

# WGS Taxonomic binning

Fernando Freire

May 14, 2019

## Contents

<b>1</b>	<b>Whole Genome Shotgun metagenomics: Taxonomic Binning Pipeline</b>	<b>2</b>
1.1	Taxonomic composition . . . . .	2
1.2	Abstract . . . . .	5
1.2.1	Pipeline comments . . . . .	5
1.2.2	Results comments . . . . .	6
1.3	Pipeline . . . . .	6
1.3.1	Check files . . . . .	6
1.3.2	Quality (fastqc) . . . . .	7
1.3.3	Trimming and decontaminating . . . . .	7
1.3.4	Blastx comparison with a viral protein database . . . . .	10

# 1 Whole Genome Shotgun metagenomics: Taxonomic Binning Pipeline

We have two fastqc files to process: 1. *Microbiome1\_200k* that contains 200,000 paired end reads from total DNA extracted from human saliva. So this is a microbiome. We tag the related information as **Bact**.

2. *Vir1\_100k* that contains 100,000 paired end reads from the same saliva sample but after purification of viral particles. So this is a virome. We tag the related information as **Vir**.

## 1.1 Taxonomic composition

### Vir\_R1

Family	Abundance
Myoviridae	338.0
Podoviridae	823.0
Siphoviridae	5677.0
unclassified Caudovirales	11.0
Phycodnaviridae	9.0
unclassified archaeal dsDNA viruses	9.0
unclassified dsDNA phages	21.0
unclassified bacterial viruses	862.0
Not assigned	4375.0

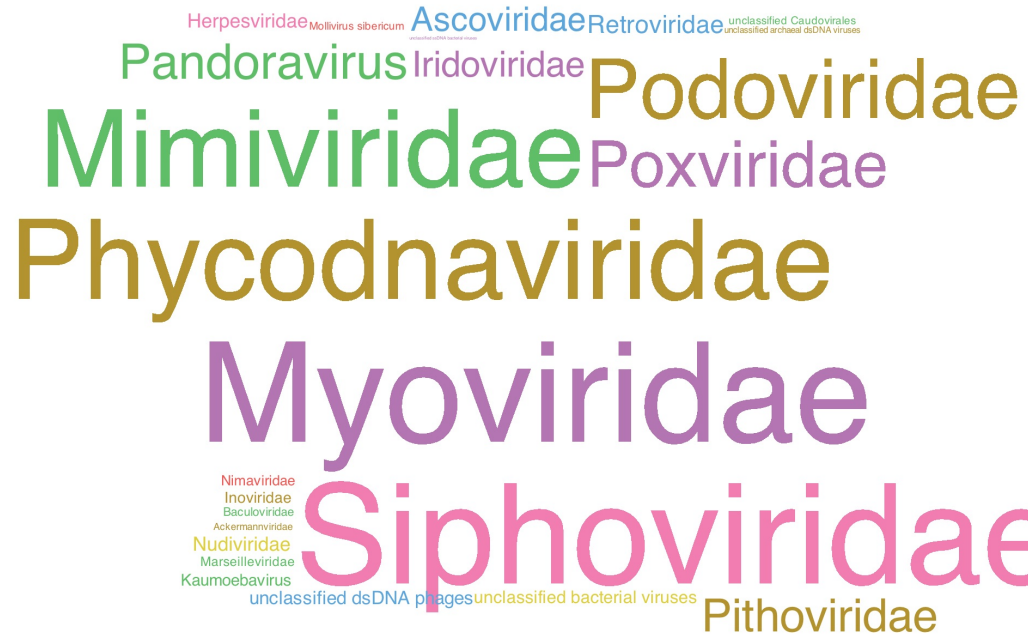
### Bact\_R1

Family	Abundance
Baculoviridae	1.0
Ackermannviridae	1.0
Myoviridae	343.0
Podoviridae	37.0
Siphoviridae	225.0
unclassified Caudovirales	3.0
Herpesviridae	3.0
Iridoviridae	8.0
Marseilleviridae	2.0
Mimiviridae	62.0
Nimaviridae	1.0
Nudiviridae	2.0
Phycodnaviridae	123.0
Poxviridae	18.0
unclassified archaeal dsDNA viruses	4.0
unclassified dsDNA phages	9.0
unclassified dsDNA viruses	28.0
Pandoravirus	15.0
Pithoviridae	13.0
Retroviridae	4.0
Inoviridae	1.0

Family	Abundance
unclassified ssDNA bacterial viruses	1.0
unclassified bacterial viruses	12.0
unclassified viruses	4.0
Not assigned	332.0

