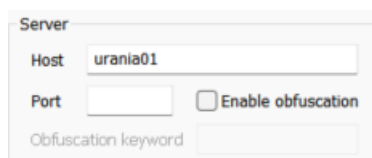


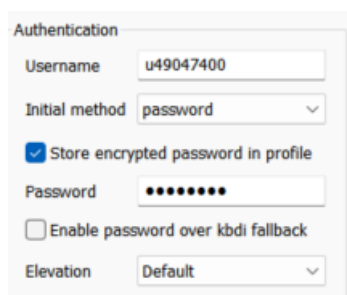
Protocolo de obtención de alineamientos de péptidos a partir de reads

Paso 1: Instalación y conexión de Bitvise SSH Client

En el apartado “Server” debemos rellenar el campo de host con “urania01”.



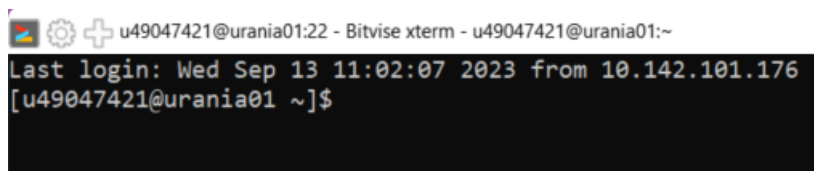
En el apartado “Authentication” debemos introducir nuestro usuario y contraseña.



Luego pulsamos el botón “Log in” para conectarnos.

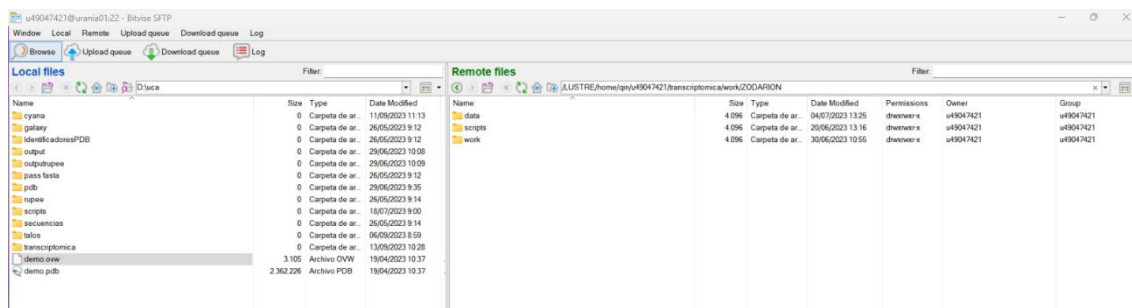
Una vez conectado, aparecen dos botones a la izquierda: “New terminal console” y “New SFTP window”.

Desde la consola podremos navegar a través del sistema de ficheros del clúster utilizando los comandos de Linux.



```
u49047421@urania01:~ - Bitvise xterm - u49047421@urania01:~
Last login: Wed Sep 13 11:02:07 2023 from 10.142.101.176
[u49047421@urania01 ~]$
```

Una opción más cómoda es utilizar el servicio SFTP, donde a la izquierda tendremos el sistema de archivos de nuestro PC y a la derecha el del clúster. Podremos navegar a través de ellos y arrastrar archivos de un lado a otro para copiarlos, crear carpetas, renombrar, etc.



Podemos modificar directamente los ficheros desde aquí abriéndolos con un editor de texto (Notepad++) o un editor de código (Visual Studio Code).

Para ejecutar los scripts debemos hacerlo desde la consola.

Utilizamos el comando “cd nombreDelDirectorio” para acceder a un directorio. Con el comando “cd ..” volvemos atrás.

Visualizamos el contenido del directorio con el comando “ls”.

Ejecutamos el script mediante el comando “sbatch nombreDelScript”.

Para comprobar el estado de la ejecución, utilizamos el comando “squeue”. Aparecerá una lista de los trabajos que se están ejecutando o están pendientes de hacerlo. Cada trabajo tiene asociado un ID.

Para cancelar la ejecución de un script debemos usar el comando “scancel” seguido del ID correspondiente.

Paso 2: Preparación del directorio de trabajo

Abre tanto la ventana SFTP como la consola, ya que se trabajará desde ambos lados.

Dentro de tu directorio personal del clúster, crea un directorio llamado “transcriptomica” si aún no lo tienes:

```
mkdir transcriptomica
```

 (También puedes crear carpetas desde la ventana SFTP)

Entramos en él:

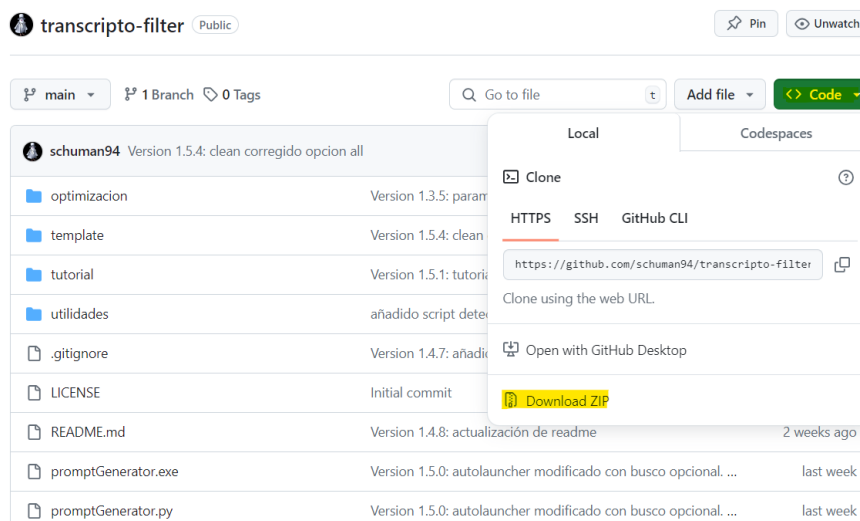
```
cd transcriptomica
```

Copia el directorio “template” con el siguiente comando:

```
cp -r /LUSTRE/home/qin/u49047421/transcriptomica/transcripto-  
filter/template .
```

Nota: el punto “.” final de la línea anterior sirve para indicar el directorio actual, es decir, a donde se copiará “template”.

