Published on Tue 29 September 2020 By Adam Freedman, Nathan Weeks , tagged as Next-Gen Sequencing , Transcriptome , Transcriptome Assembly , Trinity

Publicacion Trinity (https://informatics.fas.harvard.edu/best-practices-for-de-novo-transcriptome-assembly-with-trinity.html)

Los ensambladores de transcriptomas de novo utilizan gráficos DeBruijn, que se construyen y amplían en función de kmers, es decir, subsecuencia de longitud k que se encuentra en las lecturas. Si bien esto hace que el proceso de ensamblaje sea computacionalmente manejable, puede conducir a ensamblajes fragmentados de un gran número de contigs que son subsecuencias de las transcripciones verdaderas subyacentes.

Algunos de los factores que conducen a esta fragmentación son los errores de secuenciación, el polimorfismo, las repeticiones de secuencia y, para las transcripciones de expresión más baja, la estocasticidad de la profundidad de lectura que conduce a lagunas en la cobertura. Los ensambladores de transcriptomas, a diferencia de los ensambladores de genomas, deben manejar el amplio rango de profundidad de cobertura debido a la variación de la expresión génica. Nuestro objetivo en el desarrollo de una canalización de mejores prácticas es producir la mayoría de los ensamblajes de transcriptomas contiguos, sin errores y completos dados estos desafíos.

TRINITY (https://github.com/trinityrnaseq/trinityrnaseq/wiki) es un paquete de software para realizar el ensamblaje del transcriptoma de novo (así como la versión guiada por el genoma) a partir de datos de RNA-seq. El paquete Trinity también incluye una serie de scripts de Perl para generar estadísticas para evaluar la calidad del ensamblaje y para empaquetar herramientas externas para realizar análisis posteriores.

1.- Examinar métricas de calidad para secuenciar lecturas

El examen de la distribución de la calidad base, las frecuencias kmer y la contaminación del adaptador por posición en la lectura es un primer paso importante para comprender la calidad subyacente de sus datos. Por ejemplo, un aumento en la frecuencia del adaptador a medida que uno se mueve a lo largo de una lectura es indicativo de una eliminación incompleta de la secuencia del adaptador durante la demultiplexación, una ocurrencia bastante común. Además, la presencia de secuencias sobrerrepresentadas puede ser indicativa de contaminación del adaptador, lecturas de ARNr o quizás otra contaminación exógena. Utilice fastqc para examinar las métricas de calidad de sus lecturas. Un ejemplo de script de envío de slurm es el siguiente:

In [ ]:
```
#!/bin/bash
#SBATCH -p serial_requeue       # Partition to submit to
#SBATCH -N 1
#SBATCH -n 6                    # Number of cores
#SBATCH -t 0-3:00               # Runtime in days-hours:minutes
#SBATCH --mem 6000              # Memory in MB
#SBATCH -J FastQC               # job name
#SBATCH -o FastQC.%A.out        # File to which standard out will be w
ritten
#SBATCH -e FastQC.%A.err        # File to which standard err will be w
ritten
#SBATCH --mail-type=ALL         # Type of email notification- BEGIN,EN
D,FAIL,ALL
#SBATCH --mail-user=<PUT YOUR EMAIL ADDRESS HERE>  # Email to which no
tifications will be sent

readonly SINGULARITY_IMAGE='singularity exec --cleanenv /n/singularity
_images/informatics/trinityrnaseq/trinityrnaseq.v2.11.0.simg'

${SINGULARITY_EXEC} fastqc --threads ${SLURM_CPUS_ON_NODE} --outdir `
pwd` $1
```

El script anterior usa un argumento de línea de comando, especificado por $ 1, que sería el nombre del archivo fastq. Por lo tanto, la presentación de trabajos se vería así:

$ sbatch myfastqcscript.sh mypurpleunicorn_1.fastq

Los resultados incluirán un informe en formato html y archivos de texto que resuman diversa información de métricas de calidad. Tenga en cuenta que en el script anterior y en los siguientes, usamos la práctica de cargar primero nuevos módulos para crear acceso al conjunto de módulos actual, luego purgar cualquier módulo que pueda haberse cargado desde su archivo .bash_profile o .bashrc para evitar conflictos. y luego cargar los módulos necesarios para el análisis actual.

2.- Eliminación de k-mers erróneos de lecturas de extremo emparejado de Illumina

Debido a que los ensambladores de transcriptomas de última generación utilizan un enfoque de DeBruijn Graph que construye gráficos a partir de kmers, los k-mers erróneos pueden afectar negativamente a los ensamblajes. Debido a que es probable que los k-mers raros se deban a errores de secuenciación, corregir las lecturas de modo que los k-mers raros se corrijan a uno que ocurre con mayor frecuencia puede mejorar los ensamblajes. En teoría, las transcripciones de baja expresión pueden dar lugar a la aparición de kmers raros y biológicamente reales que se marcarán como errores. Sin embargo, las herramientas de ensamblaje generalmente no harán un buen trabajo al ensamblar transcripciones de baja expresión de todos modos. Por lo tanto, en nuestra opinión, los beneficios de eliminar los errores que afectarán el ensamblaje de muchas transcripciones superan cualquier efecto adverso en la reconstrucción de transcripciones de baja expresión cuyo ensamblaje ya se verá comprometido por una baja cobertura de lectura.

