

# 220322 - Control de calidad + ensamble de secuencias SARS-CoV-2

## Instalación conda

Es necesario tener conda para poder correr este programa.

```
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
chmod +x Miniconda3-latest-Linux-x86_64.sh
./Miniconda3-latest-Linux-x86_64.sh
```

## Instalación del ensamblador de SARS-CoV-2

Una vez tengas conda instalado, crea el ambiente del ensamblador de covid. Es recomendable crear un ambiente de conda único para el ensamblador para que no entre en conflicto con programas locales de la computadora.

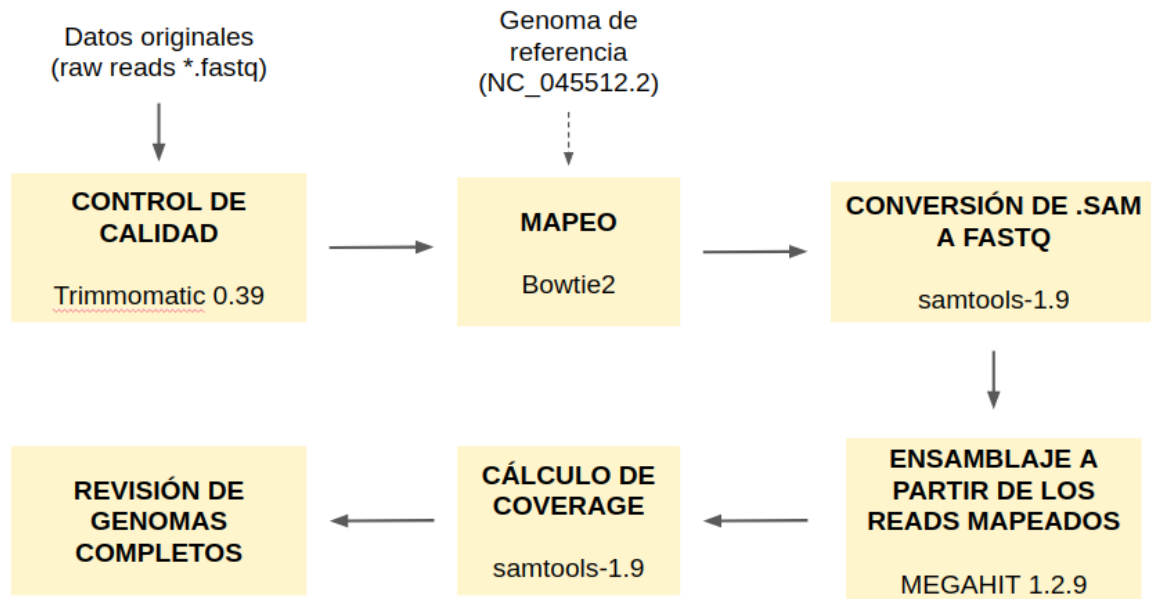
```
git clone https://github.com/leonardo467/Ensamblaje-SARS-CoV-2.git
conda env create -n covid -f environment_covid.yaml
rm environment_covid.yaml
chmod +x control_calidad.sh
```

## Activar el ambiente de conda

Ahora que ya tenemos creado el ambiente de conda, lo activamos con el siguiente comando:

```
conda activate covid
```