Este directorio es una plantilla de trabajo actualizada y preparada para ejecutar todos los scripts. También se puede obtener el directorio “template” descargando el proyecto comprimido en GitHub: <https://github.com/schuman94/transcripto-filter>



Enlace de descarga directa:

<https://github.com/schuman94/transcripto-filter/archive/refs/heads/main.zip>

Una vez descomprimido en tu PC, se arrastra el directorio “template” al lugar de trabajo utilizando la ventana SFTP de Bitvise.

Una vez que el directorio “template” se encuentre en el clúster, se debe cambiar el nombre por otro más adecuado para diferenciarlo:

```
mv template experimento1
```

(También se puede hacer desde la interfaz gráfica pulsando la opción “rename”)

Entramos dentro:

```
cd experimento1
```



```
[u49047421@urania01 protocolo]$ mkdir transcriptimica
[u49047421@urania01 protocolo]$ cd transcriptimica/
[u49047421@urania01 transcriptimica]$ cp -r /LUSTRE/home/qin/u49047421/transcriptomica/template .
[u49047421@urania01 transcriptimica]$ ls
template
[u49047421@urania01 transcriptimica]$ mv template experimento1
[u49047421@urania01 transcriptimica]$ ls
experimento1
[u49047421@urania01 transcriptimica]$ cd experimento1/
[u49047421@urania01 experimento1]$
```

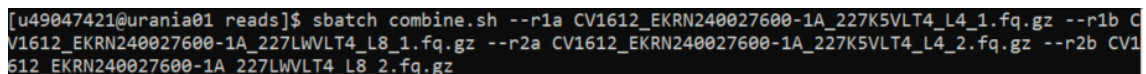
Copia al directorio “reads” los dos ficheros de partida procedentes de la secuenciación. Estos ficheros tienen un formato fastq, aunque probablemente estén comprimidos. No es necesario descomprimirlos.



Name	Size	Type	Date Modified
22ID00797_S97_R1_001.fastq.gz	4.484.405.7...	Archivo WinR...	24/05/2023 16:45
22ID00797_S97_R2_001.fastq.gz	4.653.675.7...	Archivo WinR...	24/05/2023 16:47

En algunas ocasiones es posible que existan 4 reads. Esto se debe a que la secuenciación era demasiado larga y cada read se ha dividido en dos partes, de modo que habrán dos reads para R1 y otros dos para R2.

En este caso entra en el directorio “reads” donde se encuentran los 4 ficheros y ejecuta el script “combine.sh” con el comando “sbatch” pasando como argumentos los 4 ficheros tal como se muestra en el siguiente ejemplo:



```
[u49047421@urania01 reads]$ sbatch combine.sh --r1a CV1612_EKRN240027600-1A_227K5VLT4_L4_1.fq.gz --r1b CV1612_EKRN240027600-1A_227LWVLT4_L8_1.fq.gz --r2a CV1612_EKRN240027600-1A_227K5VLT4_L4_2.fq.gz --r2b CV1612_EKRN240027600-1A_227LWVLT4_L8_2.fq.gz
```

En este ejemplo tenemos los siguientes reads:

Los reads R1 son: CV1612_EKRN240027600-1A_227K5VLT4_L4_1.fq.gz y CV1612_EKRN240027600-1A_227LWVLT4_L8_1.fq.gz

Los reads R2 son: CV1612_EKRN240027600-1A_227K5VLT4_L4_2.fq.gz y CV1612_EKRN240027600-1A_227LWVLT4_L8_2.fq.gz

Se deben indicar con --r1a, --r1b, --r2a y --r2b respectivamente.

Se debe prestar atención en los nombres de los reads para diferenciarlos, por ejemplo, en este caso lo que los diferencia "r1a" de "r1b" es la terminación "L4" y "L8" y la terminación. Si "L4" es un fichero más pesado que "L8" se entiende que es "L4" es el primero y "L8" es el que contiene el resto de la secuenciación. Es importante mantener el orden cuando se ejecuta el script.

Como resultado de la ejecución del script se obtendrán los ficheros "combined_read1.fq" y "combined_read2.fq" que podremos usar en el siguiente paso como ficheros de partida.

Paso 3: Análisis de calidad con FastQC

Entra en el directorio "fastqc" y ejecuta el script "fastqc.sh" con el comando "sbatch" pasando como argumentos los dos ficheros de partida.

```
[u49047421@urania01 fastqc]$ ls
fastqc.sh
[u49047421@urania01 fastqc]$ sbatch fastqc.sh ../reads/22ID00797_S97_R1_001.fastq.gz ../reads/22ID00797_S97_R2_001.fastq.gz
Submitted batch job 7088
[u49047421@urania01 fastqc]$
```

Con el comando "squeue" puedes comprobar el estado de la ejecución del script desde la cola de trabajo del clúster.

```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/transcriptomica/experimento1/fastqc
[u49047421@urania01 fastqc]$ squeue
      JOBID PARTITION    NAME    USER  ST       TIME  NODES NODELIST(REASON)
      7086   normal  script_W u4889375 PD        0:00      6 (BeginTime)
      7087   normal  run_UCA2 u5412258 PD        0:00      4 (Resources)
      7085   normal  DIN_Sp2 u4404262 PD        0:00      2 (Priority)
      7068   normal  lan_ts  u7579375 R       52:05      1 urc20
      7088   normal  fastqcRu u4904742 R        2:10      1 urc10
      6661   normal      cp u4956265 R  2-07:30:28      4 urc[04-07]
      6828   normal  irredsea u2648176 R  4-12:53:15      1 urc10
      7084   normal  DIN_Sp2 u4404262 R       51:53      2 urc[02-03]
[u49047421@urania01 fastqc]$
```