Usamos rCorrector, una herramienta diseñada específicamente para datos de RNA-seq. Además de ser uno de los mejores en las comparaciones lado a lado de herramientas de corrección de errores de lectura basadas en kmer, incluye etiquetas en la salida fastq que indican si la lectura se ha corregido o se ha detectado que contiene un error, pero es incorregible.

Cree un directorio en el que ejecutar rCorrector y cree enlaces simbólicos desde sus archivos fastq a este directorio (para mantener seguros sus datos sin procesar). Luego, ejecute un script sbatch:

In [ ]:
```
#!/bin/bash
#SBATCH -N 1
#SBATCH -n 12
#SBATCH -p general,shared
#SBATCH -e rcorrect_%A.e
#SBATCH -o rcorrect_%A.o
#SBATCH -J rcorrect
#SBATCH --mem=24000
#SBATCH --time=36:00:00

module purge
module load Rcorrector/20180919-fasrc01
perl /n/helmod/apps/centos7/Core/Rcorrector/20180919-fasrc01/bin/run_rcorrector.pl -t 12 -1 $1 -2 $2
```

Los $1y$ 2 en el script son argumentos de línea de comando para listas separadas por comas de archivos R1 y R2 para sus lecturas de pares. Por lo tanto, si el script se llama rcorrector.sh, enviaría el trabajo de la siguiente manera:

In [ ]:
```
$ sbatch rcorretor.sh mypurpleunicorn_1.fastq,myyellowunicorn_1.fastq mypurpleunicorn_2.fq,myyellowunicorn_2.fq
```

Los archivos fastq de salida incluirán "cor" en sus nombres. Las lecturas corregidas tendrán un sufijo "cor" en sus etiquetas, p. Ej.

In [ ]:
```
@ERR1101637.57624042/1 l:165 m:203 h:218 cor
GTACAACCCTTCCAACCTCCACCGTCTTATATACGAAGCGCCTTGAGTGTGTGTGTGCATGAGCCAAAGGGAATACCG
+
<@@D=D2ACDAHB?:<8EGE;B;FE<?;??DBDD))0)8@GDF@C)))5-;5CAE=..7;@DEEC@C;A9?BB=@>@B
```

Los valores l, m y h en los encabezados de lectura indican los recuentos de kmer (en el conjunto de archivos fastq) para los kmers de frecuencia más baja, mediana y más alta que ocurren en la lectura.

Las lecturas con kmers erróneas que no se pueden corregir también se marcarán, p. Ej.

In [ ]:
```
@ERR1101637.65/1 l:1 m:45 h:71 unfixable_error
GGTTCTGTGCCTTTCCCTACCGGAGAGGCGCCCCCAAGACGTATAAGAAGCACCGCTCTTCTCACGCGCA
TACCAACAACTTCGCAAAGTCTGTGGTCAAC
+
@<@DDFBDHHBHHGIIIIIIIIGAG:EGAEGEIIIIBGGHI;C>EAA).;BBDDCC>BD@>>CCABBDD@
-2@?::8(2?@@AB<@<@C>CC>A>?944>@
```

3.- Descartar los pares de lectura para los que una de las lecturas se considere no reparable

Estas lecturas no reparables a menudo están plagadas de N o representan otras secuencias de baja complejidad. ¿Por qué arriesgarse a incorporar basura en su ensamblaje y por qué dedicar tiempo de cálculo a lecturas que solo pueden dañar el ensamblaje? No hay una buena razón, así que quítelos. Proporcionamos un script de Python para lograr esta tarea. También quita la etiqueta "cor" de los encabezados de las secuencias corregidas, ya que pueden causar problemas para las herramientas posteriores, especialmente si está utilizando datos de SRA. No preguntes por qué ... solo te dará dolor de cabeza. Se puede obtener una secuencia de comandos de Python para eliminar cualquier par de lectura en el que al menos una lectura tenga un error que no se pueda corregir, en el repositorio TranscriptomeAssemblyTools de Harvard Informatics GitHub. Clone el repositorio y ejecute el script en SLURM usando el siguiente script:

In [ ]:
```
#!/bin/bash
#SBATCH -J rmunfix_$1                  # Job name
#SBATCH -n 1                           # Use 1 core for the job
#SBATCH -t 03:00:00                    # Runtime in HH:MM:SS
#SBATCH -p serial_requeue              # Partition to submit to
#SBATCH --mem=2000                     # Memory per node in MB
#SBATCH -o rmunfix_%A.o                # File to which STDOUT will be wr
itten
#SBATCH -e rmunfix_%A.e                # File to which STDERR will be wr
itten
#SBATCH --mail-type=ALL                # Type of email notification- BEG
IN,END,FAIL,ALL
#SBATCH --mail-user=name@harvard.edu # Email to which notifications wi
ll be sent

module purge
module load python/2.7.8-fasrc01

python /PATH/TO/FilterUncorrectabledPEfastq.py -1 $1 -2 $2 -s $3
```

$1y$ 2 son argumentos de línea cmd para nombres de archivos fastq R1 y R2, respectivamente, y $ 3 es un nombre de muestra que se agregará al registro que resumió cuántas lecturas se eliminaron.

4.- Adaptador de recorte y bases de baja calidad de archivos fastq

La tubería de demultiplexación de Illumina puede eliminar de forma incompleta las secuencias de adaptadores, y cuando los tamaños de inserción para un par de lecturas dadas dan lugar a superposiciones entre las bases secuenciadas, la secuenciación de una lectura puede extenderse al adaptador de la otra. Estas bases, así como las bases de baja calidad, deben recortarse antes de ejecutar Trinity. Sin embargo, existe evidencia de que un recorte de calidad excesivo (al establecer un umbral de calidad base alto) puede afectar negativamente a los ensamblajes, McManes et al. 2014. Con base en estos hallazgos, actualmente recomendamos filtrar las bases con cualidades por debajo de phred 5.

Si bien Trimmomatic se usa comúnmente para el recorte de adaptadores y de calidad, las gráficas de composición por ciclo del adaptador recientemente agregadas a fastqc han revelado que no elimina por completo la secuencia del adaptador, con altos índices de contaminación del adaptador que quedan en las últimas bases de una lectura. Por el contrario, TrimGalore !, una envoltura conveniente para cutadapt, parece eliminar completamente la contaminación del adaptador en las lecturas. Por tanto, es nuestra herramienta preferida. TrimGalore! no es un módulo sobre odyssey, pero se puede descargar y ejecutar de inmediato. Basta con cargar el módulo cutadapt. Primero, instale TrimGalore. en su directorio personal donde guarda el software.

In [ ]:
```
$ wget https://github.com/FelixKrueger/TrimGalore/archive/0.6.0.zip
$ unzip trim_galore_v0.6.0.zip
```

Luego, cree un directorio de salida para sus resultados de recorte, cree enlaces simbólicos de los archivos fastq corregidos por Rcorrector (con las lecturas no reparables eliminadas) y envíe el trabajo de recorte con un script sbatch:

In [ ]:
```
#!/bin/bash
#SBATCH -J trimgalore
#SBATCH -n 1                        # Use 1 cores for the job
#SBATCH -t 0-6:00                    # Runtime in D-HH:MM
#SBATCH -p serial_requeue            # Partition to submit to
#SBATCH --mem=3000                  # Memory pool for all cores (see also
--mem-per-cpu)
#SBATCH -o trimgalore_PE.%A.out  # File to which STDOUT will be writte
n
#SBATCH -e trimgalore_PE.%A.err  # File to which STDERR will be writte
n
#SBATCH --mail-type=ALL              # Type of email notification- BEGIN,
END,FAIL,ALL
#SBATCH --mail-user=name@harvard.edu # Email to send notifications to

module purge
module load cutadapt/1.8.1-fasrc01

# $1 = R1 reads
# $2 = R2 reads

/your/path/to/trim_galore --paired --retain_unpaired --phred33 --outpu
t_dir trimmed_reads --length 36 -q 5 --stringency 1 -e 0.1 $1 $2
```

Algunas de estas opciones se pueden modificar para su conjunto de datos, p. Ej. Si está analizando datos de un solo extremo, obviamente no necesita ninguno de los argumentos que especifican el manejo de datos emparejados, ¡ni necesita especificar las lecturas de R1 y R2! Además, es posible que tenga motivos para cambiar el umbral de longitud de lectura mínima o los parámetros --strigency y -e que pertenecen a la detección y el filtrado del adaptador. En nuestra experiencia, estos ajustes han funcionado bien. Si desea reutilizar este script para recortar lecturas que luego alineará con un genoma, puede optar por ser más estricto con respecto a la calidad base mínima permitida.

Nota para los usuarios de bowtie: si alguna de sus aplicaciones posteriores usa bowtie (no bowtie2), será necesario proporcionar también el indicador -t a TrimGalore. Cuando una coordenada de inicio / fin de una lectura de un par alineado está contenida dentro de su pareja, bowtie informará que la alineación no es válida. Si usó TrimGalore anteriormente y luego usó la versión anterior de los scripts de Perl de Trinity para evaluar el soporte de lectura que usaba bowtie, y no proporcionó la marca -t, la estimación informada del porcentaje de pares correctamente mapeados será incorrecta y podría subestime sustancialmente el soporte para su montaje.

