An introduction to bioinformatics: advances in biology and health sciences - Tutorial

Mario Hernández Guzmán & Valentín Pérez Hernández

2022-05-28

1. Descargar las secuencias: archivos fastq

Las secuencias son parte del estudio realizado por Hernández-Guzmán et al. (2022). De éste, se tomó una librería bajo el ID: SRA SRR9162917. El experimento completo puede tomarse del BioProject PRJNA545497.

1.1. Descargar las secuencias desde GitHub

Las secuencias, archivos para el análisis y los archivos de salidad de todos los comandos en este tutorial puede ser descargados del repositorio de GitHub:

```
## Moverse a su directorio raiz, i.e., $HOME/
$ cd

## Clonar el repositorio
$ git clone https://github.com/MaryoHg/portillo_etal

## Opcionalmente, las sencuencias pueden descargarlas directamente desde:
> https://github.com/MaryoHg/portillo_etal/tree/main/raw_input
```

1.2. Descargar las secuencias desde NCBI

Para descargar las secuencias desde la base de datos pública del *National Center for Biotechnology Information* (NCBI por sus siglas en inglés), necesitamos **instalar SRA Toolkit**. Para ello podemos guiarnos de: https://www.ncbi.nlm.nih.gov/sra/docs/sradownload/.

```
## Descargamos las secuencias: R1 y R2 fastq, por su SRA ID
$ prefetch SRR9162917

## Separamos el archivo SRA en dos archivos: R1 (forward) y R2 (reverse reads)
$ fasterq-dump --split-files SRR11180057.sra
```

2. Validar metadatos:

La validación de los metadatos se realiza para evitar problemas posteriores en el análisis. En pocas palabras, se corrobora que i. los datos estén contenidos en texto plano (archivo txt ó tsv), ii. las variables categóricas esten en orden, iii. ningún ID de muestra se repita (#SampleID column), iv. la columna "Description" se encuentre presente en el archivo (obligagorio) como última columna, v. ningún barcode se repita, vi. no existan carácteres especiales en los metadatos.

Los errores que encuentra el comando pueden ser vistos y analizados empleando el archivo HTML generado como salida del mismo script:

```
$ validate_mapping_file.py -m mappig_file.txt -o validatemap/

## Opcionalmente, puede descargar los archivos de salida de este script desde el repositorio, en:
> https://github.com/MaryoHg/portillo_etal/tree/main/02%20validatemap>.
```

3. Extraer los barcodes

Las lecturas contienen una secuencia nucleotídica al inicio de cada producto de secuenciación; esta secuecia de 8 pares de bases (bp) sirve para hacer el proceso de *demultiplexing* (ver punto 5). El *barcode* sirve, esencialmente, para identificar el origen de una secuencia y relacionarla con sus metadatos (fuente u origen).

Para correr el script necesitamos conocer la longitud del *barcode*, y su tipo. En este estudio, el tipo de *barcode* empleado fue **paired_end**, que se indica como *-c barcode_paired_end* en el comando.

```
## correr comando para extraer los barcodes de los archivos iniciales o secuencia crudas
$ extract_barcodes.py -f forward.fastq -r reverse.fastq -c barcode_paired_end -l 8 -L 8  -o barcodes/

# opciones usadas:
-f/--fastq1: ruta del archivo R1.fastq (forward reads);
-r/--fastq2: ruta del archivo R2.fastq (reverse reads);
-c/--input_type: tipo de secuencia de los barcodes, en el estudio original fue: barcode_paired_end;
-l/--bc1_len: longitud que tiene el barcode en pares de bases del archivo R1.fastq, por ejemplo: 8;
-L/--bc2_len: longitud del barcode en pares de bases del archivo R2.fastq

## Opcionalmente se pueden descargar los archivos en (deberá descomprimir los archivos)
> https://github.com/MaryoHg/portillo_etal/tree/main/03%20barcodes
```

4. Ensamblar las lecturas R1 y R2: *Merging overlap sequences*

El ensamble de la secuencias se realiza con la herramienta fastq-join (opción por *default*) empleando el comando *join_paired_ends.py* de QIIME. Este comando ensambla (o une) las lecturas R1 y R2 empleando el "sobrelape" (*overlap read*) de 100 pb que existen en las regiones V3 y V4 del gen 16S rDNA. Los archivos de entrada de este comando son los archivos de salida del paso anterior, i.e., archivos en la carpeta <./barcodes/>

NOTA: Existen otras opciones de herramientas para unir o ensamblar estas secuencias, por ejemplo PEAR (Zhang et al. (2014)). Para mas detalles sobre esta herramienta, ver https://cme.h-its.org/exelixis/web/software/pear/

```
$ join_paired_ends.py -f forward.fastq -r reverse.fastq -b barcodes.fastq -j 100 -p 10 -o join/

## opciones
-f/--forward_reads_fp: indica la ruta del archivo forward.fastq;
-r/--reverse_reads_fp: la ruta del archivo reverse.fastq;
-b/--index_reads_fp la ruta del archivo barcodes.fastq generado en el paso anterior;
-j/--min_overlap: número de pares de bases que se sobrelapan en la unión. Ejemplo: 100;
-p/--perc_max_diff: es el porcentaje (%) de diferencias permitidas en la región de sobrelape;
-o/--output_dir: salida para los archivos generados. Ejemplo: join/

## Para los detalles del comando, ver:
> http://qiime.org/scripts/join_paired_ends.html
```

5. Demultiplexing: separar secuencias por barcodes, ligando a sus metadatos

El proceso de "demultiplexado" es separar las lecturas ensambladas en el paso anterior, y agruparlas o **etiquetarlas** con base en su #SampleID ó identificador **único** dentro del archivo de **metadatos** (o *mapping file*). Su separación es por su *barcode*, por ello estos deben ser únicos en cada corrida o experimento.