Para comprobar si la ejecución ha terminado, además de consultar la cola de trabajo del clúster, puedes visualizar con el comando "less" el fichero "fastqc.out" que se ha generado.

```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/
[u49047421@urania01 fastqc]$ ls
fastqc.err fastqc.out fastqc.sh output
[u49047421@urania01 fastqc]$ less fastqc.out
```

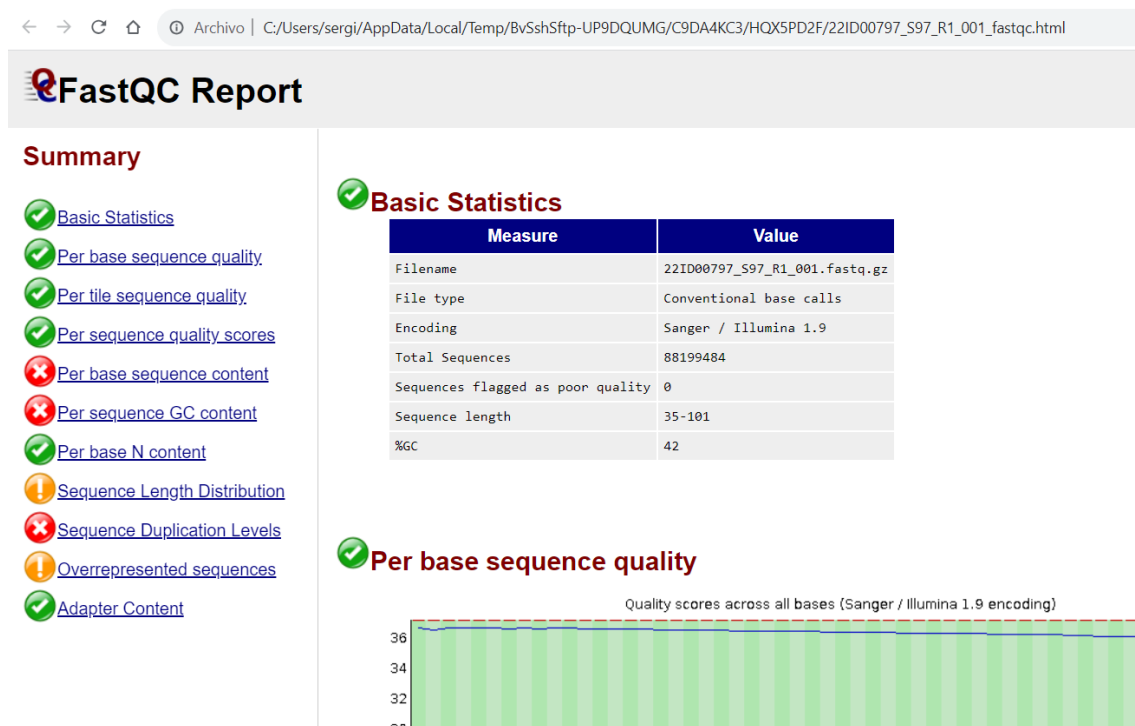
```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/transcriptomica/experimento1/fastqc
Iniciando el script en: vie sep 15 09:13:17 CEST 2023
Analysis complete for 22ID00797_S97_R1_001.fastq.gz
Analysis complete for 22ID00797_S97_R2_001.fastq.gz
Ejecucion finalizada en: vie sep 15 09:30:22 CEST 2023
fastqc.out (END)
```

Para salir pulsa la tecla q.

En el directorio “output” generado aparecerán los ficheros con el análisis de calidad de los reads.

```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/transcriptomica/experimento1
[u49047421@urania01 fastqc]$ ls
fastqc.err fastqc.out fastqc.sh output
[u49047421@urania01 fastqc]$ cd output/
[u49047421@urania01 output]$ ls
22ID00797_S97_R1_001_fastqc.html 22ID00797_S97_R2_001_fastqc.html
22ID00797_S97_R1_001_fastqc.zip 22ID00797_S97_R2_001_fastqc.zip
[u49047421@urania01 output]$
```

Puedes descargarlos desde la ventana SFTP o abrirlos directamente.



Paso 4: Ensamblaje con Trinity

Una vez comprobada la calidad de los reads, el siguiente paso es el ensamblaje para obtener un fichero fasta con todas las secuencias.

Entramos en el directorio “trinity” y ejecutamos con el comando “sbatch” el script “trinity.sh” pasándole como argumentos los dos ficheros de partida del directorio “reads”.

```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/transcriptomica/experimento1/trinity
[u49047421@urania01 trinity]$ sbatch trinity.sh ../reads/22ID00797_S97_R1_001.fastq.gz ../reads/22ID00797_S97_R2_001.fastq.gz
Submitted batch job 7090
[u49047421@urania01 trinity]$
```

Se generarán los ficheros “trinity.err” y “trinity.out” que podrás visualizar para hacer un seguimiento de la ejecución.

Al finalizar la ejecución satisfactoriamente se creará un fichero con los resultados llamado “trinity_out_dir.Trinity.fasta”.

El script de Trinity tiene una restricción de tiempo de 7 días de ejecución. Por lo general, es suficiente para realizar el ensamblado. Sin embargo, si fuese necesario ensamblar un transcriptoma excesivamente grande, será necesario modificar el script para aumentar el límite de tiempo.

Paso 5: Control de calidad del ensamblaje con BUSCO

Para ejecutar BUSCO desde un ambiente offline necesitamos una base de datos de linaje como referencia. En nuestro caso: metazoa_odb10.

