Organisation and utilisation of hologenomic datasets course

HoloFood Consortium

Sep 27, 2022

SESSIONS OF THE COURSE:

CHAPTER

ONE

SOFTWARE AND DATA REQUIRED FOR THE COURSE

Important:

- **The course instructions assume you are using a Linux environment**
- If you're using a Mac or Windows computer, you might find it easier to set up a Linux Virtual Machine using software like Virtual Box. (Instructions below.)
- However, all of the software used should also be installable on Mac or Windows computers.

1.1 Assumed file system structure

All of the practical sessions are written to refer to various pieces of data in a root directory called `/course`. If you're using a Virtual Machine, you can just make this directory (`sudo mkdir /course`) and put the various pieces of data there. If you're using your own computer, and put the data elsewhere like somewhere in your home folder, you'll need to modify the course instructions appropriately.

1.2 Setting up a Linux virtual machine for the course

- Follow the Ubuntu instructions for creating an Ubuntu VM.
- You'll need to allocate at least 8GB of memory to the VM to run every step of the course.

1.3 Installing software for the course

1.3.1 Docker

Docker allows you to run "containers": reproducible builds of certain tools. Install Docker Desktop (or alternatives like Podman).

1.3.2 Anaconda

Conda allows you to create "environments": sets of tools and libraries that depend on each other. Install Anaconda distribution.

1.3.3 Sirius

Sirius is a tool for analysting metabolite data. Install Sirius 4.

1.3.4 MZmine

MZmine is a tool for processing mass-spectrometery data. Install MZmine 3.

1.3.5 Gemma

Gemma is a tool for working with genome-wide association studies. Install Gemma 0.98.3.

1.3.6 Bedtools

Bedtools is a set of tools for genomic analysis. Install Bedtools 2.30.0.

1.3.7 Dependencies

`cd /course` (assuming you are using a Virtual Machine, see notes above)

This fetches the course notes, some code notebooks, and various dependencies and datasets: `git clone https://github.com/ebi-metagenomics/holofood-course.git docs`

This creates Conda environments with the dependencies required for the practical sessions: `cd docs/sessions/Metabolomics/`

```
conda create -f Metabolomics.yml
```

```
cd docs/sessions/metagenomics/notebooks/
```

```
conda create --name jupyter -c conda-forge jupyterlab
```

```
conda acivate jupyter
```

```
pip install -r requirements.txt
```

```
conda create --name r --channel conda-forge "r-base>=4.0.3" r-devtools
```

```
conda activate r
```

```
conda install -c conda-forge r-reshape2 r-ggplot2
```

1.4 Copying data for the course

1.4.1 For the MAG generation practical

Download all of the data from this EBI-hosted FTP site.

Unzip any of the `.tar.gz` files, using e.g. `tar -xzf eukaryotes.tar.gz`.

1.4.2 For the multi-kingdom metagenomics practical

```
wget http://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify_courses/biata_2021/virify_
↪tutorial.tar.gz
or
rsync -av --partial --progress rsync://ftp.ebi.ac.uk/pub/databases/metagenomics/mgnify_
↪courses/biata_2021/virify_tutorial.tar.gz .
```

Once downloaded, extract the files from the tarball:

```
tar -xzvf virify_tutorial.tar.gz
```

Now change into the **virify_tutorial** directory and setup the environment by running the following commands in your current terminal session:

```
cd virify_tutorial
docker load --input docker/virify.tar
docker run --rm -it -v $(pwd)/data:/opt/data virify
mkdir obs_results
```

CHAPTER

TWO

HOLOFOOD DATA IN PUBLIC ARCHIVES – PRACTICAL SESSION

In this hands-on session, we will learn about the HoloFood Data Portal – a web resource for finding and using the samples and datasets created by the HoloFood project.

2.1 Follow the HoloFood Data Portal tutorial

Open the HoloFood documentation, and follow the Tutorial.

For the first few learning objectives, you just need to use the HoloFood Data Portal.

For the last learning objective, you need to write (or copy) some Python code to fetch data, analyse data, and make a plot.

Hint: There is a Jupyter Notebook available on the course-provided virtual machines, so you don't need to set up Python or install anything. However, to follow this workshop at a later date, see the github repo for installation instructions.

