

## 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
```

2 | Basecalling

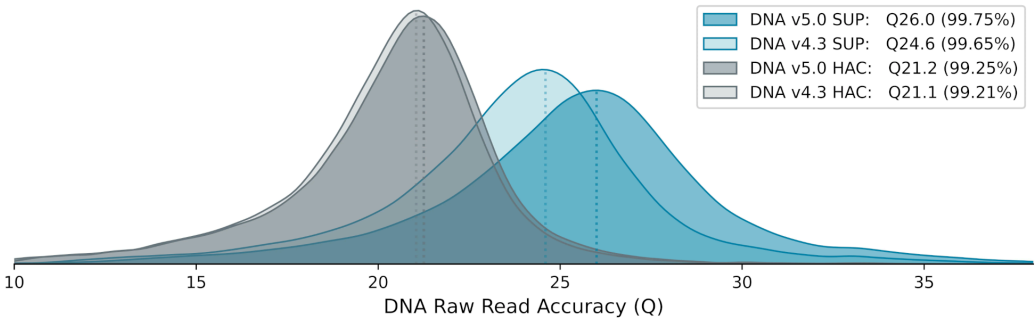
Basecalling es el proceso por el cual se convierte la señal eléctrica captada por los dispositivos de Oxford Nanopore en secuencias de nucleótidos. Esta conversión requiere de algoritmos computacionales avanzados, como las redes neuronales LSTM y los transformers utilizados por las opciones de basecalling 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
```

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`, debemos utilizar el siguiente comando:

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

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 el `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, eliminación de adaptadores, filtro de secuencias cortas y descontaminación.

La calidad de las lecturas se codifica en los archivos FASTQ generados por los dispositivos de secuenciación, y en el caso de Oxford Nanopore son 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
GATCGGAAGAGCACACGTCTGAACTCCAGTCACATCGTCTGAGGCTGCTGAACCGCTCTCCGATCTTCTGCTTGA
+
IIIIHHHHHHHHGGGGGGGGGGFFFFEEEEEEEEDDDDCCCCCBBBBBBBBBAAAAAAAA@#####
```

**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	Símbolo	Código ASCII	Q-Score	Precisión	Símbolo	Código ASCII	Q-Score	Precisión
!	33	0	0%	/	47	14	96%	=	61	28	99.84%
"	34	1	20%	0	48	15	96.8%	>	62	29	99.87%
#	35	2	36.9%	1	49	16	97.5%	?	63	30	99.9%
\$	36	3	50%	2	50	17	98%	@	64	31	99.92%
%	37	4	60%	3	51	18	98.4%	A	65	32	99.94%
&	38	5	68.4%	4	52	19	98.7%	B	66	33	99.95%
"	39	6	75%	5	53	20	99%	C	67	34	99.96%
(	40	7	80%	6	54	21	99.2%	D	68	35	99.97%
)	41	8	84.1%	7	55	22	99.4%	E	69	36	99.98%
*	42	9	87.5%	8	56	23	99.5%	F	70	37	99.98%
+	43	10	90%	9	57	24	99.6%	G	71	38	99.99%
,	44	11	92.1%	:	58	25	99.7%	H	72	39	99.99%
-	45	12	93.7%	;	59	26	99.8%	I	73	40	99.99%
.	46	13	95%	<	60	27	99.8%				

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. 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 diseñada para trabajar con datos de Oxford Nanopore.

#### Control de calidad

Para obtener métricas de calidad se utiliza el parámetro `-s / --stats`. Esto genera una tabla con cantidad de secuencias, número total de bases, tamaño y calidad promedio. Se puede usar `--report` para guardar la información generada en un archivo. Existen opciones que permiten obtener información adicional, las cuales son:

- `-v` : Información básica en formato extendido.
- `-vv` : Similar a `-v`, pero incluye una compartimentación de 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.

El archivo de entrada debe indicarse mediante el parámetro `-i / --input`.

Ejemplo de uso para control de calidad:

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

[sh](#)

#### Filtros de calidad

Al usar nanoq para realizar filtros de calidad, los parámetros más relevantes son `--min-len / -l`, `--max-len / -m`, `--min-qual / -q` y `--max-qual / -w`. Mediante el parámetro `--output / -o` se indica el archivo que va a almacenar los datos filtrados.

En el siguiente ejemplo se filtra con calidad mínima 15 y tamaño entre 1.000 y 2.000 pares de bases:

```
nanoq -i barcode01.fastq.gz -q 15 -l 1000 -m 2000 -o barcode01_filtered.fastq.gz
```

[sh](#)

Al usar filtros de calidad, se puede utilizar la opción `--report` para generar el reporte de estadísticas del archivo filtrado:

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

[sh](#)

### 3.3 | MultiQC

#### MultiQC/MultiQC

MultiQC permite presentar resultados de reportes de múltiples herramientas y múltiples muestras en un único reporte HTML. Soporta un gran cantidad de herramientas bioinformáticas como FastQC, nanoq, fastp, cutadapt y NanoPlot. La lista de herramientas soportadas se puede encontrar en la sección [MultiQC modules](#) de la documentación.

MultiQC requiere como argumento el directorio donde se encuentren los reportes generados por las herramientas. Si se indica `.`, MultiQC buscará los reportes en el directorio actual. Ejemplo:

```
multiqc reports/
```

[sh](#)

Por defecto el nombre del reporte será `multiqc_report.html`, pero se puede indicar con `--filename` :

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

[sh](#)

## 4 | Asignación taxonómica

### 4.1 | NanoCLUST

#### genomics/ITER/NanoCLUST

NanoCLUST es un flujo de trabajo desarrollado en Nextflow para la clasificación de amplicones del gen 16S obtenidos mediante secuenciación por Nanopore. Utiliza un enfoque de clustering no supervisado seguido por una proyección UMAP y una corrección de errores de cada cluster previo a la asignación taxonómica. Para la asignación taxonómica utiliza BLAST y la base de datos de 16S de Genbank.

Para utilizar esta herramienta se debe contar con una versión de nextflow menor o igual a 22.04 y se debe tener la base de datos de 16S de Genbank (instrucciones de descarga en el repositorio de NanoCLUST).

```
nextflow run main.nf \
  -profile docker \
  --reads 'sample.fastq' \
  --db "db/16S_ribosomal_RNA" \
  --tax "db/taxdb/"
```

[sh](#)

### 4.2 | EMU

#### 10.1038/s41592-022-01520-4

EMU es una herramienta diseñada para mejorar la precisión de la asignación taxonómica mediante una corrección de errores utilizando un enfoque basado en algoritmos de maximización de expectativas y alineamiento de las secuencias corregidas mediante la herramienta Minimap2. Ofrece compatibilidad con diversas bases de datos, como Genbank, RDP y Silva (v.138). Además, en el caso de realizar análisis de la región ITS, permite integrar las bases de datos de UNITE que se especializa en taxonomía de hongos y eucariotas.

Para utilizar EMU se debe descargar su base de datos e instalar las dependencias necesarias (instrucciones de instalación en el repositorio de EMU).

```
emu abundance example/full_length.fa
```

[sh](#)

### 4.3 | EPI2ME wf-16S

#### epi2me-labs/wf-16s

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 Kra-

ken2 (y bracken2 para la corrección). Permite utilizar tanto la base de datos de SILVA (versión 138) como la base de datos de Genbank de 16S y 18S.

Por defecto filtra las lecturas por tamaño (entre 800pb y 2000pb) y no realiza filtros por calidad. Para considerar una asignación taxonómica exige un porcentaje de identidad de 95 % y una cobertura de 90%. Todos estos parámetros pueden ser modificados por el usuarios.

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.

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`.

Puede ejecutarse mediante la interfaz de línea de comandos o mediante una aplicación de escritorio.

#### 4.3.1 | Mediante aplicación de escritorio

#### 4.3.2 | Mediante línea de comando

Para ejecutarlo mediante línea de comando solamente se debe contar con Nextflow ya que las bases de datos se van a descargar automáticamente: 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](#)

##### 4.3.2.1 | Estructura del archivo de muestras

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

Podemos generar el archivo mediante un editor de textos:

```
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
```

Al visualizar el archivo en un procesador de hojas de cálculo, se vería de la siguiente manera:

barcode	sample_id	alias
barcode01	1M	1M
barcode06	4H	4H
barcode08	5H	5H
barcode10	6H	6H
barcode11	6M	6M

barcode	sample_id	alías
barcode12	7H	7H
barcode13	7M	7M

#### 4.3.2.2 | Estructura del directorio

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 especies 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

## 5 | Análisis de Diversidad

vegandevs/vegan

### 5.1 | Rarefacción

La rarefacción es una técnica que permite evaluar la riqueza de especies dentro de una comunidad en función del número de secuencias obtenidas. Es útil para determinar si las secuencias obtenidas son suficientes para capturar la diversidad de



la comunidad. Para ello, se utilizan muestreos aleatorios y se visualiza en una curva el número de especies observadas a medida que aumenta el número de secuencias.

Para construir las curvas de rarefacción, podemos utilizar la función `rarecurve()` del paquete `vegan` en R. El parámetro `sample` agrega una línea vertical al gráfico, útil para comparar la riqueza observada con la muestra de menor número de secuencias.

