score	Precisión
=	61	28	99.84%
>	62	29	99.87%
?	63	30	99.9%
@	64	31	99.92%
A	65	32	99.94%
B	66	33	99.95%
C	67	34	99.96%
D	68	35	99.97%
E	69	36	99.98%
F	70	37	99.98%
G	71	38	99.99%
H	72	39	99.99%
I	73	40	99.99%

Tabla 1. Codificación de calidad en archivos FASTQ para Q-Score 1 a 40.

3.1 | FastQC

 [s-andrews/FastQC](#)

FastQC es una herramienta ampliamente utilizada para realizar el control de calidad de datos de secuenciación. Permite evaluar la calidad de las secuencias, identificar adaptadores, secuencias repetitivas y otros problemas comunes en los datos de secuenciación. Los resultados de las métricas evaluadas son presentadas en un reporte en formato HTML.

Para ejecutar FastQC se debe indicar el o los archivos de entrada como argumentos al comando. Otros parámetros relevantes son:

- `--outdir / -o`: Almacena los reportes generados en un directorio específico (debe estar creado).
- `--threads / -t`: Número de hilos a utilizar, lo que permite acelerar el proceso.
- `--memory`: Cantidad de memoria RAM (en MB) a utilizar por hilo de ejecución.

A continuación se presentan algunos ejemplos de ejecución:

```
# Ejecuta FastQC en el archivo 'sample1.fastq.gz'
fastqc sample1.fastq.gz
# Ejecuta FastQC en múltiples archivos, con 4 hilos, almacenando los reportes en 'reports'
fastqc -t 4 -o reports/ sample1.fastq.gz sample2.fastq.gz
```

 sh

3.2 | nanoq

 [esteinig/nanoq](#)  [10.21105/joss.02991](#)

Nanoq es una herramienta para realizar control y filtros de calidad específicamente desarrollada para trabajar con datos de Oxford Nanopore. Permite realizar tanto control de calidad como filtros de calidad en sí.

Para realizar control de calidad se debe indicar mediante el parámetro `-v / --verbose`. Existen tres niveles de salida:

- `-v`: Información básica (cantidad de secuencias, número total de bases, tamaño y calidad promedio).
- `-vv`: Similar a `-v`, pero incluye thresholds para la calidad y tamaño de las secuencias.
- `-vvv`: Similar a `-vv` pero incluye un ranking de las cinco secuencias con mejor calidad y mayor largo.

Al utilizar la opción de salida `-v` se debe indicar donde se va a imprimir la información, si en la salida estándar `-s` o en nuevo archivo `--report`.

El archivo de entrada debe indicarse mediante el parámetro `-i / --input`. A continuación se presenta un ejemplo de control de calidad, donde la información se almacena en el archivo `barcode01_stats.txt`:

```
nanoq -i barcode01.fastq.gz -vvs -r barcode01_stats.txt
```

 sh

Nanoq ofrece diferentes parámetros que permiten realizar filtros de calidad, los más relevantes se presentan a continuación:

- `--max-len / -m`: Tamaño máximo de las lecturas.
- `--min-len / -l`: Tamaño mínimo de las lecturas.
- `--max-qual / -w`: Calidad máxima para filtrar
- `--min-qual / -q`: Calidad mínima para filtrar.

Mediante el parámetro `-o --output` se indica el archivo que va a almacenar los datos filtrados. A continuación se presenta un ejemplo de filtros de calidad utilizando como calidad mínima 15 y tamaño entre 1.000 y 2.000 pares de bases:

```
nanoq --input barcode01.fastq.gz --min-qual 15 --min-len 1000 --max-len 2000 --output barcode01_filtered.fastq.gz
```

 sh

En caso de querer obtener las estadísticas del archivo filtrado se puede indicar que se almacene el resultado en un archivo de salida mediante el parámetro `-r o --report`.

```
nanoq --input barcode01.fastq.gz --min-qual 15 --min-len 1000 --max-len 2000 --output
barcode01_filtered.fastq.gz -vv --report barcode01_filtered_stats.txt -H
```

[sh](#)

3.3 | MultiQC

MultiQC es una herramienta que permite generar un único reporte de calidad a partir de reportes de múltiples herramientas y múltiples muestras. Soporta reportes de herramientas como FastQC, Nanoq, Fastp, cutadapt, NanoPlot entre otras. Una lista completa de las herramientas soportadas por MultiQC se pueden encontrar en la documentación oficial [MultiQC modules](#).

