Taller4_EnsamblajeDeNovo

June 28, 2025

1 Taller ENSAMBLAJE DE NOVO

Francisco Salamanca

Objetivo: Aplicar un flujo de trabajo completo de ensamblaje genómico de novo utilizando datos reales de Mycoplasma genitalium, desde el análisis de calidad hasta su ensamblaje de novo, fomentando la comprensión del funcionamiento de cada herramienta y de los archivos generados.

Dataset:

- Organismo: Mycoplasma genitalium (genoma ~580 kb)
- Datos: Lecturas Illumina MiSeq 2x150 bp (pareadas)
- Accesión: ERR486840

1.1 Introducción

Mycoplasma genitalium Mycoplasma genitalium es una bacteria de pequeño tamaño perteneciente a la clase Mollicutes. Fue identificada por primera vez en la década de 1980 y es conocida por tener uno de los genomas más pequeños de todos los organismos autosuficientes (~580 kbp), lo que la convierte en un modelo interesante para estudios de biología mínima y genómica sintética. Desde el punto de vista clínico, M. genitalium es un patógeno de transmisión sexual (ETS) que puede causar infecciones tanto en hombres como en mujeres. En hombres, se asocia comúnmente con uretritis no gonocócica, mientras que en mujeres puede provocar cervicitis, enfermedad inflamatoria pélvica, y se ha vinculado con complicaciones reproductivas.

Algunas características relevantes de M. genitalium son:

Carece de pared celular, lo que la hace intrínsecamente resistente a antibióticos -lactámicos como la penicilina.

Crecimiento lento y difícil en cultivo, lo que complica su diagnóstico clínico.

Alta tasa de resistencia a antibióticos, especialmente a macrólidos (como la azitromicina) y fluoroquinolonas, lo que plantea un desafío en su tratamiento.

Su importancia clínica ha ido en aumento en los últimos años, siendo reconocida como una ETS emergente y motivo de preocupación por su resistencia creciente y su impacto en la salud sexual y reproductiva.

1.2 Desarrollo del taller

1.2.1 Parte 0: Copia de Archivos

Enlace los archivos originales en su espacio de trabajo creando un enlace simbòlico de los mismos:

ln -s /datos/resources/examples/rawdata/mycobacterium/ERR486840_1.fastq ERR486840_1.fastq

ln -s /datos/resources/examples/rawdata/mycobacterium/ERR486840_2.fastq ERR486840_2.fastq

[1]:
```
#Crear accesos directos a los datos de ejemplo de Mycobacterium tuberculosis

#!ln -s   /datos/resources/examples/rawdata/mycobacterium/ERR486840_1.fastq␣
↪ERR486840_1.fastq

#!ln -s   /datos/resources/examples/rawdata/mycobacterium/ERR486840_2.fastq␣
↪ERR486840_2.fastq
```

1.2.2 Parte 1: Análisis de Calidad

Realice un análisis de calidad de las lecturas forward y reverse de M. genitalium. Evalúe si se requiere o no de una edición de las lecturas antes de realizar el ensamblaje.

R/ NO es necesario una edicion de las lecturas antes de realizar el ensamblaje, pues contiene una calidad buena.

Preguntas asociadas

- ¿Cual es el tamaño promedio de las lecturas?

R/ Para los dos reads el promedio de las lecturas es de 150

- ¿Cuantas lecturas hay por librería?

R/ R1 y R2 Tienen 387568 lecturas

- ¿Cuantas secuencias sobre representadas tiene cada librería?