Puedes descargarla manualmente desde <https://busco-data.ezlab.org/v5/data/lineages/> y guardarla en un nuevo directorio dentro de “transcriptomica”.

Por ejemplo, en “transcriptomica/data/BUSCO_DB/ metazoa_odb10”

Este fichero normalmente será común para todos los experimentos. Existe una copia en:

/LUSTRE/home/qin/u49047421/transcriptomica/data/BUSCO_DB/metazoa_odb10

Esta copia ya está referenciada dentro del script “busco.sh”. En caso de querer utilizar otra, es necesario modificar el script.

Accede al directorio “busco” y dentro de él ejecuta con “sbatch” el script “busco.sh” pasando como argumento el fichero fasta obtenido con Trinity.

```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/transcriptomica/experimento1/busco
[u49047421@urania01 busco]$ sbatch busco.sh ../trinity/trinity_out_dir.Trinity.fasta
Submitted batch job 7092
[u49047421@urania01 busco]$
```

Tras la ejecución se creará un directorio llamado “busco_output”.

```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/transcriptomica/experimento1/busco/
[u49047421@urania01 busco]$ ls
busco_downloads  busco.err  busco.out  busco_output  busco.sh
[u49047421@urania01 busco]$ cd busco_output/
[u49047421@urania01 busco_output]$ ls
logs                short_summary.specific.metazoa_odb10.busco_output.json
run_metazoa_odb10  short_summary.specific.metazoa_odb10.busco_output.txt
[u49047421@urania01 busco_output]$
```

Dentro de él podemos consultar el *short summary* con el comando “less” para visualizar los resultados.

```
***** Results: *****

C:76.9%[S:56.1%,D:20.8%],F:10.1%,M:13.0%,n:954
733    Complete BUSCOs (C)
535    Complete and single-copy BUSCOs (S)
198    Complete and duplicated BUSCOs (D)
96     Fragmented BUSCOs (F)
125    Missing BUSCOs (M)
954    Total BUSCO groups searched
```

Paso 6: BLASTX

El siguiente paso es utilizar Blastx para comparar las secuencias con una base de datos de péptidos.

Si se trata de una nueva base de datos, primero necesitamos construirla. Para ello necesitamos guardar el fichero fasta con la base de datos en un directorio común para todos los experimentos.

Por ejemplo, si vamos a utilizar el fichero “CTX-May23bis_completeseq.fasta” como base de datos. Podemos guardarlo en:

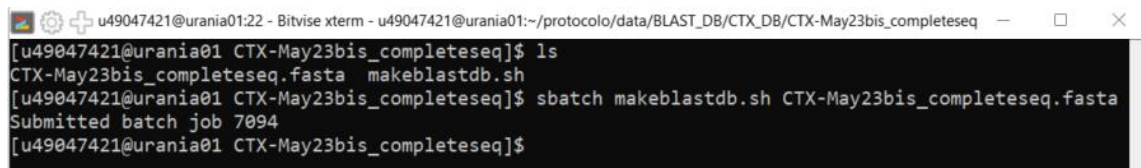
transcriptomica/data/BLAST_DB/CTX_DB/CTX-May23bis_completeseq/ CTX-May23bis_completeseq.fasta

Para construir la base de datos copia el script “makeblastdb.sh” que se encuentra en el directorio “blastx” al directorio donde se encuentra el fichero fasta.



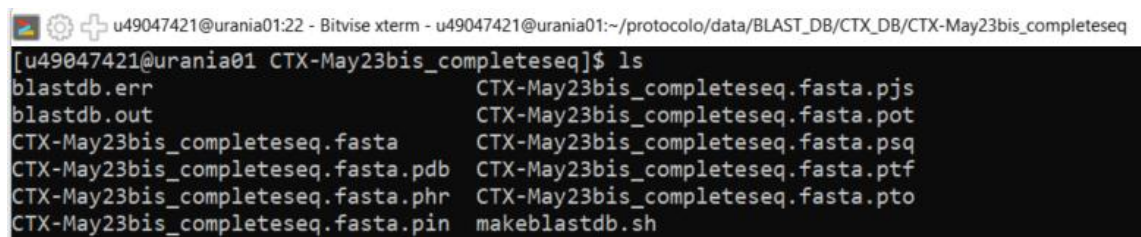
```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/transcriptomica/experimento1/blastx
[ u49047421@urania01 blastx ]$ ls
blastx.sh  makeblastdb.sh
[ u49047421@urania01 blastx ]$ cp makeblastdb.sh ../../data/BLAST_DB/CTX_DB/CTX-May23bis_completeseq/
[ u49047421@urania01 blastx ]$
```

Una vez copiado y situado dentro del directorio, ejecuta el script con “sbatch” pasándole el fichero fasta como argumento.



```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/data/BLAST_DB/CTX_DB/CTX-May23bis_completeseq
[ u49047421@urania01 CTX-May23bis_completeseq ]$ ls
CTX-May23bis_completeseq.fasta  makeblastdb.sh
[ u49047421@urania01 CTX-May23bis_completeseq ]$ sbatch makeblastdb.sh CTX-May23bis_completeseq.fasta
Submitted batch job 7094
[ u49047421@urania01 CTX-May23bis_completeseq ]$
```

Se generarán diferentes ficheros con los que Blastx trabajará automáticamente.



```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/data/BLAST_DB/CTX_DB/CTX-May23bis_completeseq
[ u49047421@urania01 CTX-May23bis_completeseq ]$ ls
blastdb.err          CTX-May23bis_completeseq.fasta.pjs
blastdb.out          CTX-May23bis_completeseq.fasta.pot
CTX-May23bis_completeseq.fasta  CTX-May23bis_completeseq.fasta.psq
CTX-May23bis_completeseq.fasta.pdb  CTX-May23bis_completeseq.fasta.ptf
CTX-May23bis_completeseq.fasta.phr  CTX-May23bis_completeseq.fasta.pto
CTX-May23bis_completeseq.fasta.pin  makeblastdb.sh
```

Una vez que ya disponemos de una base de datos construida, podemos entrar en el directorio “blastx” y ejecutar el script “blastx.sh” con el comando “sbatch”. Es necesario pasarle 2 argumentos en el orden correcto.