2.1.1 To use the Jupyter Nobteook

- Open a Terminal
- Type the following commands:

```
hf-conda-setup
conda activate jupyter
cd /course/docs/sessions/holofood-data-portal/notebooks
jupyter lab
```

Find the "HoloFood Data Portal Tutorial Objective 7.ipynb" notebook in the lefthand bar. Double-click it.

CHAPTER

THREE

MAG GENERATION

- Generation of metagenome assembled genomes (MAGs) from assemblies
- Assessment of quality (MIGMAGs)
- Taxonomic assignment
- Comparison to public genomes

3.1 Prerequisites

For this tutorial you will need to first setup the docker container by running:

```
cd /course/metagenomics-data/mag_generation/data
docker run --rm -it -v $(pwd):/opt/data quay.io/microbiome-informatics/holofood-course-
↪2022-mag-generation:latest
cd /opt/data
mkdir obs_results
```

 You will notice the outputs of assembly, binning and checkM have been pre-generated for you. Since some of these processes can take ~1hr we have provided the results to save time. To view the available data run:

```
ls /opt/data
```

3.2 Assembling data

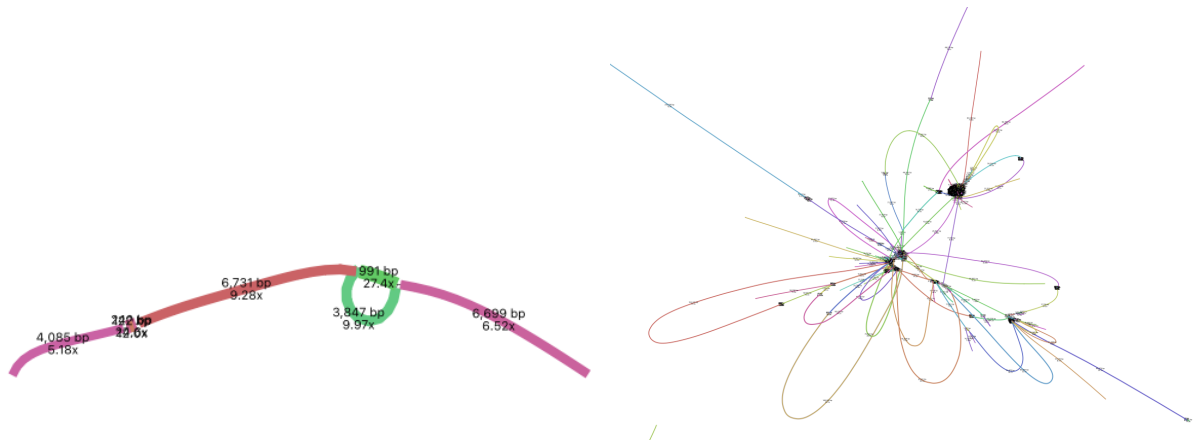
 Learning Objectives - in the following exercises an assembled HoloFood salmon gut sample has been provided. You will assess the quality of these assemblies, which are used later to generate bins.

Once quality control of sequence reads is complete, you may want to perform *de novo* assembly in addition to, or as an alternative to a read-based analyses. The first step is to assemble your sequences into contigs. There are many tools available for this, such as MetaVelvet, metaSPAdes, IDBA-UD, MEGAHIT. We generally use metaSPAdes, as in most cases it yields the best contig size statistics (i.e. more continguous assembly) and has been shown to be able to capture high degrees of community diversity (Vollmers, et al. PLOS One 2017). However, you should consider the pros and cons of different assemblers, which not only includes the accuracy of the assembly, but also their computational overhead. For example, very diverse samples with a lot of sequence data uses a lot of memory with metaSPAdes.

 The assembly step has been run for you. To run metaSPAdes we executed the following commands (but don't run it now!):

```
mkdir assemblies
metaspades.py -t 4 --only-assembler -m 10 -1 reads/ERR4918566_1.fastq.gz -2 reads/
↪ERR4918566_2.fastq.gz -o assemblies
```

MetaSPAdes also produces assembly graphs which can be visualised using tools such as Bandage. Parts of the assembly graph from the above assembly are shown below.



The simplest graph would contain a single long contig but this is not always the case.