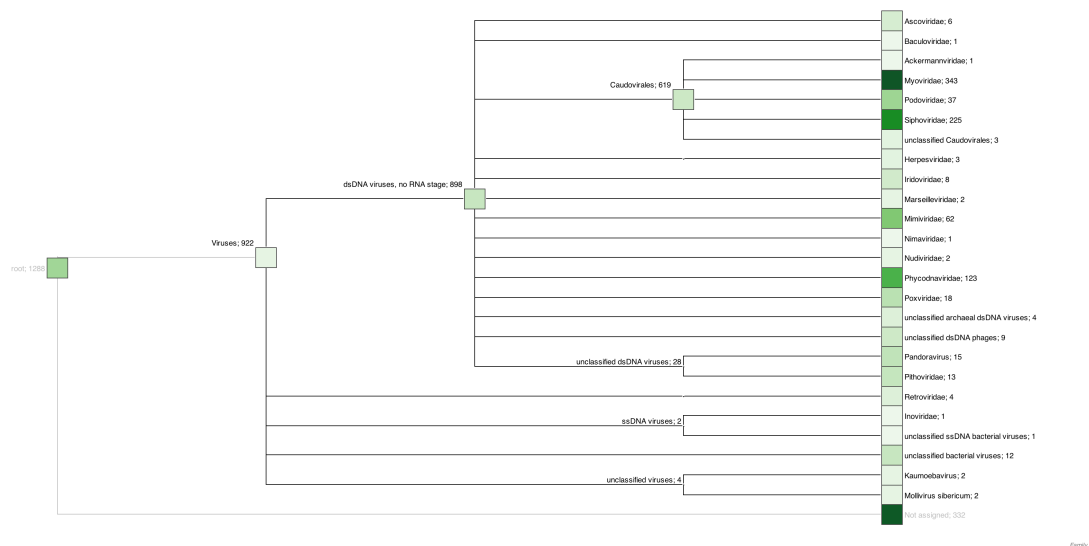


Vir\_R1 graphical word cloud

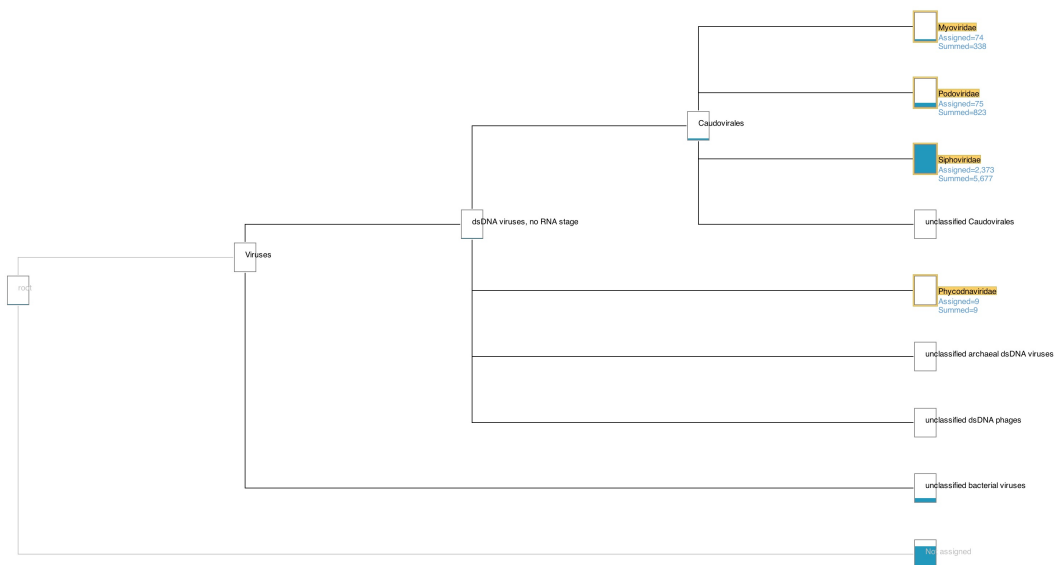


Bact\_R1 graphical word cloud

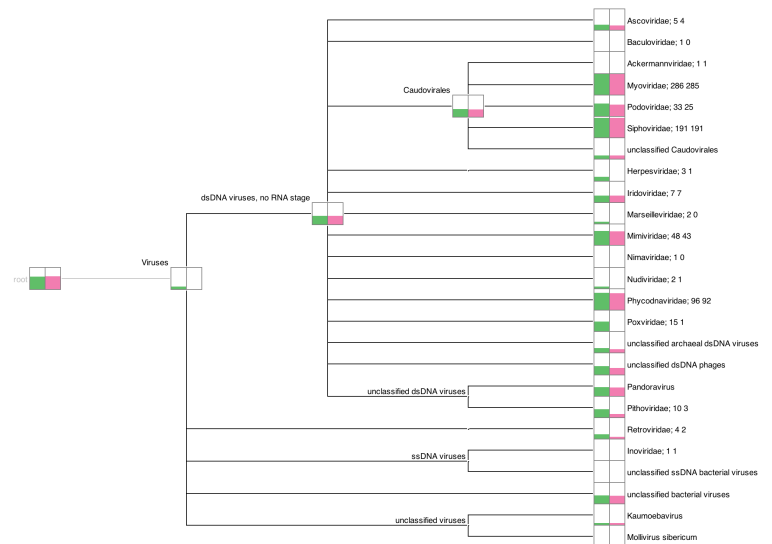
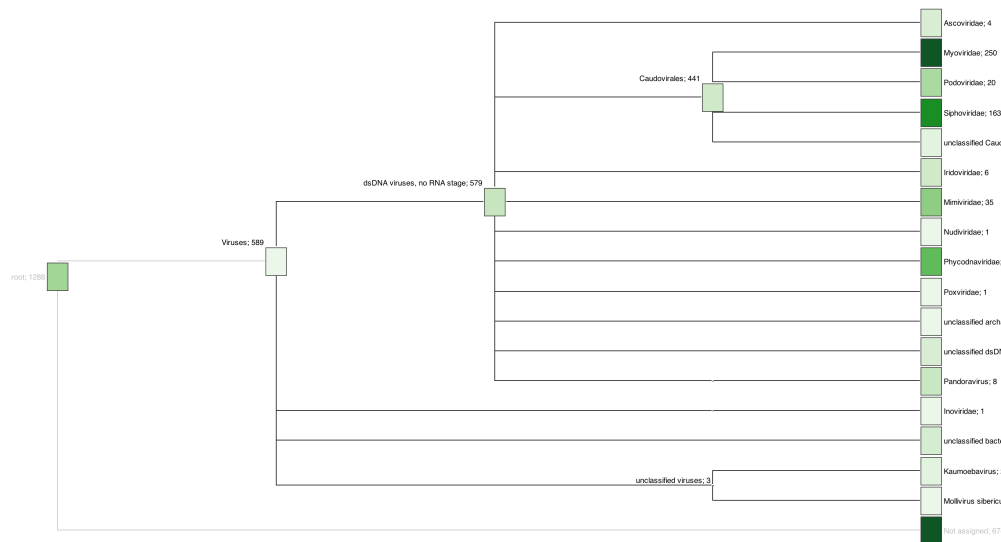
## Bact\_R1 MEGAN



## Vir\_R1 MEGAN



## Bact\_R1 paramater set 2 MEGAN



## Comparison of MEGAN executions for Bact

### 1.2 Abstract

#### 1.2.1 Pipeline comments

##### Execution procedure

We have used the same strategy as in the previous practice: launch the bash commands from a jupyter notebook, so that we have stored the executions and the outputs in the same place where we document. It also facilitates the re-executions. We opted also for the use of host machine sending the commands to guest linux machine by ssh commands, once correctly configured the protocol into linux guest.

We have tried to obtain the statistics of the processes automatically to generate from the pipeline all the requested data, since the outputs of the executions can be read and processed from python and we have achieved it in part: the programs without command line interface FASTQ and MEGAN6 do not ease these automations. *However, we have not been able to investigate this point more longer).*

##### Parameters

At *kneaddata* we employed the trimming sliding window trimmomatic option: starts scanning at the 5-end and clips the read once the average quality within the window falls below a threshold. And also we adjust the *MINLEN* parameter: drop the read if it is below a specified length. We use a conservative length of 200 bp.