1. QUERY: Fichero fasta obtenido en Trinity.
2. DB: Fichero fasta de la base de datos de péptidos previamente construida. La base de datos mencionada anteriormente ya se encuentra construida en:
/LUSTRE/home/qin/u49047421/transcriptomica/data/BLAST_DB/CTX_DB/CTX-May23bis_completeseq/CTX-May23bis_completeseq.fasta

```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/transcriptomica/experimento1/blastx
[u49047421@urania01 blastx]$ sbatch blastx.sh ../trinity/trinity_out_dir.Trinity.fasta /LUSTRE/home/qin/u49047421/transcriptomica/data/BLAST_DB/CTX_DB/CTX-May23bis_completeseq/CTX-May23bis_completeseq.fasta
Submitted batch job 7096
[u49047421@urania01 blastx]$
```

Tras la ejecución se generará un fichero “blastx_out.csv”.

Paso 7: Extracción de alineamientos

Accede al directorio “alignments” y ejecuta con “sbatch” el script “alignments.sh”. Es necesario pasarle 3 argumentos en el orden correcto.

1. BLAST_CSV: Fichero csv obtenido en Blastx.
2. TRINITY_FASTA: Fichero fasta obtenido en Trinity.
3. DB: Fichero fasta de la base de datos de péptidos previamente construida.

```
u49047421@urania01:22 - Bitvise xterm - u49047421@urania01:~/protocolo/transcriptomica/experimento1/alignments
[u49047421@urania01 alignments]$ sbatch alignments.sh ../blastx/blastx_out.csv ../trinity/trinity_out_dir.Trinity.fasta /LUSTRE/home/qin/u49047421/transcriptomica/data/BLAST_DB/CTX_DB/CTX-May23bis_completeseq/CTX-May23bis_completeseq.fasta
Submitted batch job 7098
[u49047421@urania01 alignments]$
```

Para verificar que la ejecución ha finalizado correctamente consulta el fichero “alignments.out”.

Se creará un directorio llamado “Alineamientos_mafft” en donde se encontrarán todos los ficheros con la extensión “mafft.fasta” que contendrán los alineamientos detectados.

También puede ser útil el fichero “extracted_sequences.fasta”.

Descarga los resultados desde la ventana SFTP a tu PC para continuar trabajando con los alineamientos detectados.

Paso 8: Filtrado de los alineamientos

Accede al directorio “curation_filter”. Antes de ejecutar el script es necesario revisar los parámetros de configuración del filtro. Éstos se encuentran en el fichero “config.txt” dentro del directorio “python_scripts”. Para modificar la configuración puedes editar el fichero o sustituirlo por otro con el mismo nombre y formato.

Una vez revisado los parámetros, ejecuta con “sbatch” el script “filter.sh”, no es necesario indicar ningún argumento. La ejecución puede durar alrededor de 20 minutos. Tras finalizar accede al directorio “Alineamientos_filtrados”.

Dentro de este directorio se encuentran todos los alineamientos que han pasado el filtro y un fichero llamado “informe.tsv”. Puedes descargar este fichero para consultar toda la información del filtrado.

El fichero contiene las siguientes columnas:

- **Seq_file:** El número del fichero original mafft.fasta analizado.
- **Frame_ID:** El identificador del marco de lectura analizado.
- **Ref_ID:** El identificador de la secuencia de la base de datos con la que se ha intentado alinear el marco de lectura.

- **Pasa_filtro:** Indica si el alineamiento entre las dos secuencias comparadas ha pasado el filtro.
- **Hay_segmento_de_subsecuencias_validas:** Criterio de filtrado 1 (debe ser “True”). Hace referencia a que existen partea de las dos secuencias que alinean correctamente.
- **longitud_minima_total_subseqs_superada:** Criterio de filtrado 2 (debe ser “True”). Hace referencia a la cantidad mínima de aminoácidos que deben alinearse.
- **ratio_minimo_longitud_superado:** Criterio de filtrado 3 (debe ser “True”). Hace referencia al porcentaje mínimo que la parte alineada debe abarcar en la secuencia de la base de datos.
- **Stop_codon_en_mitad_de_dos_segmentos_subsecuencias_validas:** Criterio de filtrado 4 (debe ser “False”). Hace referencia a la presencia de un codón de parada en mitad de la secuencia analizada.
- **Vector_alineamiento:** Representación binaria del alineamiento entre las dos secuencias: 1(aa alineado), 0 (aa no alineado), * (codón de parada).

Paso 9: Filtro de Metionina y clasificación de alineamientos

Accede al directorio “metionine_filter” y ejecuta con “sbatch” el script “filter.sh”

Una vez finalizada la ejecución, en el directorio “resultados” se encontrarán los alineamientos clasificados.

- **Alineamientos_Perfectos:** La metionina inicial en las secuencias de referencia coincide con la metionina inicial del frame.
- **Alineamientos_Limpios:** Se trata de los alineamientos perfectos ya formateados sin guiones, comenzando por la metionina inicial y finalizando en el codón de stop correspondiente.
- **Alineamientos_M_Previa:** La metionina inicial en las secuencias de referencia coincide con la metionina inicial del frame, pero existe una metionina anterior que podría ser el verdadero inicio de la secuencia.
- **Alineamientos_Multiframe:** Son los ficheros en donde más de un frame alinea con las secuencias de referencia.
- **Alineamientos_Revision_Manual:** Necesitan ser revisados manualmente.

