Contenido

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:

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

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 pueden requerir argumentos. Finalmente, algunos comandos poseen `[subcomandos]`, que permiten realizar tareas específicas.

> 🔥 **Ayuda en comandos**
>
> 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

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

`pwd` – Imprimir el directorio actual

```sh
pwd
```

`cd` – Cambiar de directorio

```sh
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
```

`cp` – Copiar archivos o directorios

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

`mv` – Mover o renombrar archivos

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

`rm` – Eliminar archivos o directorios

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

`mkdir` – Crear directorios

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

`head` / `tail` / `less` – Visualizar el contenido de archivos

```sh
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
```

Redirección de salida

```sh
head -n 100 sequences.fa > subset.fa    # Guarda la salida del comando ejecutado en un archivo
```

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
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)
YAML	`.yaml` `.yml`	Datos estructurados en formato YAML

Archivos comprimidos

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

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

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", entornos de trabajo aislados y reproducibles que mantienen la instalación de paquetes separada de otros ambientes y del sistema operativo. Los ambientes pueden activarse y desactivarse según sea necesario. Los paquetes están disponibles en "canales", destacando conda-forge para paquetes de Python y R y bioconda para herramientas bioinformáticas.

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

Listar, crear y eliminar ambientes

```sh
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'
```

Activar y desactivar ambientes

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

Gestionar paquetes

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

Exportar e importar ambientes

```sh
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 dipositivos 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: `fast`, `hac`, y `sup`. Estos modelos se actualizan continuamente con nuevas versiones.

Modelo	Precisión simplex	Requerimientos computacionales
fast (`fast`)	95.50%	
high accuracy (`hac`)	99.25%	7.5 veces lo requerido por `fast`
super accuracy (`sup`)	99.75%	8.5 veces lo requerido por `hac`

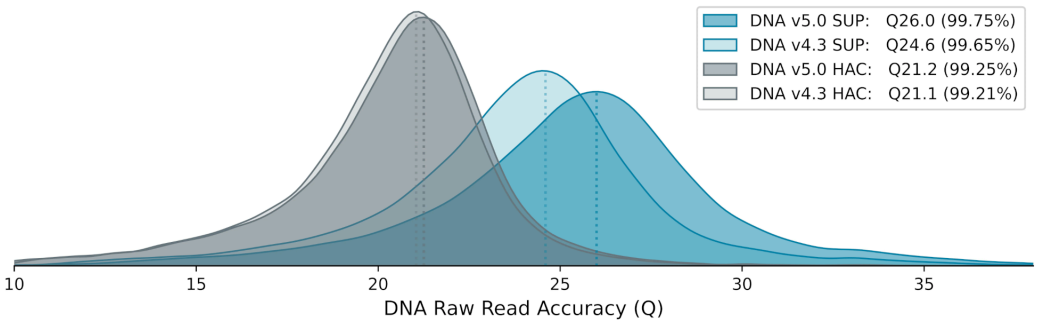


Figura 1. Precisión actual de los modelos de basecalling de Oxford Nanopore, con el Kit V14 y celdas R10.4.1 de PromethION en secuenciación de genoma humano. *Fuente: https://nanoporetech.com/es/platform/accuracy*.

El modelo `sup` ofrece la máxima precisión, pero sus altos requisitos computacionales lo hacen impráctico de utilizar sin hardware especializado (GPU de gama alta).

2.2 | Uso de Dorado

nanoporetech/dorado

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

2.2.1 | Basecalling

Para basecalling, se utiliza el subcomando `dorado basecaller`. Par ejecutar este comando se requiere como mínimo el modelo a utilizar y el directorio con los archivos POD5. Por ejemplo, si tenemos el directorio `pod5/` y queremos utilizar el modelo `sup` en su versión 5.0.0, debemos utilizar el siguiente comando:

```sh
dorado basecaller sup@5.0.0 pod5/ > reads.ubam
```

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:

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

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
GATCGGAAGAGCACACGTCTGAACTCCAGTCACATCGTCTGAGGCTGCTGAACCGCTCTTCCGATCTTCTGCTTGAAA
+
IIIIHHHHHHHHHHGGGGGGGGGGFFFFFEEEEEEEEDDDDDCCCCCCCBBBBBBBBBAAAAAAAA@@@@@@######
```

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, se puede calcular la probabilidad de error asociada a la base con la fórmula $P = 10^{\frac{-Q}{10}}$, donde Q es el Q-Score. En la Tabla 1 se presentan las equivalencias entre los códigos ASCII y los Q-Scores.