After decontamination and alignments only are retained the 4% of sequences of Bact\_R1 and the 20% of Vir\_R1 as we show in the reduction of the counters across the different steps:

Sample	Vir_R1	Bact_R1	Vir_R1(%)	Bact_R1(%)
raw single	100000	200000	100	100
trimmed single	63602	127652	63,60	63,83
decontaminated Human and PhiX	63331	32358	63,33	16,18
final single	63331	32358	63,33	16,18
diamond vs viral protein db	12725	1288	20,09	3,98

### 1.2.2 Results comments

The reduction of these counters on both samples shows that in the decontamination step many more readings have been eliminated for *Bac*. As the sequencing technology is identical, this seems to indicate that the bacteriome sample was very contaminated with the human genome. In the end, the bacteriome remains reduced to 16% while the virome retains 63% of sequences, prior to the taxonomic analysis. It is coherent with the fact that no experimental previous procedure has been applied to this sample to isolate the virome from the bacteriome or from the genome of the human host.

However, the quality (analyzed with FASTQ) of both samples, once the filtering process was executed, has improved considerably compared with the initial quality.

The mapping against the database of viral proteins carried out by diamond also indicates a lower number of hits than in the case of the virome: the reads are reduced to one fifth, while the virome is reduced by only one fourth. This indicates that many non-viral sequences still exist within the sample (although it can also be interpreted as indicating that the sample may contain traces of viruses not referenced in the reference database).