Bibliotecas construidas con el kit de ARNm Wafergen PrepX en el robot Apollo: para las bibliotecas construidas con los kits de biblioteca de ARNm direccional Wafergen PrepX en el robot Apollo, hemos visto casos en los que TrimGalore! no elimina los adaptadores en su totalidad utilizando la configuración predeterminada de TrimGalore. Si el informe fastqc para las lecturas recortadas aún indica un aumento en la ocurrencia del adaptador a medida que uno se mueve a lo largo de la lectura, especifique los complementos inversos de los adaptadores Wafergen específicos y la secuencia de índice, de modo que su TrimGalore! La línea de comando para el script de envío anterior se ve así:

In [ ]:
```
  your/path/to/trim_galore --paired --retain_unpaired --phred33 -a AAG
ATCGGAAGAGC  -a2 GATCGTCGGACTGTAGAA --output_dir trimmed_reads --lengt
h 36 -q 5 --stringency 5 -e 0.1 $1 $2
```

La secuencia proporcionada para -a es A + a subsecuencias del complemento inverso del cebador de índice 3 '(incluido, cuando se usa, un índice de código de barras). Para -a2, es una subsecuencia del complemento inverso del cebador de índice 5 '. Si bien, en principio, se podrían suministrar las secuencias de adaptadores de longitud completa (que incluirían los complementos inversos de los voladizos), en la práctica hemos descubierto que el uso de secuencias más cortas aumenta la sensibilidad y minimiza la posibilidad de retener la secuencia de adaptadores en lecturas recortadas.

Consulte los documentos de flujo de trabajo de PrepX de Wafergen Biosystems para obtener más detalles sobre las secuencias de adaptadores.

Finalmente, si tiene muchos archivos fastq que desea ejecutar por separado, ya sea en los pasos fastqc o de recorte, debe considerar escribir un script de bucle que iterará sobre los archivos y enviará envíos de sbatch, en lugar de proporcionar manualmente argumentos de línea de comando para un par de lecturas a la vez.

5.- Asignar lecturas recortadas a una lista negra para eliminar las lecturas de ARNr no deseadas - OPCIONAL

En la mayoría de los casos, los investigadores elegirán la selección poli-A sobre el agotamiento del ribo, ya sea por interés en la secuencia de codificación o por el hallazgo de que el agotamiento del ribo puede conducir a sesgos en los análisis posteriores. Lahens et al. 2014, Biología del genoma. Sin embargo, las estrategias de preparación de bibliotecas que utilizan la selección poli-A normalmente no eliminarán todo el ARNr, y hemos visto que las frecuencias de las lecturas que se originan en la post-selección del ARNr superan ocasionalmente el 30%. La eliminación de las lecturas que se originan en el ARNr reducirá el uso del clúster de Cannon y el tiempo de ensamblaje. Igualmente importante, tendrá una estimación más precisa de cuántas de sus lecturas se destinarán realmente al ensamblaje de transcripciones de ARNm.

Nuestra recomendación es que primero asigne sus lecturas a una base de datos de ARNr, como la que se puede descargar en formato rápido desde SILVA. Luego, se puede ejecutar bowtie2 para maximizar la sensibilidad del mapeo, lo que significa que maximizará el número de lecturas que considerará que se originan a partir de ARNr y, por lo tanto, dignas de filtrarse de su conjunto de lectura final para ensamblar. Desde SILVA, descargamos los archivos fasta SSUParc y LSUParc, concateándolos y reemplazando los caracteres U con T, ya que nuestras lecturas de secuencia están en el espacio de ADN. Luego, para el archivo fasta concatenado, crea un índice bowtie2 para la base de datos rRNA fasta, consulte las instrucciones en el manual bowtie2 y mapee sus lecturas:

In [ ]:
```
#!/bin/bash
#SBATCH -N 1
#SBATCH -n 12 #Number of cores
#SBATCH -t 12:00:00  #Runtime in minutes
#SBATCH -p serial_requeue  #Partition to submit to
#SBATCH --mem=12000  #Memory per node in MB
#SBATCH -e silvabt2_%A.e
#SBATCH -o silvabt2_%A.o

# $1 = full path to your silva database; do not include the fasta suff
ix (fa,fasta,etc.)
# $2 = R1 fastq file
# $3 = R2 fastq file
# $4 = sample_id (no spaces)

singularity exec --cleanenv /n/singularity_images/informatics/trinityr
naseq/trinityrnaseq.v2.11.0.simg bowtie2 --quiet --very-sensitive-loca
l --phred33  -x ${1} -1 ${2} -2 ${3} --threads 12 --met-file ${4}_bowtie2_m
etrics.txt --al-conc-gz blacklist_paired_aligned_${4}.fq.gz --un-conc-
gz blacklist_paired_unaligned_${4}.fq.gz  --al-gz blacklist_unpaired_a
ligned_${4}.fq.gz --un-gz blacklist_unpaired_unaligned_${4}.fq.gz
```

Se escribirán tanto un R1 como un R2 para cada uno de los conmutadores.