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.

Símbolo	Q-Score	Precisión	Símbolo	Q-Score	Precisión	Símbolo	Q-Score	Precisión
!	0	0%	/	14	96.0%	=	28	99.84%
"	1	20.0%	0	15	96.8%	>	29	99.87%
#	2	36.9%	1	16	97.5%	?	30	99.90%
$	3	50.0%	2	17	98.0%	@	31	99.92%
%	4	60.0%	3	18	98.4%	A	32	99.94%
&	5	68.4%	4	19	98.7%	B	33	99.95%
‘	6	75.0%	5	20	99.0%	C	34	99.96%
(	7	80.0%	6	21	99.2%	D	35	99.97%
)	8	84.1%	7	22	99.4%	E	36	99.98%
*	9	87.5%	8	23	99.5%	F	37	99.98%
+	10	90.0%	9	24	99.6%	G	38	99.99%
,	11	92.1%	:	25	99.7%	H	39	99.99%
-	12	93.7%	;	26	99.8%	I	40	99.99%
.	13	95.0%	<	27	99.8%			

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

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:

```sh
# 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
```

3.2 | nanoq

esteinig/nanoq 10.21105/joss.02991

Nanoq es una herramienta para realizar control y filtros de calidad diseñada especialmente 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 parámtero `-i / --input`.

Ejemplo de uso para control de calidad:

```sh
nanoq -i barcode01.fastq.gz -svv -r bacorde01_stats.txt
```

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:

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

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

```sh
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
```

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 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:

```sh
multiqc reports/
```

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

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

4 | Asignación Taxonómica

4.1 | NanoCLUST

genomicsITER/NanoCLUST 10.1093/bioinformatics/btaa900

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 descargar la base de datos de 16S de Genbank (instrucciones disponibles en el repositorio de NanoCLUST).

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

4.2 | EMU

treangenlab/emu 10.1038/s41592-022-01520-4

EMU es una herramienta diseñada para mejorar la precisión de la asignación taxonómica mediante un enfoque de corrección de errores basado en algoritmos de maximización de expectativas y alineamiento de las secuencias corregidas mediante la herramienta Minimap2. EMU es compatible con diversas bases de datos, como Genbank, RDP y Silva (v.138), y también permite la integración de la base de datos UNITE para el análisis de la región ITS, especializada en la taxonomía de hongos y eucariotas.

Para usar EMU, es necesario descargar su base de datos e instalar las dependencias (instrucciones en el repositorio de EMU).

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

4.3 | EPI2ME wf-16s

epi2me-labs/wf-16s

EPI2ME wf-16s es un pipeline bioinformático desarrollado por Oxford Nanopore para la asignación taxonómica de secuencias 16S. Ofrece dos alternativas para la clasificación taxonómica: alineamiento de secuencias mediante Minimap2 o asignación basada en k-mers utilizando Kraken2 y Bracken2. El pipeline permite utilizar bases de datos como SILVA (v.138) y las de 16S y 18S de Genbank.

El resultado incluye un archivo en formato tabular (TSV) con la asignación taxonómica por muestra, especificando el número de lecturas asignadas a cada taxón. También se genera un reporte en formato HTML que integra la información sobre la calidad de la secuenciación, asignación taxonómica y métricas de diversidad.

Este pipeline puede ejecutarse mediante la línea de comandos con Nextflow o mediante la aplicación de escritorio EPI2ME (https://labs.epi2me.io/).

4.3.1 | Requisitos

Para ejecutar el pipeline, es necesario tener instalados Nextflow y Docker (o como alternativa, Apptainer). El pipeline descarga automáticamente las bases de datos necesarias y todas las dependencias para su ejecución.

> **? ¿Qué es Nextflow?**
>
> Nextflow es un framework para diseñar y ejecutar pipelines bioinformáticos. Utiliza un lenguaje declarativo para definir flujos de trabajo, que pueden ejecutarse de manera reproducible en entornos locales, en la nube o en clústeres de alto rendimiento.

Estructura del archivo de muestras

El pipeline requiere un archivo CSV que contenga la información de las muestras y los códigos de barras (barcodes) asociados. A continuación se muestra un ejemplo de este archivo:

```
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
```

Contenido del archivo.

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

Representación del archivo como una tabla.