The graph on the left is made up of several kmers. It also contains a "bubble" which could be repeated sequences appearing as single nodes with multiple inputs and outputs.

The right hand graph is very complex and difficult to resolve.

Assessing genome quality

Assemblies can contain contamination from several sources e.g. host, human, PhiX and so on.

PhiX, is a small bacteriophage genome typically used as a calibration control in sequencing runs. Most library preparations will use PhiX at low concentrations, however it can still appear in the sequencing run. If not filtered out, PhiX can form small spurious contigs which could be incorrectly classified as diversity.

Lets assess the resulting assembly contigs file. Run the following to make a PhiX reference database, followed by blast to identify PhiX contigs in our assembly file:

```
# make reference database
makeblastdb -in assemblies/decontamination/phiX.fna -input_type fasta -dbtype nucl -
↪parse_seqids -out obs_results/phix_blastDB

# run blast
blastn -query assemblies/ERR4918566.fasta -db obs_results/phix_blastDB -task megablast -
↪word_size 28 -best_hit_overhang 0.1 -best_hit_score_edge 0.1 -dust yes -evalue 0.0001 -
↪min_raw_gapped_score 100 -penalty -5 -soft_masking true -window_size 100 -outfmt 6 -
↪out obs_results/ERR4918566.blast.out
```

View the blast results

```
cat obs_results/ERR4918566.blast.out
```

Use the following link to understand what is in each column https://www.metagenomics.wiki/tools/blast/blastn-output-format-6

Are there any significant hits?

What are the lengths of the matching contigs?

We would typically filter metagenomic contigs at a length of 500bp. Would any PhiX contamination remain after this filter?

Within the /opt/data/assemblies folder there is a second cleaned contigs file with contigs <500bp filtered out and contamination removed.

Lets assess the statistics of assemblies before and after quality control.

```
gunzip assemblies/ERR4918566_clean.fasta.gz

# statistics before quality control
assembly_stats assemblies/ERR4918566.fasta > obs_results/assembly_stats.json

# statistics after quality control
assembly_stats assemblies/ERR4918566_clean.fasta > obs_results/assembly_stats_clean.json
```

This will output two simple tables in JSON format, but it is fairly simple to read. To view each file you can open it via the folders or run:

```
cat obs_results/assembly_stats.json
cat obs_results/assembly_stats_clean.json
```

Looking at the 'Contig stats' for both, what is the length of longest and shortest contigs before and after quality control?

What is the N50 of the two assembly files? Given that the input sequences were ~150bp long paired-end sequences, what does this tell you about the assembly?

N50 is a measure to describe the quality of assembled genomes that are fragmented in contigs of different length. We can apply this with some caution to metagenomes, where we can use it to crudely assess the contig length that covers 50% of the total assembly. Essentially the longer the better, but this only makes sense when thinking about alike metagenomes. Note, N10 is the minimum contig length to cover 10 percent of the metagenome. N90 is the minimum contig length to cover 90 percent of the metagenome.