## Workflow bioinformático del tutorial



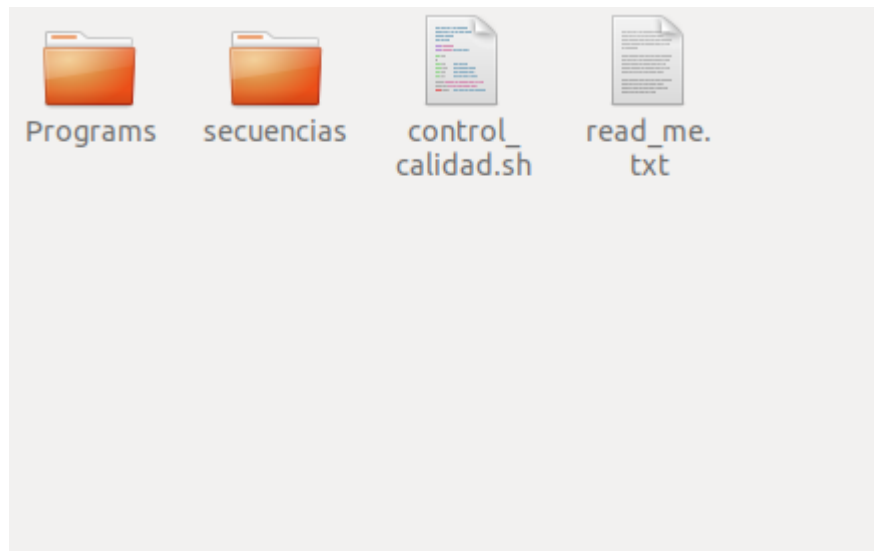
Las secuencias crudas pasan por un filtro calidad (Se eliminan secuencias que en promedio no tengan un puntaje de calidad de Q20 como mínimo) con el programa Trimmomatic. Luego corremos Bowtie2 para quedarnos nos con las secuencias que hagan match con el genoma de referencia de Sars-CoV-2 (para no ensamblar secuencias que no sean de virus). Corremos Samtools para convertir los genomas alineados (\*.sam) a fastq. Ensamblamos los fastqs filtrados con MEGAHIT y calculamos el coverage del ensamblado con samtools. Al final el script cuenta cuantos contigs tiene cada genoma, para saber si estamos frente a un genoma completo o no. Un genoma completo debería contar un solo contig para ser considerado completo.

### Nota:

- El script asume que tienes un procesador de 4 núcleos. Puedes modificar este valor entrando al archivo "control\_calidad.sh" y cambiando el valor de la variable "threads". También asume que usaste adaptadores truseq para tu secuenciamiento. Puedes cambiar estos adaptadores por los de tu interés agregándolos en la carpeta "Programs" y escribiendo su nombre exacto en la variable "adapters"

### Inicio de tutorial

1. Deposita tus secuencias (paired-end) en la carpeta "secuencias". Es importante que el nombre sea "secuencias" para que pueda correr el código. Puedes descargar [esta carpeta de secuencias de prueba](#) para probar el script. Tu espacio de trabajo se debe ver algo así:



## 2. Activa el ambiente conda

```
conda activate covid
```

## 3. Ejecutar el programa

```
./control_calidad.sh
```

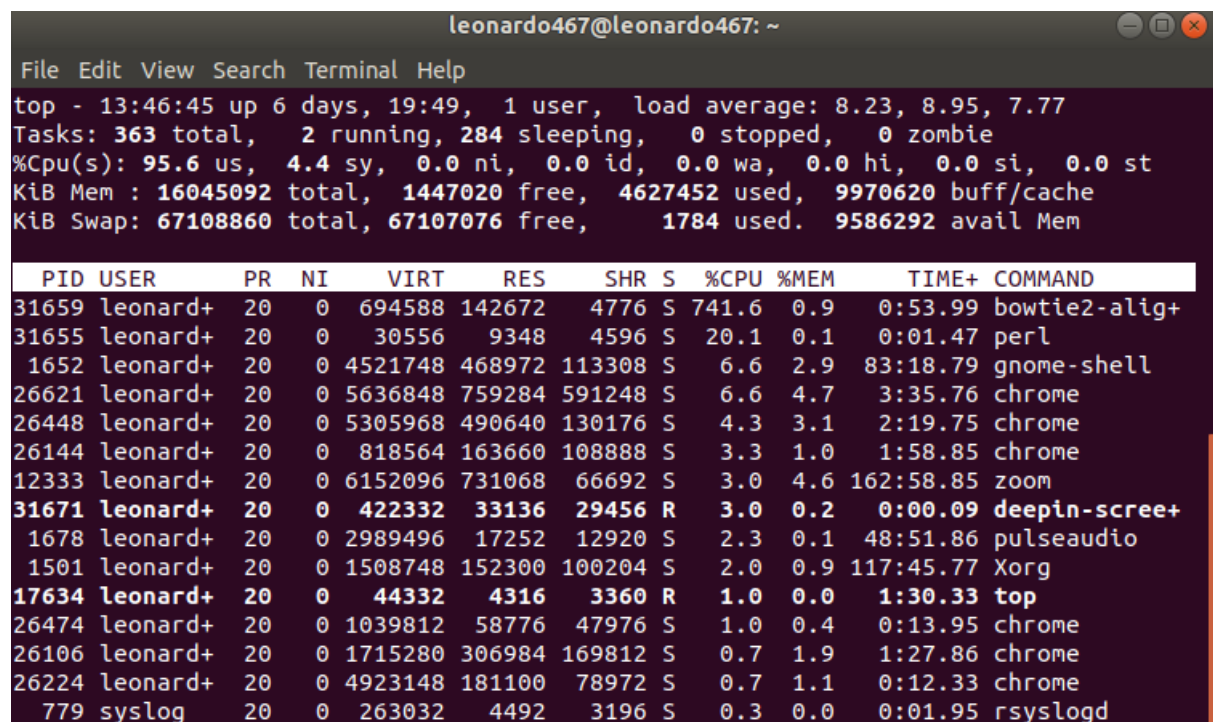
En caso que quieras guardar el log del análisis bioinformático, ejecutar el script usando el siguiente comando:

```
./control_calidad.sh 2>&1 | tee > run.log
```