Las lecturas que desea conservar son las correspondientes a los pares de lectura para los que ninguna de las lecturas se asignó a la base de datos de ARNr, es decir. lecturas "paired_unaligned", especificadas después de la marca --un-conc-gz.

Si sus bibliotecas de RNA-seq están creadas con un protocolo trenzado, debe usar la configuración bowtie2 relevante:

para bibliotecas basadas en dUTP (Illumina TruSeq, NEBNext Ultra Directional), que son "RF" en el lenguaje de Trinity, use la bandera bowtie2 --nofw para los protocolos de ligadura trenzada, es decir, bibliotecas de ARNm direccional Wafergen PrepX creadas en el robot Apollo, que son "FR" en el lenguaje de Trinity, use la bandera bowtie2 --norc Si tiene dudas sobre qué protocolo de trenzado se utiliza en el kit de preparación de su biblioteca, consulte el manual del usuario o póngase en contacto con el soporte técnico del fabricante. por supuesto, si su kit no es un protocolo específico de cadena, ¡ignore todo esto! NOTA: Se puede construir una base de datos de lista negra a partir de cualquier secuencia no objetivo, p. parásitos, bacterias, otras fuentes potenciales de ARN exógeno. También se puede aumentar una base de datos de rRNA con secuencias de rRNA conocidas para su taxón o uno muy relacionado, si no aparecen en SILVA.

6.- Ejecute fastqc en sus lecturas procesadas que pasan el control de calidad y el filtrado de los pasos anteriores

Utilice el mismo formato de envío de sbatch del paso 2 (arriba). Idealmente, no habrá una tendencia en la contaminación del adaptador por ciclo, y habrá una mayor uniformidad en las distribuciones de kmer, contenido de GC, sin secuencias sobrerrepresentadas, etc.

7.- Eliminar secuencias restantes sobrerrepresentadas - OPCIONAL

En ocasiones, fastqc detectará las secuencias sobrerrepresentadas incluso después de ejecutar los pasos anteriores. Uno debe BLASTARlos para ver qué son y considerar el uso de un script para eliminar los pares de lectura que contienen las secuencias sobrerrepresentadas. Por lo general, estos serán ARNr que no estaban suficientemente representados en la base de datos SILVA (por ejemplo, 5S), u otros ARNr más comunes que sus lecturas no se asignarán porque es demasiado divergente de los organismos utilizados para construir la base de datos. Proporcionamos un script de Python, RemoveFastqcOverrepSequenceReads.py, para usar las secuencias marcadas por fastqc para eliminar pares de lectura donde cualquiera de las lecturas contiene una de sus respectivas secuencias sobrerrepresentadas.

8.- Ejecutar Trinity

Seguimos evaluando otros ensambladores de transcriptomas de novo, pero en la actualidad recomendamos Trinity, ya que funciona relativamente bien, utiliza los recursos informáticos de manera eficiente y cuenta con el apoyo continuo de sus desarrolladores, y la distribución incluye scripts para realizar una serie de análisis posteriores para la calidad del ensamblaje evaluación y estimación de expresión.

La configuración utilizada para Trinity dependerá de varios factores, incluida la estrategia de secuenciación, si las bibliotecas están varadas y el tamaño del conjunto de datos.

Normalización. En la última versión de Trinity, la normalización in silico se realiza de forma predeterminada. Para conjuntos de datos con> 200 millones de lecturas después de los pasos de filtrado anteriores, por consideraciones computacionales, recomendamos usar el modo predeterminado y permitir que Trinity normalice las lecturas. Si desea desactivar la normalización, incluya la marca --no_normalize_reads en su línea de comandos de Trinity.

Ejecutando Trinity dentro de una imagen de contenedor de Singuarlity. Reconstruir un nuevo módulo Trinity con cada actualización es cada vez más complicado, ya que se agrega funcionalidad junto con dependencias adicionales. Por lo tanto, nuestro modo preferido de ejecutar Trinity (así como el modo preferido para los desarrolladores de Trinity) es hacerlo dentro de una imagen de contenedor de Singularity, que funciona como una máquina virtual liviana, dentro de la cual las dependencias de software están convenientemente agrupadas y son relevantes. las variables de entorno están configuradas correctamente. Para la comodidad de los usuarios del Harvard FAS Cannon Cluster, proporcionamos imágenes del Trinity Singularity Image Archive en / n / singularity_images / informatics / trinityrnaseq /.