R/ R1 y R2 no tienen secuencias sobre representadas

[6]:
```
!ls
!cd /Users/fjosesala/Documents/GitHub/BFOC-2025-1/Taller4_EnsamblajeDeNovo/
↪datos && ls && fastqc  *.fastq
```

```
Taller4_EnsamblajeDeNovo.ipynb datos
ERR486840_1.fastq ERR486840_2.fastq
null
null
Started analysis of ERR486840_1.fastq
Approx 5% complete for ERR486840_1.fastq
Approx 10% complete for ERR486840_1.fastq
Approx 15% complete for ERR486840_1.fastq
```

```
Approx 20% complete for ERR486840_1.fastq
Approx 25% complete for ERR486840_1.fastq
Approx 30% complete for ERR486840_1.fastq
Approx 35% complete for ERR486840_1.fastq
Approx 40% complete for ERR486840_1.fastq
Approx 45% complete for ERR486840_1.fastq
Approx 50% complete for ERR486840_1.fastq
Approx 55% complete for ERR486840_1.fastq
Approx 60% complete for ERR486840_1.fastq
Approx 65% complete for ERR486840_1.fastq
Approx 70% complete for ERR486840_1.fastq
Approx 75% complete for ERR486840_1.fastq
Approx 80% complete for ERR486840_1.fastq
Approx 85% complete for ERR486840_1.fastq
Approx 90% complete for ERR486840_1.fastq
Approx 95% complete for ERR486840_1.fastq
Analysis complete for ERR486840_1.fastq
Started analysis of ERR486840_2.fastq
Approx 5% complete for ERR486840_2.fastq
Approx 10% complete for ERR486840_2.fastq
Approx 15% complete for ERR486840_2.fastq
Approx 20% complete for ERR486840_2.fastq
Approx 25% complete for ERR486840_2.fastq
Approx 30% complete for ERR486840_2.fastq
Approx 35% complete for ERR486840_2.fastq
Approx 40% complete for ERR486840_2.fastq
Approx 45% complete for ERR486840_2.fastq
Approx 50% complete for ERR486840_2.fastq
Approx 55% complete for ERR486840_2.fastq
Approx 60% complete for ERR486840_2.fastq
Approx 65% complete for ERR486840_2.fastq
Approx 70% complete for ERR486840_2.fastq
Approx 75% complete for ERR486840_2.fastq
Approx 80% complete for ERR486840_2.fastq
Approx 85% complete for ERR486840_2.fastq
Approx 90% complete for ERR486840_2.fastq
Approx 95% complete for ERR486840_2.fastq
Analysis complete for ERR486840_2.fastq
```

1.2.3 Parte 2: Ensamblaje de novo mediante Velvet

Velvet es un ensamblador de genomas diseñado específicamente para secuencias cortas generadas por tecnologías de secuenciación de alto rendimiento, como Illumina. Fue desarrollado por Daniel Zerbino y Ewan Birney en el EMBL-EBI y publicado en 2008.

Publicación: https://europepmc.org/article/pmc/2336801

Características principales: - Basado en grafos de de Bruijn: Velvet transforma las lecturas en un grafo de de Bruijn, lo que permite ensamblar eficientemente secuencias cortas, incluso en

presencia de errores.

- Optimizado para lecturas cortas: Ideal para lecturas de entre 35 y 300 pb, que eran comunes en las primeras generaciones de secuenciación masiva.

Componentes clave: - velveth: construye el grafo de de Bruijn a partir de las lecturas.

- velvetg: refina el grafo, elimina errores, resuelve ambigüedades y genera los contigs ensamblados.

Ventajas: - Rápido y eficiente en recursos para su época.

- Permite integrar pares de extremos (paired-end reads) para mejorar el ensamblaje.
- Ofrece varias opciones para limpieza de errores y ajuste de parámetros de ensamblaje.

Limitaciones: - Su rendimiento disminuye con lecturas más largas o datos más ruidosos.

- No es ideal para ensamblajes altamente complejos como metagenomas o genomas muy repetitivos (aunque hay extensiones como MetaVelvet para metagenómica).

[ ]:
```
#1. Crear el hash con velveth (usar k=31):
!velveth velvet31 31 -shortPaired -fastq -separate ERR486840_1.fastq␣
 ↪ERR486840_2.fastq

#velveth prepara la estructura de datos: separa las lecturas por tipo, calcula␣
 ↪los k-mers, y genera el directorio base.

#Explora el contenido del directorio velvet31/ y describe brevemente (asegurate␣
 ↪de entender) los archivos generados (Roadmaps, Sequences, etc).
```

[ ]:
```
#2. Ejecutar el ensamblaje:

!velvetg velvet31 -exp_cov auto -cov_cutoff auto

#velvetg ensambla los contigs usando el grafo de de Bruijn generado por velveth.
```

- ¿Qué función tienen los argumentos -exp_cov auto y -cov_cutoff auto ?

R/ Al usar auto en estas dos opciones permite que el velvetg decida automaticaente los valores que considere mas optimos segun la distrbucion de k-mers qu velveth genero en el paso 1.

-exp_cov auto (Expected K-mer Coverage), le indica a velvetg cual es la cobertura promedio que se espera para los k-mers que pertenecen al genoma real.

-cov_cutoff auto (Coverage Cutoff), establece un umbral minimo de cobertura para que un k-mer sea considerado en el ensamblaje. Elimina k-mers con baja frecuencia.

- Revisa los archivos contigs.fa, stats.txt, y LastGraph. ¿Qué métricas reporta stats.txt?

R/

Stasts.txt reporta:

- ID: Un identificador único para cada nodo.