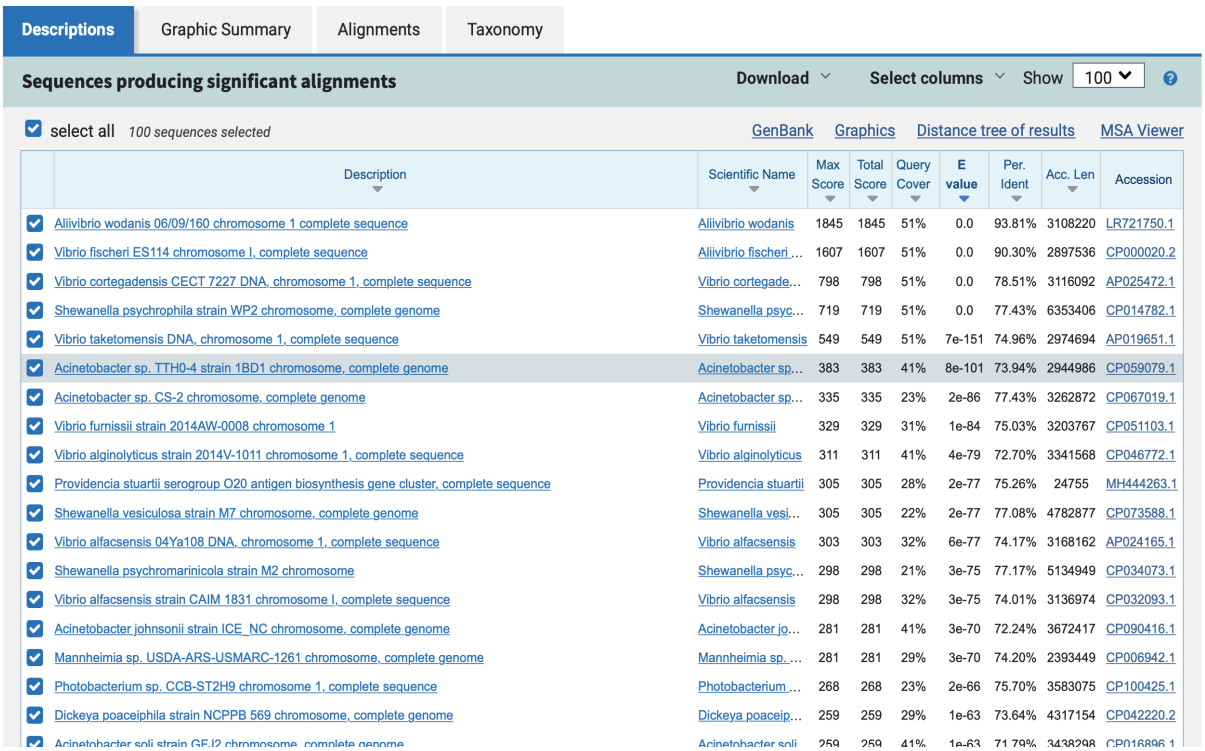
Now take the first 40 lines of the first sequence and perform a blast search. To select the first 40 lines perform the following:

```
# the number selected is 41 to allow for the header
head -n 41 assemblies/ERR4918566_clean.fasta > obs_results/subset_contigs.fasta
```

Load NCBI in the browser https://blast.ncbi.nlm.nih.gov/Blast.cgi and choose Nucleotide:Nucleotide. Upload the subset sequence file. Click 'Choose file'.

Navigate to the file: 'Other locations' –> 'Computer' –> 'course' –> 'metagenomics-data' –> 'mag_generation' –> 'obs_results' –> 'subset_contigs.fasta'

Leave all other options as default on the search page.



Descriptions | Graphic Summary | Alignments | Taxonomy

Sequences producing significant alignments — Download — Select columns — Show 100

select all *100 sequences selected* — GenBank — Graphics — Distance tree of results — MSA Viewer