Ejecutar Trinity a través de Singularity implica dos pasos. Primero ejecutamos Trinity como un trabajo SLURM. A continuación se muestra un script de ejemplo para un trabajo de Trinity (con normalización):

El sistema de archivos recomendado actualmente en Cannon desde el cual ejecutar trabajos de Trinity es holyscratch01, ya que está ubicado en el mismo centro de datos que los nodos de cómputo de Cannon (Holyoke). No envíe trabajos de Trinity desde un sistema de archivos boslfs / boslfs02 (ubicado en Boston). Los archivos de entrada FASTQ / FASTA pueden residir en boslfs / boslfs02 siempre que trinity_out_dir (creado de forma predeterminada en el directorio desde el que se envía el script de trabajo; consulte la variable TRINITY_OUT_DIR a continuación) se encuentre en holyscratch01.

The current recommended file system on Cannon from which to run Trinity jobs is holyscratch01, as it is colocated in the same data center as the Cannon compute nodes (Holyoke). Do not submit Trinity jobs from a boslfs / boslfs02 file system (located in Boston). Input FASTQ/FASTA files may reside on boslfs/boslfs02 as long as the trinity_out_dir (created by default in the directory from which the job script is sumitted; see the TRINITY_OUT_DIR variable below) is located on holyscratch01.

In [ ]:
```
#!/bin/sh
# use an entire node in the specified Slurm partition
#SBATCH --nodes=1
#SBATCH --mem=0
#SBATCH --exclusive
# Adjust wall time limit as appropriate for partition. If the job is c
ancelled
# due to time limit exceeded, the job script can be resubmitted, and T
rinity
# will resume execution after the last completed.
#SBATCH --time=72:00:00
# Resubmit to the "bigmem" partition if inchworm std::bad_alloc error
occurs.
# The job will then stop after inchworm completes, and can be resubmit
ted to the
# "shared" or "test" partition, which are preferred for faster / more
reliable
# execution and sooner job start time.
#SBATCH --partition=shared

set -o nounset -o errexit -o xtrace

########################################
# parameters
########################################
readonly SINGULARITY_IMAGE=/n/singularity_images/informatics/trinityrn
aseq/trinityrnaseq.v2.11.0.simg
readonly TRINITY_OUT_DIR=trinity_out_dir
# To see all options:
#    singularity exec --cleanenv ${SINGULARITY_IMAGE}  Trinity --show_
full_usage_info
readonly TRINITY_OPTIONS="--output ${TRINITY_OUT_DIR} --max_memory $((
8*$(ulimit -m)/(1024**2)/10))G --CPU ${SLURM_CPUS_ON_NODE} $@"
```

```
########################################
# ... don't modify below here ...
if [ ! -s "${TRINITY_OUT_DIR}/read_partitions.img" ]
then
  mkdir -p "${TRINITY_OUT_DIR}"
  readonly tmpdir=$(mktemp -d)
  mkdir -m 777 -p ${tmpdir}/upper ${tmpdir}/work
  truncate -s 2T "${TRINITY_OUT_DIR}/read_partitions.img"
  singularity exec --cleanenv ${SINGULARITY_IMAGE} mkfs.ext3 -d "${tmp
dir}" "${TRINITY_OUT_DIR}/read_partitions.img"
  singularity exec --cleanenv --overlay ${TRINITY_OUT_DIR}/read_partit
ions.img ${SINGULARITY_IMAGE} mkdir /read_partitions
  ln -sf /read_partitions ${TRINITY_OUT_DIR}/read_partitions
  rm -rf "${tmpdir}"
fi

# if on a bigmem node, stop after inchworm
case ${SLURM_JOB_PARTITION} in
  bigmem) no_run_chrysalis='--no_run_chrysalis' ;;
       *) no_run_chrysalis='' ;;
esac

srun -n 1 env time -v singularity exec \
                            --cleanenv \
                            --no-home \
                            --overlay ${TRINITY_OUT_DIR}/read_partition
s.img \
                            "${SINGULARITY_IMAGE}" \
  Trinity ${TRINITY_OPTIONS} ${no_run_chrysalis}
```

Este script establece automáticamente las opciones Trinity --max_memory y --CPU según las características del hardware del nodo de cálculo en el que se ejecuta el trabajo.