```
library(vegan)

min_reads <- min(rowSums(count_data_t)) # número mínimo de secuencias en una muestra

rarecurve(count_data_t, step = 20, sample = min_reads)
```

## 5.2 | Diversidad Alfa

Las métricas de diversidad alfa se emplean para medir la diversidad dentro de una muestra o ecosistema, es decir, la cantidad de especies y/o su abundancia relativa.

Las métricas más comunes de diversidad alfa incluyen:

- **Riqueza:** Número de especies observadas en una muestra.
- **Equidad:** Grado de uniformidad en la distribución de las abundancias de las especies en una muestra.  
*Interpretación:* Valores cercanos a uno indican que todas las especies tienen abundancias similares, mientras que valores cercanos a cero sugieren que una o pocas especies dominan la comunidad.
- **Shannon:** Índice que mide la diversidad considerando tanto la riqueza como la equidad de las especies en una muestra.  
*Interpretación:* Valores altos indican mayor diversidad y equidad, mientras que valores bajos pueden señalar una baja equidad o riqueza.
- **Chao1:** Estimación de la riqueza total, incluyendo especies no observadas.  
*Interpretación:* Si el valor de Chao1 es significativamente mayor que la riqueza observada, indica que el muestreo fue insuficiente para detectar todas las especies presentes.

Para calcular estas métricas en R, también usaremos el paquete `vegan`.

```
library(vegan)

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

### *i* Info

La diversidad alfa habitualmente se calcula sobre los datos sin procesar y sin normalizar, ya que se busca evaluar la diversidad por muestra y no comparar muestras entre sí.

### Pruebas estadísticas

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.

## 5.3 | Diversidad Beta

La diversidad beta se utiliza para evaluar las diferencias de diversidad entre muestras o ecosistemas, es decir, qué tan similares o diferentes son las comunidades microbianas entre sí. Estas métricas de distancia varían entre cero y uno, y las más comunes son:

- **Bray-Curtis:** Calcula la disimilitud entre muestras basándose en la abundancia de los taxones presentes en ellas.

- **Jaccard**: Calcula la disimilitud tomando en cuenta solo la presencia o ausencia de los taxones, sin considerar la abundancia.
- **Unifrac**: Calcula la distancia filogenética entre comunidades. Se puede calcular en dos formas:
  - **Unweighted UniFrac**: Considera solo la presencia o ausencia de OTUs (sin abundancia).
  - **Weighted UniFrac**: Incluye tanto la abundancia como la evolución filogenética de los OTUs.

Para calcular la diversidad beta, necesitamos el archivo de abundancias generado en el paso anterior y un archivo de metadatos. A continuación, un ejemplo de archivo de metadatos:

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 visualizar la diversidad beta, se utilizan matrices de disimilitud, como Bray-Curtis o Jaccard, que luego se proyectan en un espacio bidimensional utilizando un Análisis de Coordenadas Principales (PCoA). Esta técnica reduce la dimensionalidad de los datos, facilitando la visualización de las relaciones entre muestras.

Algunas funciones útiles para este análisis en R son:

- `vegdist()` : Calcula una matriz de disimilitud entre las muestras, permitiendo elegir entre métricas como Bray-Curtis, Jaccard, Euclidiana, entre otras.
- `cmdscale()` : Realiza un análisis de coordenadas principales (PCoA) a partir de la matriz de disimilitud.

```
library(vegan)
```

R

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



### Consejo

Al realizar un análisis de coordenadas principales (PCoA), es importante tener en cuenta que los valores propios (eigenvalues) representan la varianza explicada por cada componente. Un eigenvalue alto indica que ese eje captura una mayor cantidad de la varianza total de los datos, lo que permite una mejor interpretación de la estructura de las muestras.

## HACER ANALISIS ESTADISTICOS