Paso 10: Clasificación por superfamilias de conotoxinas y otros péptidos

Accede al directorio “superfamily” y ejecuta con “sbatch” el script “SF-filter.sh”. Es necesario pasarle 3 argumentos con las opciones --db1, --db2 y --db3 tal como se puede ver en la imagen de ejemplo.

1. --db1: Fichero fasta de la base de datos de péptidos que hemos usado en el paso 6.
2. --db2: Fichero fasta de la base de datos de péptidos señales de conotoxinas.
3. --db3: Fichero fasta de la base de datos de superfamilias de conotoxinas.

```
[u49047421@urania01 superfamily_16-12]$ sbatch SF-filter.sh --db1 /LUSTRE/home/qor/u31332563/transcr
a --db2 /LUSTRE/home/qor/u31332563/transcriptomica/data/BLAST_DB/CTX_DB/CTX_SF_signal/CTX_SF_signal.
DB/VirroDB_all/VirroDB_all.fasta
```

Una vez finalizada la ejecución tendremos principalmente dos directorios de interés.

En el directorio “resultados” tendremos todos los ficheros csv con las secuencias clasificadas e indicando su mejor match (blast) con la base de datos de superfamilias.

En el directorio alineamientos_NoSF tendremos todos los ficheros mafft.fasta con las secuencias alineadas frente a todos sus match (blast) con la base de datos de superfamilias.

También se habrán creado ficheros fasta en el directorio buscar_alineamientos_en_mfilter en donde se aparecen todas las secuencias que no han hecho match con la base de datos de superfamilias, pero sí tenían matches con el primer blastx que se hizo en el paso 6.

Paso 11: Cálculo de los niveles de expresión

Con este paso se añadirá dos nuevas columnas a los ficheros csv de las secuencias en donde se indica los niveles de expresión (tpm y numreads).

Este paso solo es posible realizarse en caso de que se haya partido desde los reads y se haya realizado su ensamblaje con Trinity, ya que la información de los niveles de expresión estará contenida en los archivos de salida de dicho ensamblaje.

Accede al directorio “quantification” y ejecuta con “sbatch” el script “results.sh”.

Una vez terminada la ejecución, aparecerá un directorio llamado “resultados”. Lee el paso 12 para continuar.

Nota: en caso de que hayamos partido de un fichero previamente ensamblado con Trinity pero también tengamos disponibles los reads con los que se ensamblaron, es posible realizar manualmente el cálculo de los niveles de expresión. Para ello ejecuta con “sbatch” el script “quantify.sh”.

Es necesario indicar 3 argumentos:

```
--r1 ruta/del/read1  
--r2 ruta/del/read2  
--trinity ruta/del/fichero/ensamblado
```

```
[u49047421@urania01 quantification]$ sbatch quantify.sh --r1 ../reads/22ID00802_S102_R1_001.fastq.gz  
--r2 ../reads/22ID00802_S102_R2_001.fastq.gz --trinity ../trinity/trinity_out_dir.Trinity.fasta
```

De esta forma primero se realizará el cálculo de los niveles de expresión y luego se añadirá en una nueva columna a los ficheros csv que se crearán en el directorio de resultados.

Paso 12: Obtención de los resultados

Debemos descargarnos el directorio “resultados” (ya sea el obtenido en el paso 10 u 11). Esto se puede hacer arrastrando el directorio a nuestro PC desde la interfaz gráfica de Bitvise.

Utilizando en nuestro PC la herramienta “resultadosToExcel.exe” podemos seleccionar ese directorio para crear un fichero Excel con todos los resultados ordenados y clasificados.

Nota: este script se encuentra en el GitHub de transcripto-filter:

<https://github.com/schuman94/transcripto-filter/tree/main/resultadosToExcel>

Nota: Esta herramienta también se encuentra como script de Python (resultadosToExcel.py). Si trabajamos en Linux, podemos ejecutarlo desde la terminal si disponemos de todas las dependencias necesarias.

Una vez ejecutado, obtendremos un fichero Excel con todos los resultados en diferentes hojas.

- **Perfectos_SF**: Lista de secuencias (del directorio “Alineamientos_Perfectos”) en donde se ha identificado un péptido señal de la base de datos de péptidos señal y posteriormente ha alineado con la base de datos de superfamilias. En la columna “SF_signal” se indica al péptido señal con el que alinea con un umbral del 70% de coincidencia (pident_signal). En la columna “match” nos aparece el mejor resultado y en la columna “SF-hormone” la superfamilia a la que pertenece.
- **Perfectos_NoSF**: Lista de secuencias (del directorio “Alineamientos_Perfectos”) en donde no se ha identificado un péptido señal de la base de datos de péptidos señal pero posteriormente ha alineado con la base de datos de superfamilias.
- **Perfectos_noMatch**: Lista de secuencias (del directorio “Alineamientos_Perfectos”) que no han alineado con ninguna secuencia de la base de datos de superfamilias.
- **Mprevia_SF**: Lista de secuencias (del directorio “Alineamientos_M_Previa”) en donde se ha identificado un péptido señal de la base de datos de péptidos señal y posteriormente ha alineado con la base de datos de superfamilias.
- **Mprevia_NoSF**: Lista de secuencias (del directorio “Alineamientos_M_Previa”) en donde no se ha identificado un péptido señal de la base de datos de péptidos señal pero posteriormente ha alineado con la base de datos de superfamilias.
- **Mprevia_noMatch**: Lista de secuencias (del directorio “Alineamientos_M_Previa”) que no han alineado con ninguna secuencia de la base de datos de superfamilias.
- **Revision_manual_Match**: Lista de secuencias (del directorio “Alineamientos_Revision_Manual”) que han alineado con la base de datos de superfamilias.
- **Revision_manual_noMatch**: Lista de secuencias (del directorio “Alineamientos_Revision_Manual”) que no han alineado con la base de datos de superfamilias.