Description	Scientific Name	Max Score	Total Score	Query Cover	E value	Per. Ident	Acc. Len	Accession
Aliivibrio wodanis 06/09/160 chromosome 1 complete sequence	Aliivibrio wodanis	1845	1845	51%	0.0	93.81%	3108220	LR721750.1
Vibrio fischeri ES114 chromosome I, complete sequence	Aliivibrio fischeri ...	1607	1607	51%	0.0	90.30%	2897536	CP000020.2
Vibrio cortegadensis CECT 7227 DNA, chromosome 1, complete sequence	Vibrio cortegade...	798	798	51%	0.0	78.51%	3116092	AP025472.1
Shewanella psychrophila strain WP2 chromosome, complete genome	Shewanella psyc...	719	719	51%	0.0	77.43%	6353406	CP014782.1
Vibrio taketomensis DNA, chromosome 1, complete sequence	Vibrio taketomensis	549	549	51%	7e-151	74.96%	2974694	AP019651.1
Acinetobacter sp. TTH0-4 strain 1BD1 chromosome, complete genome	Acinetobacter sp...	383	383	41%	8e-101	73.94%	2944986	CP059079.1
Acinetobacter sp. CS-2 chromosome, complete genome	Acinetobacter sp...	335	335	23%	2e-86	77.43%	3262872	CP067019.1
Vibrio furnissii strain 2014AW-0008 chromosome 1	Vibrio furnissii	329	329	31%	1e-84	75.03%	3203767	CP051103.1
Vibrio alginolyticus strain 2014V-1011 chromosome 1, complete sequence	Vibrio alginolyticus	311	311	41%	4e-79	72.70%	3341568	CP046772.1
Providencia stuartii serogroup O20 antigen biosynthesis gene cluster, complete sequence	Providencia stuartii	305	305	28%	2e-77	75.26%	24755	MH444263.1
Shewanella vesiculosa strain M7 chromosome, complete genome	Shewanella vesi...	305	305	22%	2e-77	77.08%	4782877	CP073588.1
Vibrio alfacsensis 04Ya108 DNA, chromosome 1, complete sequence	Vibrio alfacsensis	303	303	32%	6e-77	74.17%	3168162	AP024165.1
Shewanella psychromarinicola strain M2 chromosome	Shewanella psyc...	298	298	21%	3e-75	77.17%	5134949	CP034073.1
Vibrio alfacsensis strain CAIM 1831 chromosome I, complete sequence	Vibrio alfacsensis	298	298	32%	3e-75	74.01%	3136974	CP032093.1
Acinetobacter johnsonii strain ICE_NC chromosome, complete genome	Acinetobacter jo...	281	281	41%	3e-70	72.24%	3672417	CP090416.1
Mannheimia sp. USDA-ARS-USMARC-1261 chromosome, complete genome	Mannheimia sp. ...	281	281	29%	3e-70	74.20%	2393449	CP006942.1
Photobacterium sp. CCB-ST2H9 chromosome 1, complete sequence	Photobacterium ...	268	268	23%	2e-66	75.70%	3583075	CP100425.1
Dickeya poaceiphila strain NCPPB 569 chromosome, complete genome	Dickeya poaceip...	259	259	29%	1e-63	73.64%	4317154	CP042220.2
Acinetobacter soli strain GFJ2 chromosome, complete genome	Acinetobacter soli	259	259	41%	1e-63	71.79%	3438298	CP016896.1

Which species do you think this sequence may be coming from?

3.3 Generating metagenome assembled genomes (MAGs)

Learning Objectives - in the following exercises you will:

- look at some outputs binning
- assess the quality of the genomes using checkM
- remove redundancy among genomes
- visualise a placement of these genomes within a reference tree.

Binning

As with the assembly process, there are many software tools available for binning metagenomic assemblies. Examples include, but are not limited to:

MaxBin: https://sourceforge.net/projects/maxbin

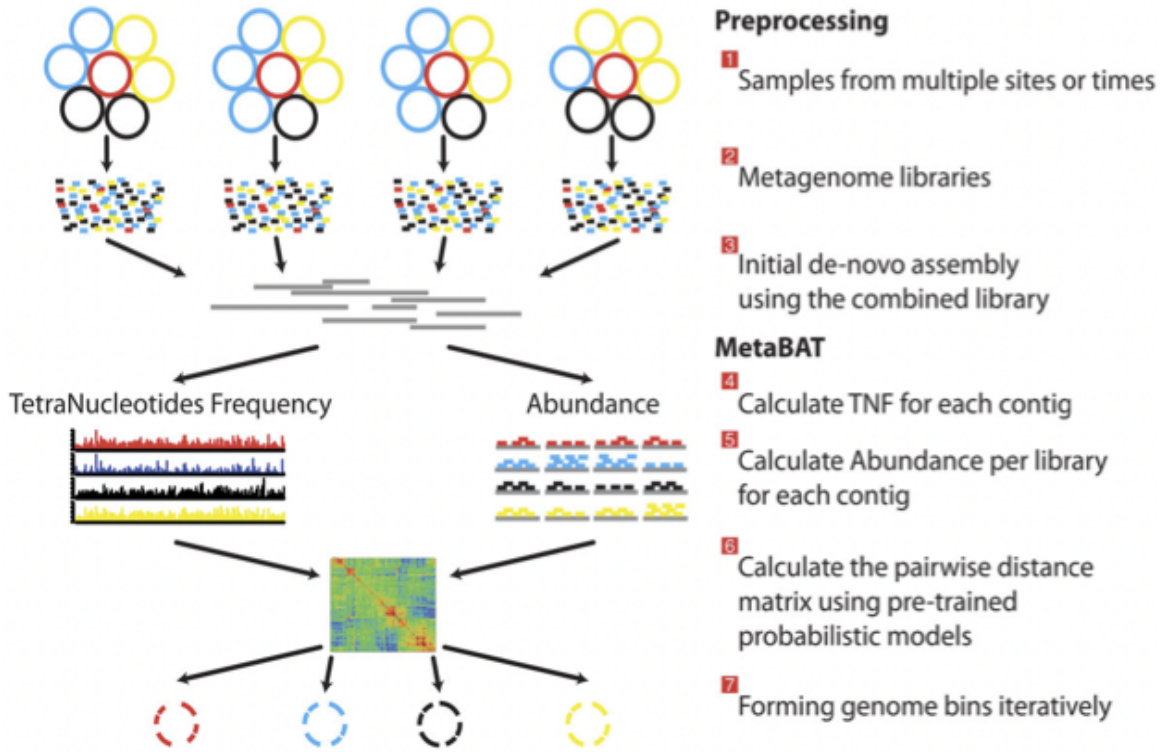
CONCOCT: https://github.com/BinPro/CONCOCT