Estructura de los datos de entrada

El pipeline está diseñado para ejecutarse después de la etapa de basecalling. Se espera que los archivos FASTQ estén organizados por barcode en directorios correspondientes, como se muestra a continuación:

```
input_directory
├── barcode01
│   ├── reads0.fastq
│   └── reads1.fastq
├── barcode02
│   └── reads0.fastq
└── barcode03
    └── reads0.fastq
```

4.3.2 | Configuración

Este pipeline incluye múltiples parámetros configurables mediante parámetros de línea de comandos si se ejecuta mediante Nextflow, o mediante opciones gráficas si se utiliza la aplicación de escritorio EPI2ME.

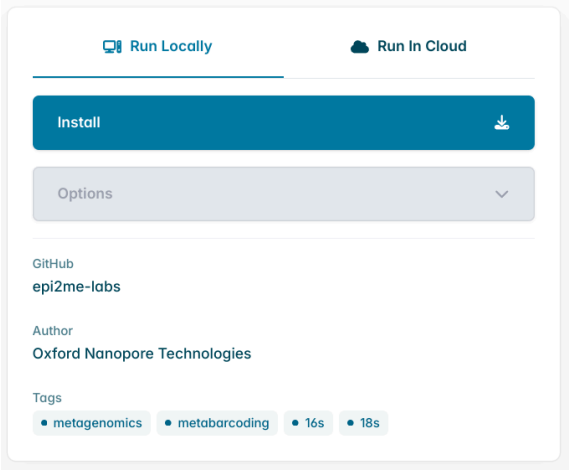
Por defecto, las lecturas son filtradas por tamaño entre 800 pb y 2000 pb y no se aplican filtros de calidad. El porcentaje de identidad requerido para la asignación taxonómica es del 95% y la cobertura del 90%.

El pipeline utiliza Minimap2 con la base de datos de 16S de Genbank de forma predeterminada. Para el método de clasificación se puede escoger entre `kraken2` y `minimap2`, mientras que para la base de datos las opciones son `ncbi_16s_18s`, `ncbi_16s_18s_28s_ITS` y `SILVA_138_1`.

4.3.3 | Ejecución mediante aplicación de escritorio

La aplicación EPI2ME permite ejecutar diversos pipelines de análisis de datos de secuenciación de Oxford Nanopore. En este caso, debemos seleccionar el pipeline 16s e iniciar su instalación. La aplicación verificará si el equipo cumple con los requisitos computacionales y descargará el pipeline en un directorio local.

Selección del pipeline 16s en la aplicación EPI2ME.



Cuadro de instalación del pipeline.

Una vez instalado, el botón `Install` cambiará a `Launch`. Al hacer clic en `Launch`, se abrirán las opciones de configuración del pipeline, donde seleccionaremos el directorio con los archivos de entrada, la base de datos, el método de clasificación, la planilla de muestras y otros parámetros (Figura 3).

Al finalizar la ejecución, podremos visualizar el reporte generado en la misma aplicación y acceder al directorio con los resultados.