Si te interesa saber cuantos recursos está usando la computadora, ejecutar el siguiente comando (en una ventana de terminal aparte):

```
top
```

Aparecerá esta pantalla:



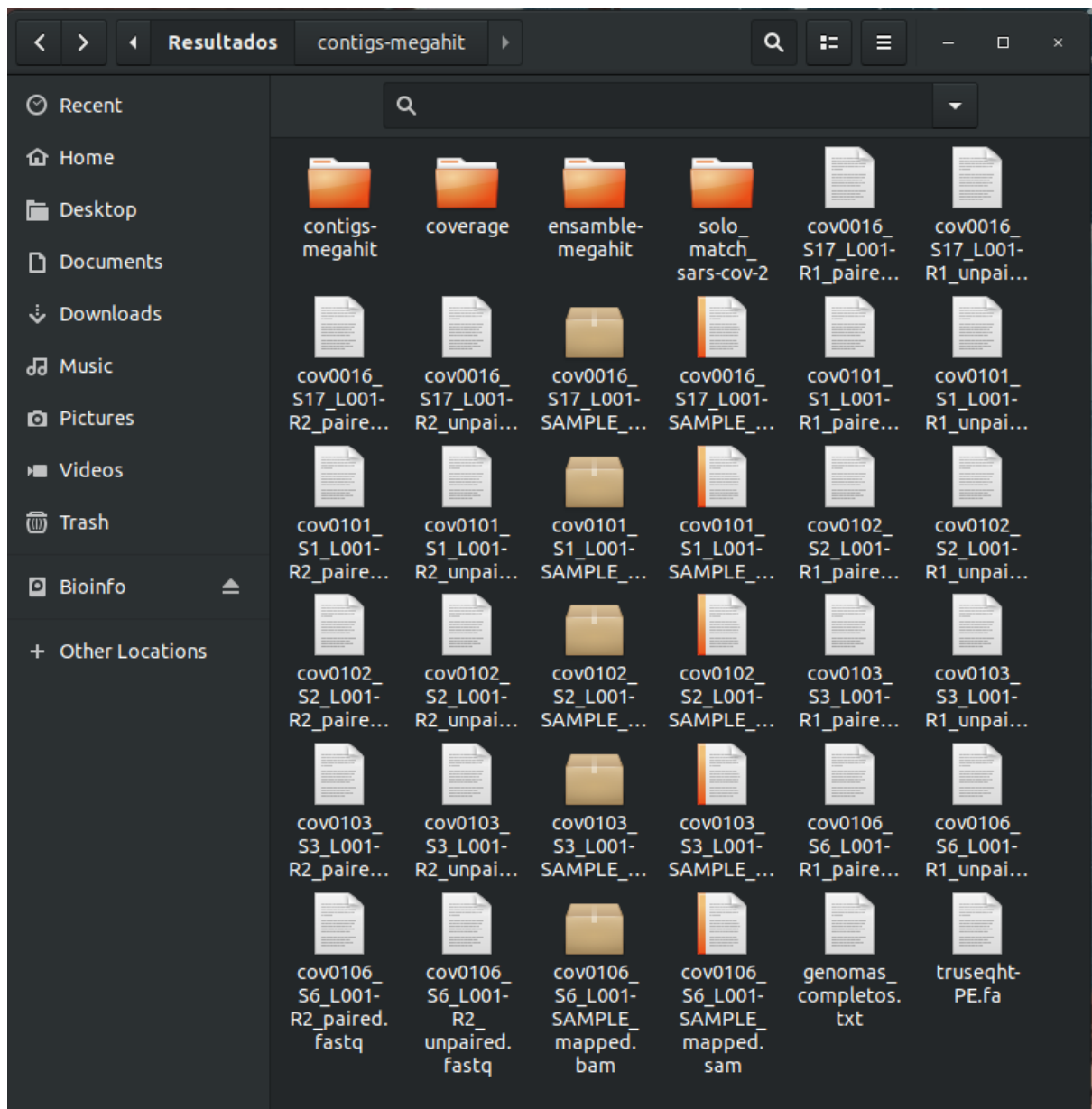
```
leonardo467@leonardo467: ~
File Edit View Search Terminal Help
top - 13:46:45 up 6 days, 19:49, 1 user, load average: 8.23, 8.95, 7.77
Tasks: 363 total, 2 running, 284 sleeping, 0 stopped, 0 zombie
%Cpu(s): 95.6 us, 4.4 sy, 0.0 ni, 0.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 16045092 total, 1447020 free, 4627452 used, 9970620 buff/cache
KiB Swap: 67108860 total, 67107076 free, 1784 used. 9586292 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
31659	leonard+	20	0	694588	142672	4776	S	741.6	0.9	0:53.99	bowtie2-alig+
31655	leonard+	20	0	30556	9348	4596	S	20.1	0.1	0:01.47	perl
1652	leonard+	20	0	4521748	468972	113308	S	6.6	2.9	83:18.79	gnome-shell
26621	leonard+	20	0	5636848	759284	591248	S	6.6	4.7	3:35.76	chrome
26448	leonard+	20	0	5305968	490640	130176	S	4.3	3.1	2:19.75	chrome
26144	leonard+	20	0	818564	163660	108888	S	3.3	1.0	1:58.85	chrome
12333	leonard+	20	0	6152096	731068	66692	S	3.0	4.6	162:58.85	zoom
31671	leonard+	20	0	422332	33136	29456	R	3.0	0.2	0:00.09	deepin-scre+