Another relevant fact is that after the analysis made with MEGA, we observed a greater viral biodiversity in the bacteriome (it can be clearly seen in the word cloud output format offered by the application) These traces must be provided by the bacterial genome, since it is assumed that we are analyzing the same viral ecosystem in both samples, this is coherent with the idea that bacteria (and other taxa) have been incorporating across evolution pieces of virus DNA, either directly from viruses or through horizontal transfer from other bacteria.

One of the mechanisms by which the viral genome is incorporated into the bacterial one, could be related with the *CRISPR-CAS* subsystem, being integrated in the *CRISPR* locus, provided that these inserts being long enough to throw positive matchings against the reference virome.

## 1.3 Pipeline

### Script 1.3.1 (python)

```
1 import warnings
2 warnings.filterwarnings('ignore')
3 import pandas as pd
```

### 1.3.1 Check files

Check the number of reads of our files and look at the header with grep and head commands

#### Script 1.3.2 (bash)

```
1 %%bash
2 ssh microbiointf@192.168.56.101 2>/dev/null /bin/bash <<'EOT'
3 cd Documentos/Tema_4a
4 head -n 1 *fq
5 grep -c "@M02255" *fq
6 EOT
```

#### Output

```
==> Bact1_R1_200000.fq <==
@M02255:131:000000000-AJC6R:1:2102:25217:13392_1:N:0:ACAGTG

==> Bact1_R2_200000.fq <==
@M02255:131:000000000-AJC6R:1:2102:25217:13392_2:N:0:ACAGTG

==> Vir1_R1_100000.fq <==
@M02255:131:000000000-AJC6R:1:1105:23249:10170_1:N:0:AGTCAA

==> Vir1_R2_100000.fq <==
@M02255:131:000000000-AJC6R:1:1105:23249:10170_2:N:0:AGTCAA
Bact1_R1_200000.fq:200000
Bact1_R2_200000.fq:200000
Vir1_R1_100000.fq:100000
Vir1_R2_100000.fq:100000
```

### 1.3.2 Quality (fastqc)

#### Script 1.3.3 (bash)

```
1 %%bash
2 ssh microbiointf@192.168.56.101 2>/dev/null /bin/bash <<'EOT'
3 export PATH=$PATH:/home/microbiointf/miniconda3/bin
4 cd Documentos/Tema_4a
5 fastqc Vir1_R1_100000.fq -o fastQC_output/
6 fastqc Vir1_R2_100000.fq -o fastQC_output/
7 fastqc Bact1_R1_200000.fq -o fastQC_output/
8 fastqc Bact1_R2_200000.fq -o fastQC_output/
9 EOT
```

#### Output

```
Analysis complete for Vir1_R1_100000.fq
Analysis complete for Vir1_R2_100000.fq
Analysis complete for Bact1_R1_200000.fq
```

Analysis complete for Bact1\_R2\_200000.fq

### 1.3.3 Trimming and decontaminating

Trimming poor quality ends and short sequences (**Trimmomatic**) and removal of reads aligning to the human and phiX174 genomes (**\*bowtie2**). The later one is a contaminant used as spike by Illumina kits to control quality of the sequencing process.

We are only filtering only R1 files because forward reads have usually better quality than reverse reads.

#### Processing

##### Script 1.3.4 (bash)

```
1 %%bash
2 ssh microbioinf@192.168.56.101 2>/dev/null /bin/bash <<'EOT'
3 export PATH=$PATH:/home/microbioinf/miniconda3/bin
4 cd Documentos/Tema_4a
5 kneaddata -i Bact1_R1_200000.fq -o kneaddata_out \
6 -db /home/shared/bowtiedb/GRCh38_PhiX \
7 --trimmomatic /home/microbioinf/miniconda3/pkgs/trimmomatic-0.38-1/share/trimmomatic-0.38-1/
8 \
9 -t 2 --trimmomatic-options "SLIDINGWINDOW:4:20 MINLEN:200" \
10 --bowtie2-options " --sensitive" --remove-intermediate-output
11 EOT
```

##### Output

```
Initial number of reads ( /home/microbioinf/Documentos/Tema_4a/Bact1_R1_200000.fq ): 200000
Running Trimmomatic ...
Total reads after trimming ( /home/microbioinf/Documentos/Tema_4a/kneaddata_out/Bact1_R1_2000_
→ 00_kneaddata.trimmed.fastq ):
→ 127652
Decontaminating ...
Running bowtie2 ...
Total reads after removing those found in reference database ( /home/microbioinf/Documentos/T
→ ema_4a/kneaddata_out/Bact1_R1_200000_kneaddata_GRCh38_PhiX_bowtie2_clean.fastq ):
→ 32358
Total reads after merging results from multiple databases (
→ /home/microbioinf/Documentos/Tema_4a/kneaddata_out/Bact1_R1_200000_kneaddata.fastq ):
→ 32358

Final output file created:
/home/microbioinf/Documentos/Tema_4a/kneaddata_out/Bact1_R1_200000_kneaddata.fastq
```