MultiQC necesita el directorio donde se encuentren los reportes generados por las herramientas, el cual puede indicarse inmediatamente luego del comando.

```
multiqc reports_directory
```

[sh](#)

Se puede indicar el nombre del archivo mediante el parámetro `--filename`, en caso de no indicarlo por defecto será `multiqc_report.html`.

```
multiqc --filename multiqc_raw.html reports/
```

[sh](#)

Se puede utilizar como shortcut el simbolo `.` para indicarle a multiqc que escanee el directorio actual y busque todos los reportes.

```
multiqc --filename multiqc_raw.html .
```

[sh](#)

4 | Asignación taxonómica

4.1 | NanoCLUST

4.2 | EMU

[10.1038/s41592-022-01520-4](#)

4.3 | EPI2ME wf-16S

[epi2me-labs/wf-16s](#)

4.3.1 | Mediante aplicación de escritorio

4.3.2 | Mediante línea de comando

Pipeline bioinformático desarrollado por EPI2ME-Labs. Cuenta con dos enfoques para la asignación taxonómica: Alineamiento de secuencias mediante Minimap2, o asignación taxonómica basada en k-mers mediante Kraken2. Permite utilizar tanto la base de datos de SILVA (versión 138) como la base de datos de Genbank de 16S y 18S.

En caso de utilizar Kraken2 se utiliza Bracken2 para la estimación de las abundancias.

El resultado es un archivo en formato tabular (TSV) que contiene la información de la asignación taxonómica por cada muestra, detallando la cantidad de lecturas asignadas a cada categoría taxonómica. Adicionalmente, genera un reporte en formato HTML que integra la información de asignación taxonómica, calidad de la secuenciación y métricas de diversidad por muestra.

Filtra las lecturas por tamaño (entre 800pb y 2000pb) pero por defecto no realiza filtros por calidad. Para considerar una asignación taxonómica exige un porcentaje de identidad de 95 % y una cobertura de 90%.

Por defecto el pipeline realiza la asignación taxonómica con la herramienta Minimap2 y la base de datos de 16S de Genbank. Para cambiar la herramienta de clasificación, utiliza el parámetro `--classifier`, eligiendo entre `kraken2` y `minimap2`. Para seleccionar una base de datos diferente, usa el parámetro `--database_set`, con alguna de las siguientes opciones: `ncbi_16s_18s`, `ncbi_16s_18s_28s_ITS` y `SILVA_138_1`.

4.3.3 | Ejemplo de uso

```
nextflow run epi2me-labs/wf-16s \
  --classifier kraken2 --database_set SILVA_138_1 \
  --sample_sheet samples.csv \
  --taxonomic_rank G --fastq data \
  --out_dir wf-16s_minimap_ncbi \
  -profile singularity -resume
```

sh

El pipeline requiere un archivo de muestras en formato CSV que contenga la información de las muestras y los barcodes asociados.

4.3.3.1 | Estructura del directorio

```
barcode,sample_id,alias
barcode01,1M,1M
barcode06,4H,4H
barcode08,5H,5H
barcode10,6H,6H
barcode11,6M,6M
barcode12,7H,7H
barcode13,7M,7M
```

4.3.3.2 | Estructura del archivo de muestras

Este pipeline esta pensando para ser ejecutado luego de la etapa de basecalling, por lo que se espera que los archivos FASTQ estén en la carpeta correspondiente de cada barcode. Cada carpeta de los barcodes a analizar debe encontrarse dentro de la carpeta que se indicara con el parámetro `--fastq`. La estructura del directorio debe ser la siguiente:

```
— input_directory
  |— barcode01
  |   |— reads0.fastq
  |— barcode02
  |   |— reads0.fastq
  |— barcode03
  |   |— reads0.fastq
```

4.4 | Eliminación de especies poco abundantes y normalización por muestra

Una práctica común en el análisis de datos de microbioma es la eliminación de especies poco abundantes, ya que estas pueden representar ruido en los análisis afectando la interpretación de los resultados.

Para esto, se puede establecer un umbral de abundancia mínima, eliminando las especies que no cumplen con este criterio. En este caso, eliminaremos todos los taxones que no cumplan con un umbral de abundancia del 0.1% en alguna muestra. También eliminaremos las lecturas que no lograron clasificarse, esto lo haremos mediante R:

```
count_data_filtered <- count_data %>%
  column_to_rownames("tax") %>%
  filter_all(any_vars(. / sum(.) > 0.0001)) %>%
  select(-total, -starts_with("Unclassified")) %>%
```

R

4.5 | Normalización por muestra

Muchas veces la cantidad de lecturas varía significativamente entre las muestras, y eso hace que los conteos obtenidos en la asignación taxonómica varíen de manera notoria. Es por esto, que se hace necesario normalizar los datos para poder comparar las diferentes muestras. La normalización de los datos se realiza para corregir el sesgo en la abundancia de las es-

pecies debido a la cantidad de secuencias generadas por muestra. El objetivo de la normalización es tener el mismo tamaño de librería para todas las muestras.