1678	leonard+	20	0	2989496	17252	12920	S	2.3	0.1	48:51.86	pulseaudio
1501	leonard+	20	0	1508748	152300	100204	S	2.0	0.9	117:45.77	Xorg
17634	leonard+	20	0	44332	4316	3360	R	1.0	0.0	1:30.33	top
26474	leonard+	20	0	1039812	58776	47976	S	1.0	0.4	0:13.95	chrome
26106	leonard+	20	0	1715280	306984	169812	S	0.7	1.9	1:27.86	chrome
26224	leonard+	20	0	4923148	181100	78972	S	0.7	1.1	0:12.33	chrome
779	syslog	20	0	263032	4492	3196	S	0.3	0.0	0:01.95	rsyslogd

Al terminar, el programa nos entregará una carpeta llamada “Resultados”, en donde podremos encontrar un archivo llamado “genomas\_completos.txt”.

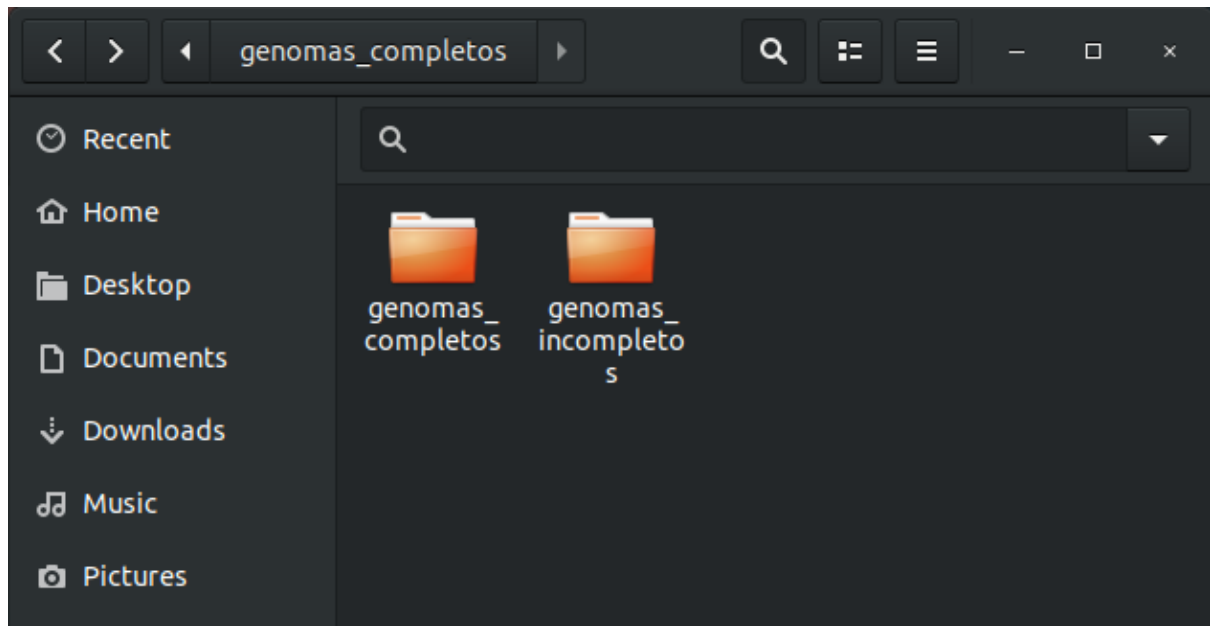
### 3. Revisión de productos del programa

Vamos a encontrar la carpeta llamada "Resultados" en vamos a encontrar lo siguiente:



A nosotros solo nos interesa el contenido de las carpetas, lo demás lo podemos ignorar. Voy a describir el contenido de cada una de las carpetas:

**Contigs-megahit:** Aquí vas a encontrar los fastas correspondientes a los ensambles de cada una de las muestras. Los contigs van a estar separados en dos carpetas: `genomas_incompletos` y `genomas_completos`. Puedes corroborar esta información en el archivo de texto “`genomas_completos.txt`”.



Ejemplo de genoma completo:

```

Open  cov0101_S1_L001.fasta  Save
Bioinfo /media/leonardo467/Bioinfo/UPCH...ial_ensamble/Res...
1 >k29_6 flag=1 multi=4143.9927 len=29848
2 AAGGACATAAGATGATAGCCTAAGAAGCTATTAAAATCACATGGGGATAGCACTACTAAAATTAATTTTACA/

```

Ejemplo de genoma no-completo:

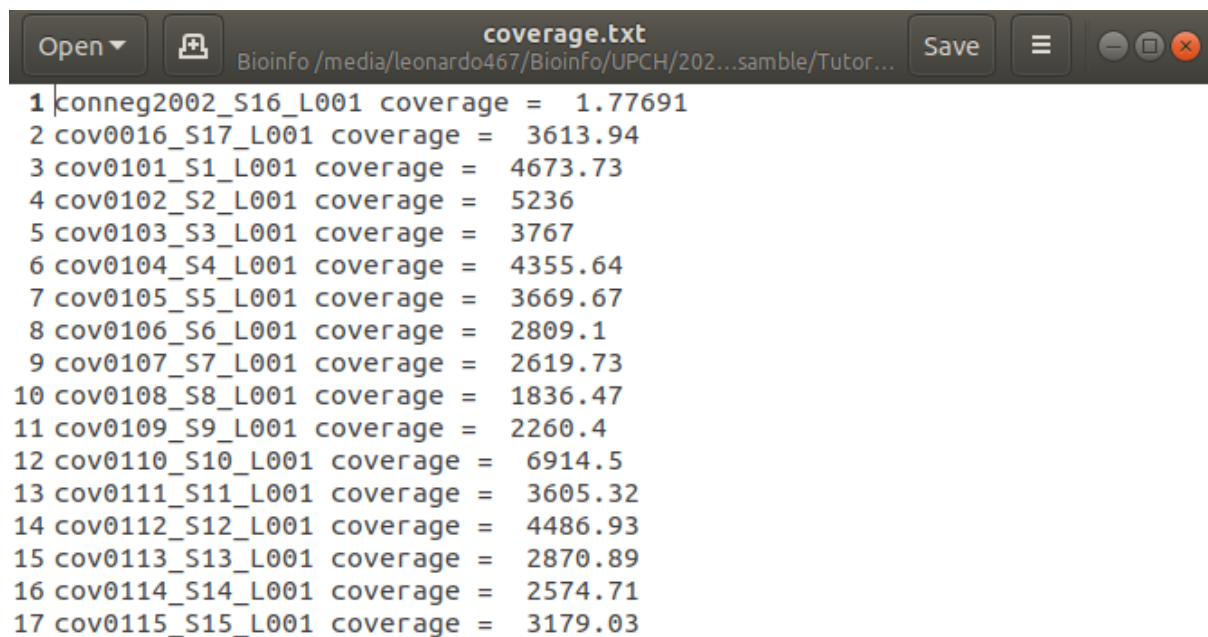
```