Si este script se guarda en un archivo llamado trinity.sh, enviaríamos el trabajo Slurm de esta manera:

In [ ]:
```
sbatch trinity.sh --seqType fq --left {comma-separated R1 fastq files}
--right {comma-separated R2 fastq files}
```

Alternativamente, en lugar de agregar opciones al comando sbatch, la cadena TRINITY_OPTIONS en el script trinity.sh podría editarse para reflejar características particulares deseadas, por ejemplo:

_Desactivar la normalización (--no_normalize*reads)*

_Para las bibliotecas direccionales, --SS_lib*type debe establecerse en FR o RF para la construcción de bibliotecas ligadas y basadas en dUTP, respectivamente.*

_Una forma alternativa de especificar una gran cantidad de archivos fastq es usar --left_list y --right*list y hacer que los argumentos apunten a archivos txt que proporcionen los nombres de ruta completos de los archivos R1 y R2, respectivamente, con 1 fila por archivo*

Luego, el script del trabajo se enviaría sin argumentos:

In [ ]:
```
sbatch trinity.sh
```

Una vez que la ejecución de Trinity se haya completado con éxito, será necesario inspeccionar los resultados, que están escritos (por defecto) en trinity_out_dir / Trinity.fasta.

El directorio trinity_out_dir / read_partitions puede contener cientos de miles o millones de archivos de componentes generados por Chrysalis (gráficos de Brujin y lecturas de secuencia particionadas) que se ingresan para Butterfly, que genera archivos adicionales que contienen transcripciones completas. Para la eficiencia de E / S, el contenido del directorio trinity_out_dir / read_partitions se escribe en un archivo de imagen de superposición (trinity_out_dir / read_partitions.img). El archivo de imagen read_partitions.img se crea como un archivo disperso, por lo que inicialmente consume mucho menos espacio en disco (como lo indica el comando du, que informa el uso del disco) que su tamaño máximo (2T):

In [ ]:
```
$ ls -lh read_partitions.img
-rw-rw----+ 1 user group 2.0T Dec  5 16:22 read_partitions.img
$ du -h read_partitions.img
34G     read_partitions.img
```

9.- Evaluación de la calidad del ensamblaje, paso 1: métricas de resumen de alineación básicas

Las métricas como N50 nunca deben, por sí solas, tratarse como buenos indicadores de la calidad del ensamblaje. Un ejemplo obvio, aunque extremo, es que si ensambla incorrectamente todas sus lecturas en un contig gigantesco, su N50 será muy grande. Sin embargo, los N50 extremadamente cortos, de modo que representen una fracción del tamaño esperado de los fragmentos en su biblioteca, podrían indicar otros problemas. De manera similar, el número de "transcripciones" y "genes" en su ensamblaje Trinity no proporciona ninguna métrica absoluta de calidad. Sin embargo, cuantos más "genes" ensambla Trinity, particularmente si tienen solo unos pocos cientos de bases, más probable es que sus contigs representen subsecuencias de genes reales. Dejando de lado estas advertencias, puede generar fácilmente estadísticas N50 y recuentos del número de contigs Trinity en un trabajo interactivo utilizando el script TrinityStats.pl que viene con Trinity.

In [ ]:
```
srun --pty -p shared -t 00:20:00 --mem 500 /bin/bash
singularity exec --cleanenv /n/singularity_images/informatics/trinityr
naseq/trinityrnaseq.v2.11.0.simg sh -c '$TRINITY_HOME/util/TrinityStat
s.pl Trinity.fasta' > Trinity_assembly.metrics
```

10.- Evaluación de la calidad del ensamblaje, paso 2: cuantifique el soporte de lectura para el ensamblaje

Como se explica en la documentación de Trinity, las transcripciones ensambladas pueden no representar el complemento completo de las lecturas de pares. Esto ocurrirá porque, para contigs muy cortos, solo una lectura de la lectura del extremo emparejado se alineará con él. Simplemente mapear sus lecturas con pajarita (o su alineador de elección) a las transcripciones no arrojará ninguna información sobre este fenómeno, ya que solo se informarán los pares de lectura correctamente mapeados. Evaluar el soporte de lectura para el ensamblaje es un proceso de tres pasos. Primero, crea un índice bowtie2 para su ensamblaje.

In [ ]:
```
#!/bin/bash
#SBATCH -N 1
#SBATCH -n 4 #Number of cores
#SBATCH -t 0:00:00  #Runtime in minutes
#SBATCH -p serial_requeue,shared  #Partition to submit to
#SBATCH --mem=8000  #Memory per node in MB
#SBATCH -e bt2build.e
#SBATCH -o b2build.o

# $1 = your assembly fasta
assembly_prefix=$(basename $1 |sed 's/.fasta//g')

readonly SINGULARITY_EXEC='singularity exec --cleanenv /n/singularity_
images/informatics/trinityrnaseq/trinityrnaseq.v2.11.0.simg'

${SINGULARITY_EXEC} bowtie2-build --threads 4 $1 $assembly_prefix
```

A continuación, mapee sus lecturas y calcule las estadísticas de alineación.