##### Script 1.3.5 (bash)

```
1 %%bash
2 ssh microbioinf@192.168.56.101 2>/dev/null /bin/bash <<'EOT'
3 export PATH=$PATH:/home/microbioinf/miniconda3/bin
```



```

4 cd Documentos/Tema_4a
5 kneaddata -i Vir1_R1_100000.fq -o kneaddata_out \
6 -db /home/shared/bowtiedb/GRCh38_PhiX \
7 --trimmomatic /home/microbioinf/miniconda3/pkgs/trimmomatic-0.38-1/share/trimmomatic-0.38-1/
8 \
9 -t 2 --trimmomatic-options "SLIDINGWINDOW:4:20 MINLEN:200" \
10 --bowtie2-options " --sensitive" --remove-intermediate-output
EOT

```

## Output

```

Initial number of reads ( /home/microbioinf/Documentos/Tema_4a/Vir1_R1_100000.fq ): 100000
Running Trimmomatic ...
Total reads after trimming (
  ↪ /home/microbioinf/Documentos/Tema_4a/kneaddata_out/Vir1_R1_100000_kneaddata.trimmed.fastq
  ↪ ): 63602
Decontaminating ...
Running bowtie2 ...
Total reads after removing those found in reference database ( /home/microbioinf/Documentos/T
  ↪ ema_4a/kneaddata_out/Vir1_R1_100000_kneaddata_GRCh38_PhiX_bowtie2_clean.fastq ):
  ↪ 63331
Total reads after merging results from multiple databases (
  ↪ /home/microbioinf/Documentos/Tema_4a/kneaddata_out/Vir1_R1_100000_kneaddata.fastq ): 63331

Final output file created:
/home/microbioinf/Documentos/Tema_4a/kneaddata_out/Vir1_R1_100000_kneaddata.fastq

```

## Process statistics

### Script 1.3.6 (bash)

```

1 %%bash
2 ssh microbioinf@192.168.56.101 2>/dev/null /bin/bash <<'EOT'
3 export PATH=$PATH:/home/microbioinf/miniconda3/bin
4 cd Documentos/Tema_4a/kneaddata_out
5 kneaddata_read_count_table --input ./ --output kneaddata_read_counts.txt
6 EOT

```

## Output

```

Reading log: ./Bact1_R1_200000_kneaddata.log
Reading log: ./Vir1_R1_100000_kneaddata.log
Read count table written: kneaddata_read_counts.txt

```

### Script 1.3.7 (python)

```

1 data = ""
2 cat Documentos/Tema_4a/kneaddata_out/kneaddata_read_counts.txt
3 EOT

```

```

4 """
5 output = !ssh microbioinf@192.168.56.101 2>/dev/null /bin/bash <<'EOT' {data}
6
7 data = []
8 # To list of lists
9 for row in output:
10     data.append(row.split('\t'))
11 # To dataframe
12 df = pd.DataFrame(data[1:], columns=data[0])
13 df.style.hide_index().set_properties(**{'text-align': 'right', 'font-family' : 'courier',
14     ↪ 'color' : 'darkgreen', "font-size" : "11pt"}).\
15 set_properties(**{'text-align': 'right', 'font-family' : 'courier', 'color' : 'darkblue',
16     ↪ "font-size" : "12pt"}, subset=['Sample'])

```

### Display output

```
<pandas.io.formats.style.Styler at 0x11bc43e48>
```

With grep we can identify the non-contaminated high-quality files

### Script 1.3.8 (bash)

```

1 %%bash
2 ssh microbioinf@192.168.56.101 2>/dev/null /bin/bash <<'EOT'
3 cd Documentos/Tema_4a/kneaddata_out
4 grep -c "@M02255:" *fastq
5 EOT

```

### Output

```

Bact1_R1_200000_kneaddata.fastq:32358
Bact1_R1_200000_kneaddata_GRCh38_PhiX_bowtie2_contam.fastq:95294
Vir1_R1_100000_kneaddata.fastq:63331
Vir1_R1_100000_kneaddata_GRCh38_PhiX_bowtie2_contam.fastq:271

```

## Check quality

### Script 1.3.9 (bash)

```

1 %%bash
2 ssh microbioinf@192.168.56.101 2>/dev/null /bin/bash <<'EOT'
3 cd Documentos/Tema_4a/kneaddata_out
4 mkdir fastQC_HighQuality
5 export PATH=$PATH:/home/microbioinf/miniconda3/bin
6 fastqc Bact1_R1_200000_kneaddata.fastq -o fastQC_HighQuality/
7 fastqc Vir1_R1_100000_kneaddata.fastq -o fastQC_HighQuality/
8 EOT

```

## Output

```
Analysis complete for Bact1_R1_200000_kneaddata.fastq
Analysis complete for Vir1_R1_100000_kneaddata.fastq
```

### 1.3.4 Blastx comparison with a viral protein database

We are going to use a Refseq database of viral proteins (around 100Mb) from ncbi (<ftp://ftp.ncbi.nlm.nih.gov/refseq/release/viral/>), and you have to download it in two separated files that can be joined into one with cat.