El directorio “alineamientos_NoSF” dentro de “superfamily” (paso 10) contiene los alineamientos en formato “mafft.fasta” de las secuencias de “Perfectos_NoSF”, “Mprevia_NoSF” y “Revision_manual_Match” frente a todas las secuencias de la base de datos de superfamilias con las que ha habido un match. En estos alineamientos se puede comprobar en la descripción de cada secuencia a qué superfamilia pertenecen.

Tras la ejecución de este script también podemos realizar la misma comprobación de las descripciones en los alineamientos “mafft.fasta” del directorio “metionine_filter” del paso 9.

Esta revisión será útil para las secuencias que no han obtenido un match con la base de datos de superfamilias (noMatch) pero sí lo obtuvieron con la base de datos utilizada en el paso 6. Como se ha comentado al final del paso 10, también se puede realizar la comprobación en el directorio output de superfamily/buscar_alineamientos_en_mfilter.

Modo automático: Ejecución de autolauncher

Todo el flujo de trabajo descrito anteriormente se puede realizar de forma automática. Existen dos puntos de partida:

- **Punto de partida inicial**: Indicando los dos **reads** y la **base de datos de referencia**.
- **Punto de partida después del ensamblaje**: Indicando el **fichero fasta ensamblado** con Trinity y la **base de datos de referencia**.

Para trabajar con el modo automático se realizan los pasos 1, 2 y 3 descritos anteriormente para preparar nuestro directorio de trabajo con los dos reads.

Si disponemos del fichero fasta ensamblado debemos copiarlo al directorio “trinity”.

Una vez tengamos el directorio de trabajo preparado, nos situamos en él para ejecutar “`sbatch autolauncher.sh`” seguido de las opciones con los argumentos necesarios.

Las opciones son las siguientes:

```
--r1 ruta/del/read1
--r2 ruta/del/read2
--db1 ruta/de/la/base/de/datos
--trinity ruta/del/fichero/ensamblado
--db2 ruta/de/la/base/de/datos/de/péptidos/señal
--db3 ruta/de/la/base/de/datos/de/superfamilias
--fastqc true/false
--busco true/false
```

Las opciones fastqc y busco son opcionales. Si no se indican, se ejecutarán por defecto.

Si no se parte desde los reads, no se realizará el cálculo de los niveles de expresión.

Las opciones db2 y db3 también son opcionales. Debemos incluirlos si queremos que se haga la clasificación por superfamilias. En ese caso, también debemos descargarnos el directorio de resultados ya se dentro de “superfamily” o dentro de “quantification” si se ha realizado el cálculo de los niveles de expresión.

Una vez tengamos el directorio en nuestro PC, se debe ejecutar la herramienta “resultadosToExcel.exe”.

En caso de que no se hayan indicado las opciones db2 y db3, el filtro acabará con la ejecución del paso 9 y deberemos revisar los alineamientos obtenidos en “metionine_filter”. Nota: tampoco se realizará el cálculo de los niveles de expresión.

Para facilitar la escritura del comando sbatch, disponemos de una aplicación ejecutable en Windows llamada “promptGenerator.exe” en la que podemos introducir las rutas de los ficheros y marcar las diferentes casillas de opciones.

Generador de comandos para Transcripto-filter

Ruta del read1 (r1):

Ruta del read2 (r2):

Ruta de la base de datos general (db1):

Ruta del fichero fasta ensamblado con Trinity (trinity):

Ruta de la base de datos de péptidos señal (db2):

Ruta de la base de datos de superfamilias (db3):