Esto se puede hacer mediante varias metodologías: normalización por submuestreo utilizando un tamaño mínimo; escalamiento donde se divide cada abundancia por un factor para eliminar el sesgo de muestreo desigual, etc

Utilizaremos una normalización por XXX mediante la función `decostat` de la librería `vegan` en R.

```
data_normalized <- decostand(count_data_filtered, method = "total")
```

R

<https://carpentries-lab.github.io/metagenomics-analysis/08-Diversity-tackled-with-R/index.html>

`avgdist` -> no todas las muestras tienen la misma cantidad de seqs -> sampling

5 | Curvas de rarefacción

Las curvas de rarefacción permiten evaluar la riqueza de especies dentro de una comunidad en función del número de secuencias obtenidas. Nos permiten determinar si el número de secuencias obtenidas es suficiente para capturar la diversidad de la comunidad.

Para ello, se realizan muestreos aleatorios y se visualiza el número de especies observadas a medida que aumenta el número de secuencias.

Utilizaremos el paquete `iNEXT` en R para realizar las curvas de rarefacción.

```
D_abund <- iNEXT(df, datatype = 'incidence_filtered')
plot(D_abund)
```

R

6 | Índices de diversidad alfa

Las métricas de diversidad alfa se utilizan para medir la diversidad dentro de una muestra o ecosistema, es decir, qué hay y cuánto hay en términos de especies.

Las métricas mas comunes de diversidad alfa son:

- Riqueza: Número de especies observadas en una muestra.
- Chao1: Estima la riqueza total (número de especies no observadas en una muestra).
- Shannon: Mide la diversidad de especies en una muestra, considerando la abundancia de las especies.

Para esto, utilizaremos el paquete `vegan` en R. Calcularemos la riqueza, equidad, índice de Shannon y Chao1.

```
data_richness <- estimateR(data_otu)
data_evenness <- diversity(data_otu) / log(specnumber(data_otu))
data_shannon <- diversity(data_otu, index = "shannon")
```

R

Podemos utilizar diferentes test estadísticos para comprobar si existen diferencias significativas entre los grupos: pruebas no paramétricas como el test de Kruskal-Wallis o el test de Mann-Whitney o pruebas paramétricas como t-test y ANOVA. Antes de utilizar pruebas paramétricas se debe comprobar la normalidad y homocedasticidad de los datos.

7 | Índices de diversidad beta

La diversidad beta nos permite representar las diferencias de diversidad entre muestras o ecosistemas, es decir, que tan similares o diferentes son las comunidades microbianas.

Estas métricas de distancia varían entre cero y uno. Las más usadas son las siguientes:

- Bray-Curtis: Mide la disimilitud entre muestras. Se basa en la abundancia de los taxones en las muestra.

- Jaccard: Mide disimilitud. Se basa en la presencia/ausencia de los taxones en las muestras, sin incluir información de la abundancia.
- Unifrac: Mide la distancia filogenética entre comunidades, considerando la presencia/ausencia, abundancias y evolución filogenética. Unweighted UniFrac considera solo la presencia o ausencia de otus (sin considerar abundancia), Weighted UniFrac considera las abundancias

Para calcular la diversidad beta necesitamos el archivo de abundancias generado en el paso anterior y un archivo de meta-data.

sample	sex	Area	latitude	long	deep
1M	Male	48.2	60° 25,0	46° 41.8	60-80
4H	Female	48.2	60° 33,1	46° 02.3	120-150
5H	Female	48.2	60° 30,0	46° 36.4	30-30
6H	Female	48.2	60° 30,1	46° 42.7	30-33
6M	Male	48.2	60° 30,1	46° 42.7	30-33
7H	Female	48.1	62° 37,0	55° 26.7	30-29
7M	Male	48.1	62° 37,0	55° 26.7	30-29

CSV

Para calcular la diversidad beta utilizaremos las matrices de disimilitud de Bray-curtis y Jaccard y las proyectaremos en un espacio bidimensional mediante una PCoA.

Una PCoA ((Principal Coordinate Analysis) es una técnica de ordenación que permite reducir la dimensionalidad de los datos y visualizar la diversidad beta en un espacio de menor dimensión. Algunas funciones utiles a utilizar son:

- `vegdist`: Permite calcular la matri una matriz de disimilitud entre muestras. Se pueden utilizar diferentes métricas de distancia, como Bray-curtis, Jacard, Euclidean, entre otras.
- `cmdscale`: Realiza un análisis de coordenadas principales (PCoA) a partir de una matriz de disimilitud.

```
bray_dist <- vegdist(data_otu, method = "bray")
pcoa_res <- cmdscale(bray_dist, eig = TRUE)
```

R

HACER ANALISIS ESTADISTICOS