**Set up the reference database for diamond** This will create a binary DIAMOND database file with the name: *viralprotein.dmmnd*

#### Script 1.3.10 (bash)

```
1 %%bash
2 ssh microbiointf@192.168.56.101 2>/dev/null /bin/bash <<'EOT'
3 cd Documentos/Tema_4a
4 export PATH=$PATH:/home/microbioinf/miniconda3/bin
5 diamond makedb --in viral.protein.faa -d viralprotein
6 EOT
```

## Output

```
diamond v0.8.36.98 | by Benjamin Buchfink <buchfink@gmail.com>
Check http://github.com/bbuchfink/diamond for updates.

#CPU threads: 3
Database file: viral.protein.faa
Opening the database file... [0.015813s]
Loading sequence data (0 sequences processed)... [0.149662s]
Loading sequence data (100000 sequences processed)... [0.142183s]
Loading sequence data (200000 sequences processed)... [0.158145s]
Loading sequence data (300000 sequences processed)... [0.040807s]
Writing trailer... [0.003161s]
Closing the input file... [4e-05s]
Closing the database file... [0.061918s]
Processed 323029 sequences, 82978170 letters.
Total time = 0.572598s
```

### Blastx alignments

#### Script 1.3.11 (bash)

```
1 %%bash
2 ssh microbiointf@192.168.56.101 2>/dev/null /bin/bash <<'EOT'
3 cd Documentos/Tema_4a
4 export PATH=$PATH:/home/microbioinf/miniconda3/bin
```

```

5 diamond blastx -d viralprotein.dmnd -q kneaddata_out/Vir1_R1_100000_kneaddata.fastq -o
  ↳ Vir1HQ_Vs_viralprotein.m8
6 diamond blastx -d viralprotein.dmnd -q kneaddata_out/Bact1_R1_200000_kneaddata.fastq -o
  ↳ Bact1HQ_Vs_viralprotein.m8
7 EOT

```

## Output

```

diamond v0.8.36.98 | by Benjamin Buchfink <buchfink@gmail.com>
Check http://github.com/bbuchfink/diamond for updates.

#CPU threads: 3
Scoring parameters: (Matrix=BLOSUM62 Lambda=0.267 K=0.041 Penalties=11/1)
#Target sequences to report alignments for: 25
Temporary directory:
Opening the database... [1.5e-05s]
Opening the input file... [2.8e-05s]
Opening the output file... [4.1e-05s]
Loading query sequences... [0.235952s]
Running complexity filter... [3.78898s]
Building query histograms... [0.318931s]
Allocating buffers... [6.2e-05s]
Loading reference sequences... [0.107758s]
Building reference histograms... [1.18488s]
Allocating buffers... [6.1e-05s]
Initializing temporary storage... [0.000619s]
Processing query chunk 0, reference chunk 0, shape 0, index chunk 0.
Building reference index... [0.969958s]
Building query index... [0.217957s]
Building seed filter... [0.091635s]
Searching alignments... [0.137262s]
Processing query chunk 0, reference chunk 0, shape 0, index chunk 1.
Building reference index... [1.11468s]
Building query index... [0.246178s]
Building seed filter... [0.090143s]
Searching alignments... [0.138724s]
Processing query chunk 0, reference chunk 0, shape 0, index chunk 2.
Building reference index... [1.127s]
Building query index... [0.249563s]
Building seed filter... [0.109712s]
Searching alignments... [0.183329s]
Processing query chunk 0, reference chunk 0, shape 0, index chunk 3.
Building reference index... [1.26957s]
Building query index... [0.370472s]
Building seed filter... [0.150778s]
Searching alignments... [0.22253s]
Processing query chunk 0, reference chunk 0, shape 1, index chunk 0.
Building reference index... [1.34065s]
Building query index... [0.265761s]
Building seed filter... [0.113665s]
Searching alignments... [0.429611s]