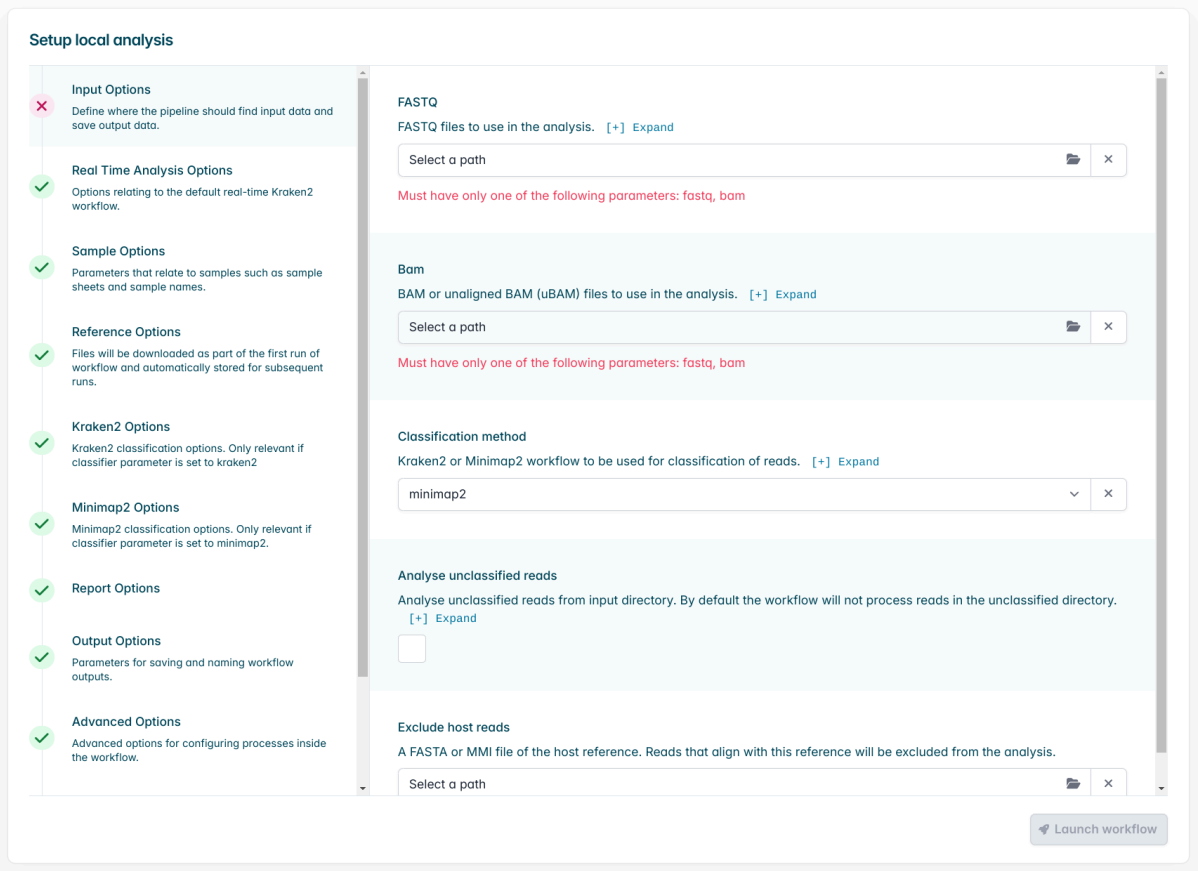


Figura 3. Configuración del pipeline EPI2ME wf-16s.

4.3.4 | Ejecución mediante línea de comandos

Para ejecutar el pipeline mediante la línea de comandos, debe especificarse como mínimo los datos de entrada con el parámetro `--fastq`. Para configurar el clasificador debe utilizarse el parámetro `--classifier`, y para configurar la base de datos se utiliza el parámetro `--database_set`.

Ejemplo de uso:

```sh
nextflow run epi2me-labs/wf-16s \
    --fastq data \
    --sample_sheet samples.csv \
    --out_dir wf-16s_kraken2_silva \
    --classifier kraken2 \
    --database_set SILVA_138_1 \
    --taxonomic_rank G \
    -profile docker
```

5 | Post Procesamiento de Asignación Taxonómica

5.1 | Eliminación de especies poco abundantes

En el análisis de datos de microbioma, es común eliminar especies poco abundantes, ya que pueden introducir ruido y afectar la interpretación de los resultados. Se puede establecer un umbral de abundancia mínima y eliminar los taxones que no lo cumplan.

Por ejemplo, podemos eliminar todos los taxones cuya abundancia sea inferior al 0.1% en todas las muestras. Adicionalmente, podemos excluir las lecturas no clasificadas. Todo esto podemos lograrlo con el siguiente código en R:

```r
library(tidyverse)

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

5.2 | Normalización por muestra

Es común que la cantidad de lecturas varíe significativamente entre las muestras, lo que provoca que los conteos obtenidos en la asignación taxonómica presenten grandes variaciones. Para comparar las muestras de manera justa, es necesario normalizar los datos y corregir el sesgo en la abundancia de especies debido a estas diferencias.

Existen diversas metodologías para normalizar: el submuestreo utilizando un tamaño mínimo, el escalamiento dividiendo cada abundancia por un factor para compensar el sesgo de muestreo, entre otras.

Por ejemplo, podemos normalizar mediante Total Sum Scaling (TSS), usando la función `decostand` de la librería vegan en R.

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

6 | Análisis de Diversidad

vegandevs/vegan

6.1 | 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.

```r
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)
```

6.2 | 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.

```r
library(vegan)

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

> **Normalización y diversidad alfa**
>
> 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 estádisticos 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 parámetricas como t-test y ANOVA. Antes de utilizar pruebas parámetricas se debe comprobar la normalidad y hococedasticidad de los datos.

6.3 | 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:

CSV

```
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
```

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.

R

```r
library(vegan)

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

> 🔥 **Varianza explicada en un PCoA**
>
> 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.