Open  cov0108_S8_L001.fasta  Save
Bioinfo /media/leonardo467/Bioinfo/UPCH...ial_ensamble/Res...
1 >k59_1 flag=1 multi=2175.0000 len=5025
2 AGCCCTTGAGACAACTACAGACCAACCAACTTTTCGATCTCTTGTAGATCTGTTCTCTAAACGAACTTTAAA/
3 >k59_0 flag=1 multi=5766.3880 len=5811
4 AACAACTGCCACCATCACAACCAGGCAAGTTAAGGTTAGATAGCACTCTAGTGTCAAATCTACAAACAATC/
5 >k59_2 flag=1 multi=2547.6044 len=8386
6 TCCATCATATGCAGCTTTTGTGGAAATACCCACAAGTTAATGGTTTAACTTCTATTAAATGGGCAGATAACA/
7 >k59_3 flag=1 multi=2088.0000 len=10335
8 CTCCTAAGAAGCTATTAAAATCACATGGGGATAGCACTACTAAAATTAATTTTACACATTAGGGCTCTTCCA/

```

Podemos intentar re-construir los genomas mediante un alineamiento contra un genoma de referencia.

**Coverage:** Aquí vamos a poder encontrar el coverage (profundidad de secuenciamiento) de las muestras. El programa samtools cuenta el número de nucleótidos que pasaron el filtro de calidad e hicieron match con el genoma de Sars-CoV-2 y lo divide entre el tamaño del genoma de referencia (que es 29 990 aprox). Este cálculo es hecho a partir de los archivos bam de las muestras (las cuales las vas a encontrar dentro de esta carpeta). Toda la info se encuentra dentro del archivo de texto "coverage.txt".



```
1 conneg2002_S16_L001 coverage = 1.77691
2 cov0016_S17_L001 coverage = 3613.94
3 cov0101_S1_L001 coverage = 4673.73
4 cov0102_S2_L001 coverage = 5236
5 cov0103_S3_L001 coverage = 3767
6 cov0104_S4_L001 coverage = 4355.64
7 cov0105_S5_L001 coverage = 3669.67
8 cov0106_S6_L001 coverage = 2809.1
9 cov0107_S7_L001 coverage = 2619.73
10 cov0108_S8_L001 coverage = 1836.47
11 cov0109_S9_L001 coverage = 2260.4
12 cov0110_S10_L001 coverage = 6914.5
13 cov0111_S11_L001 coverage = 3605.32
14 cov0112_S12_L001 coverage = 4486.93
15 cov0113_S13_L001 coverage = 2870.89
16 cov0114_S14_L001 coverage = 2574.71
17 cov0115_S15_L001 coverage = 3179.03
```

**Ensamble Mega-hit:** Aquí vamos a encontrar las carpetas que produjo megahit a partir de cada muestra. Aquí se encuentran los archivos fastas de cada ensamble + el log del ensamble (por si nos interesa esta info para algo). Los archivos fastas de cada ensamble ya se encuentran en la carpeta "Contigs-megahit" por el hecho que era demasiado complicado buscar fasta por fasta en los out-puts que generaba megahit.

**Solo\_match\_sars-cov-2:** Aquí se van a encontrar los archivos fastqs que han pasado por todos los filtros de Trimmomatic y bowtie2. Estos son los archivos fastq que fueron utilizados para hacer el ensamble en MEGAHIT.