```

Processing query chunk 0, reference chunk 0, shape 1, index chunk 1.  
Building reference index... [1.4586s]  
Building query index... [0.295433s]  
Building seed filter... [0.120444s]  
Searching alignments... [0.16581s]  
Processing query chunk 0, reference chunk 0, shape 1, index chunk 2.  
Building reference index... [1.45692s]  
Building query index... [0.302227s]  
Building seed filter... [0.110777s]  
Searching alignments... [0.159347s]  
Processing query chunk 0, reference chunk 0, shape 1, index chunk 3.  
Building reference index... [1.3176s]  
Building query index... [0.278914s]  
Building seed filter... [0.115424s]  
Searching alignments... [0.165368s]  
Processing query chunk 0, reference chunk 0, shape 2, index chunk 0.  
Building reference index... [1.33535s]  
Building query index... [0.246887s]  
Building seed filter... [0.117666s]  
Searching alignments... [0.147365s]  
Processing query chunk 0, reference chunk 0, shape 2, index chunk 1.  
Building reference index... [1.31552s]  
Building query index... [0.204659s]  
Building seed filter... [0.087173s]  
Searching alignments... [0.110421s]  
Processing query chunk 0, reference chunk 0, shape 2, index chunk 2.  
Building reference index... [1.10052s]  
Building query index... [0.21389s]  
Building seed filter... [0.086331s]  
Searching alignments... [0.109901s]  
Processing query chunk 0, reference chunk 0, shape 2, index chunk 3.  
Building reference index... [0.938978s]  
Building query index... [0.179356s]  
Building seed filter... [0.085301s]  
Searching alignments... [0.109947s]  
Processing query chunk 0, reference chunk 0, shape 3, index chunk 0.  
Building reference index... [0.920597s]  
Building query index... [0.171162s]  
Building seed filter... [0.085914s]  
Searching alignments... [0.101702s]  
Processing query chunk 0, reference chunk 0, shape 3, index chunk 1.  
Building reference index... [1.04422s]  
Building query index... [0.199129s]  
Building seed filter... [0.084341s]  
Searching alignments... [0.106007s]  
Processing query chunk 0, reference chunk 0, shape 3, index chunk 2.  
Building reference index... [1.0729s]  
Building query index... [0.204229s]  
Building seed filter... [0.085288s]  
Searching alignments... [0.127639s]  
Processing query chunk 0, reference chunk 0, shape 3, index chunk 3.  
Building reference index... [0.903671s]

```

Building query index... [0.167927s]
Building seed filter... [0.089396s]
Searching alignments... [0.105627s]
Deallocating buffers... [0.009547s]
Computing alignments... [1.84423s]
Deallocating reference... [0.00306s]
Loading reference sequences... [2.2e-05s]
Deallocating buffers... [0.002337s]
Deallocating queries... [0.002664s]
Loading query sequences... [2.2e-05s]
Closing the input file... [7e-06s]
Closing the output file... [0.005769s]
Closing the database file... [1.7e-05s]
Total time = 34.1638s
Reported 75459 pairwise alignments, 75459 HSSPs.
12725 queries aligned.
diamond v0.8.36.98 | by Benjamin Buchfink <buchfink@gmail.com>
Check http://github.com/bbuchfink/diamond for updates.

#CPU threads: 3
Scoring parameters: (Matrix=BLOSUM62 Lambda=0.267 K=0.041 Penalties=11/1)
#Target sequences to report alignments for: 25
Temporary directory:
Opening the database... [1.5e-05s]
Opening the input file... [2.8e-05s]
Opening the output file... [7.2e-05s]
Loading query sequences... [0.128431s]
Running complexity filter... [1.89243s]
Building query histograms... [0.160397s]
Allocating buffers... [7.4e-05s]
Loading reference sequences... [0.112703s]
Building reference histograms... [1.18714s]
Allocating buffers... [0.000131s]
Initializing temporary storage... [0.000738s]
Processing query chunk 0, reference chunk 0, shape 0, index chunk 0.
Building reference index... [0.973812s]
Building query index... [0.118915s]
Building seed filter... [0.085079s]
Searching alignments... [0.088929s]
Processing query chunk 0, reference chunk 0, shape 0, index chunk 1.
Building reference index... [1.2453s]
Building query index... [0.13635s]
Building seed filter... [0.087573s]
Searching alignments... [0.096991s]
Processing query chunk 0, reference chunk 0, shape 0, index chunk 2.
Building reference index... [1.17932s]
Building query index... [0.136485s]
Building seed filter... [0.08577s]
Searching alignments... [0.087322s]
Processing query chunk 0, reference chunk 0, shape 0, index chunk 3.
Building reference index... [0.960882s]
Building query index... [0.116979s]