☒ Ejecutar FASTQC

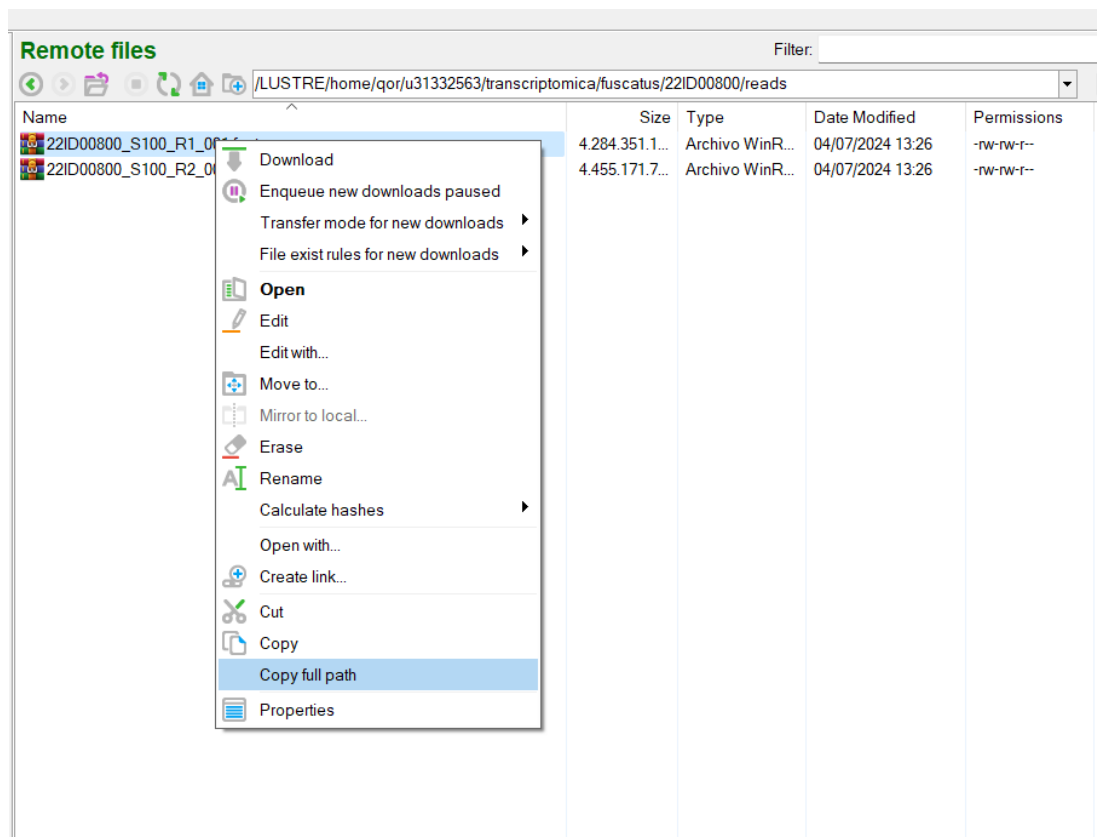
☒ Ejecutar BUSCO

☒ Incluir el análisis de superfamilias

☐ Usar fichero fasta previamente ensamblado con Trinity

Generar comando

Para introducir correctamente las rutas de los ficheros en los campos de texto, podemos utilizar la función “copy full path” en el explorador de archivos de Bitvise.



Pegamos las rutas con Ctrl+v en los campos de texto correspondientes y pulsamos el botón “Generar comando”.

Generador de comandos para Transcripto-filter

Ruta del read1 (r1): /LUSTRE/home/qor/u31332563/transcriptomica/fuscatus/22ID00800/reads/22ID00800_S100_R1_001.fastq.gz

Ruta del read2 (r2): /LUSTRE/home/qor/u31332563/transcriptomica/fuscatus/22ID00800/reads/22ID00800_S100_R2_001.fastq.gz

Ruta de la base de datos general (db1): E/home/qor/u31332563/transcriptomica/data/BLAST_DB/CTX_DB/CTX-Oct23_cleanseq/CTX-Oct23_cleanseq.fasta

Ruta del fichero fasta ensamblado con Trinity (trinity):

Ruta de la base de datos de péptidos señal (db2): /LUSTRE/home/qor/u31332563/transcriptomica/data/BLAST_DB/CTX_DB/CTX_SF_signal/CTX_SF_signal.fasta

Ruta de la base de datos de superfamilias (db3): /home/qor/u31332563/transcriptomica/data/BLAST_DB/CTX_DB/conotoxinas-virroDB/conotoxinas-virroDB.fasta

☒ Ejecutar FASTQC

☒ Ejecutar BUSCO

☒ Incluir el análisis de superfamilias

☐ Usar fichero fasta previamente ensamblado con Trinity

Generar comando

Luego, dentro de la consola de comandos y situados en el directorio de trabajo donde se encuentra “autolauncher.sh”, pulsamos click derecho con el ratón para que el comando se copie en la consola y pulsamos enter para ejecutarlo.

Ejemplo de comando:

```
[u49047421@urania01 MAU123A]$ sbatch autolauncher.sh --r1 /LUSTRE/home/qor/u31332563/transcriptomica/fuscatus/22ID00800/reads/22ID00800_S100_R1_001.fastq.gz --r2 /LUSTRE/home/qor/u31332563/transcriptomica/fuscatus/22ID00800/reads/22ID00800_S100_R2_001.fastq.gz --db1 /LUSTRE/home/qor/u31332563/transcriptomica/data/BLAST_DB/CTX_DB/CTX-Oct23_cleanseq/CTX-Oct23_cleanseq.fasta --db2 /LUSTRE/home/qor/u31332563/transcriptomica/data/BLAST_DB/CTX_DB/CTX_SF_signal/CTX_SF_signal.fasta --db3 /LUSTRE/home/qor/u31332563/transcriptomica/data/BLAST_DB/CTX_DB/conotoxinas-virroDB/conotoxinas-virroDB.fasta
```

Nota: Los ficheros “t-filter.err” y “t-filter.out” generados contienen detalles de la ejecución del script.

Una vez ejecutado podremos descargarnos el directorio “resultados” correspondiente para su transformación con la herramienta “resultadosToExcel”.

Limpiar el directorio de trabajo

Si durante la ejecución de autolauncher ha ocurrido algún error, no se podrá volver a ejecutar autolauncher con seguridad debido a la posible interferencia con archivos intermediarios creados en la primera ejecución.

Para continuar debemos utilizar primero el script clean.sh. Éste se encarga de limpiar el directorio. Para ello se ejecutará el comando:

- `sbatch clean.sh` (opción por defecto: elimina todos los ficheros creados durante la ejecución excepto los reads y los resultados de FASTQC, TRINITY y BUSCO)
- `sbatch clean.sh --all` (con esta opción elimina TODO excepto los reads. Puede tardar más de media hora si necesita eliminar el contenido generado por Trinity)

Otra opción es crear un nuevo directorio de trabajo a partir del template e incluir los reads o el fichero fasta ensamblado del antiguo directorio. Una vez rescatado del antiguo directorio los ficheros necesarios, ya podemos eliminarlo con el comando: `rm -r nombreDirectorio`

Nota: Para ejecutar este comando debemos situarnos fuera del directorio, es decir, si antes ejecutamos “ls” deberemos ver como dicho directorio aparece en la lista. Una vez eliminado, ya podemos volver a usar ese nombre para renombrar el nuevo template.

Paso extra: Alineamiento de secuencias de la base de datos final

Si se necesita volver a alinear un grupo de secuencias procedentes del fichero txt de la base de datos final, podemos seleccionarlas y copiarlas a un nuevo fichero de texto.

El fichero de texto se debe subir al directorio “mafftFromTxt” en el supercomputador.

Una vez dentro de ella, ejecuta `sbatch mafftFromTxt.sh` seguido del nombre del fichero de texto.

En cuestión de segundos se creará un nuevo fichero de texto con las secuencias alineadas en el formato adecuado para seleccionarlas y volverlas a copiar a la base de datos.