- lgth: La longitud del nodo dada en bp.
- out: Número de aristas (conexiones) que salen de este nodo hacia otros.
- in : Número de aristas que llegan a este nodo desde otros
- long_cov: La cobertura del nodo por lecturas largas. Si es cero significa que no se usaron
- short1_cov: La cobertura del nodo por el primer set de lecturas cortas.
- short1_Ocov: La cobertura del "otro extremo" para el primer set de lecturas pareadas (paired-end).
- short2_cov: La cobertura por un segundo set de lecturas cortas .Si es cero significa que no se usa

contigs.fa contiene:

Secuencias de los contigs ensamblados en formato FASTA. Cada entrada tiene un encabezado que indica el ID del nodo (corresponde al del stats.txt), su longitud y su cobertura promedio.

LastGraph describe:

La estructura final del grafo de De Bruijn construido por Velvet. Es una representacion textual del ensamblaje final.

Contiene información detallada sobre cada nodo, incluyendo:

- ID del nodo, longitud, y numero de lecturas que lo componen.
- Secuencias de los nodos.
- Conexiones entre nodos, indicando cuales estan conectados y su orientacion
- in fo de el número de lecturas en cada conexion

1.2.4 Parte 3: Ensamblaje de novo mediante MegaHit

MEGAHIT es un ensamblador de genomas diseñado para manejar grandes volúmenes de datos de secuenciación, especialmente (aunqe no limitado) aquellos provenientes de estudios metagenómicos, donde se busca ensamblar múltiples genomas microbianos a partir de muestras ambientales complejas.

Características principales:

- Basado en grafos de de Bruijn comprimidos: Utiliza una estructura de datos eficiente llamada succinct de Bruijn graph (SdBG), que permite representar el grafo con un uso mínimo de memoria.
- Diseñado para escalar bien: Capaz de ensamblar datasets muy grandes, incluso en computadoras con recursos moderados, gracias a su enfoque optimizado de uso de memoria y CPU.
- Soporte para secuencias cortas y largas: Aunque fue optimizado inicialmente para lecturas cortas (Illumina), versiones más recientes permiten trabajar también con lecturas largas (por ejemplo, Nanopore o PacBio) mediante estrategias híbridas.

Componentes clave:

- Monolítico: todo el proceso de ensamblaje está integrado en un solo comando (megahit), simplificando su uso.
- Permite ensamblajes de novo, sin necesidad de genoma de referencia.

- Integra mecanismos de limpieza de errores y construcción jerárquica del grafo (multi-k-mer), lo que mejora la calidad del ensamblaje.

Ventajas:

- Altamente eficiente en memoria y velocidad, incluso con muestras metagenómicas complejas.
- Requiere pocos pasos y es fácil de usar.
- Bien mantenido y ampliamente adoptado en estudios metagenómicos.

Limitaciones:

- Aunque muy robusto, en algunos casos puede generar ensamblajes fragmentados, especialmente en presencia de organismos con genomas muy similares.
- Menos adecuado para ensamblajes donde se requiera información estructural detallada (como reordenamientos o ensamblajes haplotípicos), ya que su enfoque se centra en la eficiencia y escalabilidad.

[ ]:
```
#Correr megahit
!megahit -1 ERR486840_1.fastq -2 ERR486840_2.fastq -o megahit_out
```

[ ]:
```
#Hallar la cantidad de contigs ensamblados
!megahit_out % grep ">" final.contigs.fa | wc
```

Examina el archivo megahit_out/log para conocer los pasos ejecutados.

- El archivo final.contigs.fa contiene los contigs ensamblados. ¿Cuantos contigs obtuvo?

R/ La cantidad de contigs ensamblados es de 22.

- ¿Cómo podria optimizar el tiempo de análisis de megahit para genomas o metagenomas de gran tamaño?

R/ Utilizando los parametros por defecto que tiene megahit, como:

- meta-sensitive: Similar a la configuración por defecto. Bueno para metagenomas de complejidad media, pero puede ser lento para conjuntos de datos muy grandes.
- meta-large: Recomendado para metagenomas grandes y complejos, ya que usa kmers mas espaciados, reduciendo complejidad y tiempo de ejecucion
- meta huge: Para ensamblajes de metagenomas extremadamente grandes (Gbp o Tbp)

Tambien utilizando eficientemente los recursos del hardware:

- Ajuste de hilos: megahit -t 16 (16 hilos)
- Ajuste de memoria: megahit -m 0.8 (80% de RAM)
- Ajuste de SSD.

Finalmente, teniendo lecturas procesadas como entrada, que hayan pasado por un buen contol de calidad.