```

Building seed filter... [0.08987s]  
Searching alignments... [0.086879s]  
Processing query chunk 0, reference chunk 0, shape 1, index chunk 0.  
Building reference index... [0.947816s]  
Building query index... [0.125525s]  
Building seed filter... [0.089527s]  
Searching alignments... [0.081632s]  
Processing query chunk 0, reference chunk 0, shape 1, index chunk 1.  
Building reference index... [1.09524s]  
Building query index... [0.129516s]  
Building seed filter... [0.088017s]  
Searching alignments... [0.080841s]  
Processing query chunk 0, reference chunk 0, shape 1, index chunk 2.  
Building reference index... [1.11805s]  
Building query index... [0.12736s]  
Building seed filter... [0.086305s]  
Searching alignments... [0.083204s]  
Processing query chunk 0, reference chunk 0, shape 1, index chunk 3.  
Building reference index... [0.932964s]  
Building query index... [0.109149s]  
Building seed filter... [0.082084s]  
Searching alignments... [0.080282s]  
Processing query chunk 0, reference chunk 0, shape 2, index chunk 0.  
Building reference index... [0.93305s]  
Building query index... [0.101445s]  
Building seed filter... [0.081855s]  
Searching alignments... [0.07451s]  
Processing query chunk 0, reference chunk 0, shape 2, index chunk 1.  
Building reference index... [1.04542s]  
Building query index... [0.114735s]  
Building seed filter... [0.081284s]  
Searching alignments... [0.074126s]  
Processing query chunk 0, reference chunk 0, shape 2, index chunk 2.  
Building reference index... [1.08343s]  
Building query index... [0.124619s]  
Building seed filter... [0.082408s]  
Searching alignments... [0.069044s]  
Processing query chunk 0, reference chunk 0, shape 2, index chunk 3.  
Building reference index... [0.901484s]  
Building query index... [0.104237s]  
Building seed filter... [0.081542s]  
Searching alignments... [0.070414s]  
Processing query chunk 0, reference chunk 0, shape 3, index chunk 0.  
Building reference index... [0.907908s]  
Building query index... [0.094342s]  
Building seed filter... [0.079934s]  
Searching alignments... [0.070951s]  
Processing query chunk 0, reference chunk 0, shape 3, index chunk 1.  
Building reference index... [1.05012s]  
Building query index... [0.109936s]  
Building seed filter... [0.082596s]  
Searching alignments... [0.07083s]

```

Processing query chunk 0, reference chunk 0, shape 3, index chunk 2.
Building reference index... [1.09294s]
Building query index... [0.117459s]
Building seed filter... [0.083417s]
Searching alignments... [0.073389s]
Processing query chunk 0, reference chunk 0, shape 3, index chunk 3.
Building reference index... [0.929055s]
Building query index... [0.099267s]
Building seed filter... [0.082223s]
Searching alignments... [0.069398s]
Deallocating buffers... [0.010661s]
Computing alignments... [0.215504s]
Deallocating reference... [0.004664s]
Loading reference sequences... [5.2e-05s]
Deallocating buffers... [0.000984s]
Deallocating queries... [0.001405s]
Loading query sequences... [2.7e-05s]
Closing the input file... [1.9e-05s]
Closing the output file... [0.002279s]
Closing the database file... [1.7e-05s]
Total time = 24.599s
Reported 6449 pairwise alignments, 6449 HSSPs.
1288 queries aligned.

```

#### Script 1.3.12 (bash)

```

1 %bash --out output
2 ssh microbioint@192.168.56.101 2>/dev/null /bin/bash <<'EOT'
3 cd Documentos/Tema_4a
4 head -n 1 *fq
5 grep -c "@M02255" *fq
6 EOT

```

#### Script 1.3.13 (python)

```

1 print(output)

```

#### Output

```

==> Bact1_R1_200000.fq <==
@M02255:131:000000000-AJC6R:1:2102:25217:13392_1:N:0:ACAGTG

==> Bact1_R2_200000.fq <==
@M02255:131:000000000-AJC6R:1:2102:25217:13392_2:N:0:ACAGTG

==> Vir1_R1_100000.fq <==
@M02255:131:000000000-AJC6R:1:1105:23249:10170_1:N:0:AGTCAA

==> Vir1_R2_100000.fq <==
@M02255:131:000000000-AJC6R:1:1105:23249:10170_2:N:0:AGTCAA

```



```
Bact1_R1_200000.fq:200000
Bact1_R2_200000.fq:200000
Vir1_R1_100000.fq:100000
Vir1_R2_100000.fq:100000
```

### Script 1.3.14 (python)

```
1 alist = output.split('\n')
2 alist
```

### Display output

```
['==> Bact1_R1_200000.fq <==',
 '@M02255:131:000000000-AJC6R:1:2102:25217:13392_1:N:0:ACAGTG',
 '',
 '==> Bact1_R2_200000.fq <==',
 '@M02255:131:000000000-AJC6R:1:2102:25217:13392_2:N:0:ACAGTG',
 '',
 '==> Vir1_R1_100000.fq <==',
 '@M02255:131:000000000-AJC6R:1:1105:23249:10170_1:N:0:AGTCAA',
 '',
 '==> Vir1_R2_100000.fq <==',
 '@M02255:131:000000000-AJC6R:1:1105:23249:10170_2:N:0:AGTCAA',
 'Bact1_R1_200000.fq:200000',
 'Bact1_R2_200000.fq:200000',
 'Vir1_R1_100000.fq:100000',
 'Vir1_R2_100000.fq:100000',
 '']
```