Para esto, se emplea el comando *split_libraries_fastq.py*, con los siguientes argumentos:

```
## Correr comando
$ split_libraries_fastq.py -i fastqjoin.join.fastq -b fastqjoin.join_barcodes.fastq -m mapping_file.txt

## Argumentos del comando:
-i/--sequence_read_fps: ruta del archivo fastqjoin.join.fastq;
-b/--barcode_read_fps: ruta del archivo fastqjoin.join_barcodes.fastq; -m/--mapping_fps: archivo de tex
--max_barcode_errors. número máximo de errores en el barcode; --barcode_type: tipo de barcode empleado,
-q/--phred_quality_threshold: Phred score (Q-score) mínima permisible que tendrán las secuencias, por e
-o/--output_dir: directorio de salida. En este ejemplo: split/

## Mayor detalle del comando, ver:
> http://qiime.org/scripts/split_libraries_fastq.html
```

6. Identificar y remover secuencias quiméricas:

La identificación y remoción de lecturas quiméricas es posible hacerlo antes y después del agrupamiento por OTUs (unidad operacional taxonómica; *operational taxonomic units*, por sus siglas en inglés). En este paso se emplean dos pasos: se identificar las lecturas que son quiméricas, y luego se toma dicha lista, y se remueve de las secuencias ensambladas y demultiplexadas.

```
## Identificar qué lecturas son quimeras:
$ identify_chimeric_seqs.py -i seqs.fna -r gg_13_8_otus/rep_set/97_otus.fasta -m usearch61 --threads 4

## remover las lecturas quiméricas identificadas:
$ filter_fasta.py -f seqs.fna -s chimeras/chimeras.txt -n -o non_chimeric_seqs.fna

## Para mayor información, puede ayudarse de la siguiente fuente:
> http://qiime.org/tutorials/chimera_checking.html
```

6.1. VSEARCH como herramienta para identificar y remover lecturas quiméricas

VSEARCH (Rognes et al. (2016); ver en GitHub) es otra herramienta para la identificación de quimeras, así como para su remoción. Su uso y detalles pueden ser revisados en https://github.com/torognes/vsearch/wiki/Chimera.

7. Agrupando las secuencias en OTUs: pick open otus

```
## Agrupando las secuencias en OTUs: de novo + closed reference
$ pick_open_reference_otus.py -i non_chimeric_
seqs.fna -p parameters_pickotus.txt -s 0.1 -m usearch61 -aO 4 -o pickotus/ --verbose

## opciones:
-m: método de agrupamiento; el default es *uclust*, pero puede usarse otros.
```

```
-a0 4: para indicarle al script que corra en paralelo algunos procesos, empleando 4 threads;
-p: archivo de texto plano que indica cambios al comportamiento del comando general. El archivo usado es
```

8. Resumen o estadísticos de la OTU-Table: BIOM Table

Para conocer datos generales de la OTU table generada en el paso anterior, bajo el nombre ***otu_table_mc2_w_tax_no_pynast_failures.biom***, podemos emplear el siguiente comando:

```
## Imprimir los datos directamente en la consola
$ biom summarize -i otu_table_mc2_w_tax_no_pynast_failures.biom

## Imprimir los datos a un archivo de texto plano
$ biom summarize -i otu_table_mc2_w_tax_no_pynast_failures.biom -o biomsummary.txt
```

9. Convertir OTU table a otros formatos (e.g., TSV or CSV)

La OTU-Table puede interconvertirse en otros formatos, e.g., table separated values (ó TSV), txt, csv, entre otros. Para conocer los detalles puede ver https://biom-format.org/documentation/biom_conversion.html.

A continuación se provee del comando regularmente empleado para hacer otros análisis a los datos de asignación taxoómica, empleando herramientas como la de R (y/o RStudio).

```
## Convertir la OTU Table a un archivo separado por tabulaciones (tsv) y que contenga la taxonomía asig
$ biom convert -i otu_table_mc2_w_tax_no_pynast_failures.biom -o table.from_biom_w_taxonomy.txt --to-ts
```

10. Otras fuentes de información

Finalmente, se provee de información adicional para continuar mejorando en el análisis de secuencias de marcados moleculares (16S rDNA) y el mundo de la bioinformática.

i. Página oficial de QIIME v1: http://qiime.org/index.html
ii. Tutorial de otros investigadores en GitHub de interés: https://github.com/frederic-mahe/swarm/wiki/Fred's-metabarcoding-pipeline; https://github.com/alexcritschristoph/Qiime16sTutorial
iii. "An Introduction to QIIME": https://twbattaglia.gitbooks.io/introduction-to-qiime/content/
iv. Artículo y flujo de trabajo detallados en Navas-Molina et al. (2013) - Advancing Our Understanding of the Human Microbiome Using QIIME, ver https://www.sciencedirect.com/science/article/abs/pii/B9780124078635000198?via%3Dihub
v. Foro dedicado a QIIME v1, en Google Forums: https://groups.google.com/g/qiime-forum