In [ ]:
```
#!/bin/bash
#SBATCH -N 1
#SBATCH -n 16 #Number of cores
#SBATCH -t 12:00:00  #Runtime in minutes
#SBATCH -p serial_requeue,shared  #Partition to submit to
#SBATCH --mem=16000  #Memory per node in MB

readonly SINGULARITY_EXEC='singularity exec --cleanenv /n/singularity_
images/informatics/trinityrnaseq/trinityrnaseq.v2.11.0.simg'

# $1 name of your assembly (without the .fasta suffix)
# $2 comma separated list of left read file names
# $3 comma separated list of right read file names

${SINGULARITY_EXEC} bowtie2 -p 10 -q --no-unal -k 20 -x $1 -1 $2 -2 $
3  2>align_stats.txt| ${SINGULARITY_EXEC} samtools view -@10 -Sb -o bo
wtie2.bam
```

El archivo align_stats.txt proporcionará información sobre el porcentaje de pares de lectura que se asignaron de manera concordante, así como una tasa de alineación general.

** NOTA: puede usar el indicador apropiado para bibliotecas varadas (consulte la información en la sección 6 anterior sobre: los indicadores apropiados para usar. Además, dependiendo del tamaño de su conjunto de datos de lectura, es posible que desee especificar más o menos tiempo para este trabajo. Si necesita> 24 horas, asegúrese de especificar la partición compartida.

10.- Evaluación de la calidad del ensamblaje, paso 3: cuantificación de la completitud

Otra métrica de la calidad del ensamblaje es evaluar hasta qué punto recupera ortólogos de copia única que están presentes en agrupaciones taxonómicas superiores. Si bien en ausencia de saber qué transcripciones se expresan realmente en una muestra, es difícil determinar una expectativa absoluta de recuperación de estos ortólogos, claramente, un gran número de genes clasificados como faltantes en un ensamblaje debe considerarse una posible señal de alerta. Además, los ensamblajes basados en los mismos datos leídos pueden evaluarse con respecto al número de genes que están completos, fragmentados o que faltan en el ensamblaje.

Para evaluar la integridad, utilizamos BUSCO. BUSCO requiere que especifique un conjunto de datos BUSCO (Benchmarking Universal Single-Copy Orthologs). Los conjuntos de datos se encuentran en el clúster de Cannon en / n / holyscratch01 / external_repos / INFORMATICS / BUSCO /. Contáctenos si la base de datos que necesita no se encuentra actualmente en ese directorio. BUSCO envuelve HMMER y usa perfiles del modelo de Markov oculto para determinar si los contigs de ensamblaje son orthologus con una entrada particular del conjunto de datos de BUSCO. Le recomendamos que cree un entorno conda que contenga BUSCO y luego lo ejecute utilizando ese entorno. Primero, crea el entorno iniciando un trabajo interactivo (por ejemplo, srun --pty -p shared -t 01:00:00 --mem 1000 / bin / bash), luego haga:

In [ ]:
```
module load python
conda create –n busco –c bioconda busco
```

Una vez que se ha construido el entorno, simplemente actívelo en su script de trabajo BUSCO:

In [ ]:
```
#!/bin/bash
#SBATCH -N 1
#SBATCH -n 16
#SBATCH -p serial_requeue,shared
#SBATCH -e BUSCO.err           # File to which STDERR will be written
#SBATCH -o BUSCO.out         # File to which STDOUT will be written
#SBATCH -J BUSCO                # Job name
#SBATCH --mem=6000                    # Memory requested
#SBATCH --time=04:00:00               # Runtime in HH:MM:SS
#SBATCH --mail-type=ALL               # Type of email notification- B
EGIN,END,FAIL,ALL
#SBATCH --mail-user=Your.Email.Address # Email to send notifications t
o

module load python
source activate busco

# $1 input fasta file (your assembly, e.g. Trinty.fasta)
# $2 output directory (BUSCO will prepend run_)
# $3 lineage directory see lineages to choose from at: http://busco.ez
lab.org/

run_BUSCO.py -c 16 -o $2 -in $1 -l $3 -m transcriptome

Example submission: sbatch BUSCO.sh Trinity.fasta trinity_BUSCO /n/hol
yscratch01/external_repos/INFORMATICS/BUSCO/eukaryota_odb9
```

Además de los directorios que contienen resultados de HMMER, proteínas traducidas y tablas de resultados de BUSCO (y BUSCO faltantes), BUSCO genera una tabla de resumen breve útil, y un ejemplo de la cual se muestra a continuación:

In [ ]:
```
BUSCO was run in mode: trans

Summarized benchmarks in BUSCO notation:
C:70%[D:24%],F:4.2%,M:25%,n:3023

Representing:
    1387    Complete Single-copy BUSCOs
    750 Complete Duplicated BUSCOs
    128 Fragmented BUSCOs
    758 Missing BUSCOs
    3023    Total BUSCO groups searched
```

Debido a que varias transcripciones pueden afectar a un BUSCO, esto puede llevar a la clasificación de BUSCO como "duplicado completo". Más informativo es el número total de BUSCO completos (tanto simples como duplicados).