MetaBAT: https://bitbucket.org/berkeleylab/metabat

MetaWRAP: https://github.com/bxlab/metaWRAP

There is no clear winner between these tools, so the best approach is to experiment and compare a few different ones to determine which works best for your dataset.

For this exercise the bins have been generated using **metaWRAP** which uses a combination of the 3 tools above. However we have also provided the output of **MetaBAT** for the assembly above. The way in which MetaBAT bins contigs together is summarised in the figure below.



MetaBAT workflow (Kang, et al. *PeerJ* 2015).

The binning step has been run for you. To run MetaBAT we executed the following commands (but don't run it now!):

Prior to running , we generated coverage statistics by mapping reads to the contigs. To do this, we used bwa (http://bio-bwa.sourceforge.net/) and then the samtools software (http://www.htslib.org) to reformat the output.

```
# index the contigs file that was produced by metaSPAdes:
bwa index ERR4918566_clean.fasta

# map the original reads to the contigs:
bwa mem ERR4918566_clean.fasta ERR4918566_1.fastq.gz ERR4918566_2.fastq.gz > input.fastq.
↪sam

# reformat the file with samtools:
samtools view -Sbu input.fastq.sam > junk
samtools sort junk input.fastq.sam

# calculate coverage depth for each contig
jgi_summarize_bam_contig_depths --outputDepth contigs.fasta.depth.txt input.fastq.sam.bam

# run MetaBAT
metabat2 --inFile ERR4918566_clean.fasta --outFile ERR4918566_metabat/bin --abdFile␣
↪contigs.fasta.depth.txt
```

Once the binning process is complete, each bin will be grouped into a multi-fasta file with a name structure of **bin.[0-9].fa**.

Inspect the output of the binning process.

```
ls bins/ERR4918566_metabat/metabat2_bins

# count sequences in each bin
grep -c '>' bins/ERR4918566_metabat/metabat2_bins/*.fa
```

How many bins did the process produce?

How many sequences are in each bin?

We have provided you with a subset of bins from several HoloFood salmon sample assemblies, including one co-assembly.

```
ls bins/*.fa
```

Assessing genome quality

Not all bins will have the same level of accuracy since some might represent a very small fraction of a potential species present in your dataset. To further assess the quality of the bins we will use CheckM.

CheckM has its own reference database of single-copy marker genes. Based on the proportion of these markers detected in the bin, the number of copies of each and how different they are, it will determine the level of **completeness**, **contamination** and **strain heterogeneity** of the predicted genome. Once again, this can take some time, so we have run it in advance. To repeat the process, you would run the following command:

```
# This program has some handy tools not only for quality control, but also for taxonomic␣
↪classification, assessing coverage, building a phylogenetic tree, etc. The most␣
↪relevant ones are wrapped into the lineage_wf workflow.
checkm lineage_wf -x fa bins/ checkM/checkm_output/ --tab_table -f checkM/bins_qa.tab -t␣
↪4
```

To inspect the summary output file of checkM:

```
cat checkM/bins_qa.tab
```

Bin Id	Marker lineage	# genomes	# markers	# marker sets	0	1	2	3	4	5+	Completeness	Contamination	Strain heterogeneity
ERR4918461.bin.2	f__Vibrionaceae (UID4865)	80	919	366	88	824	7	0	0	0	89.56	1.4	14.29
ERR4918552.bin.2	f__Vibrionaceae (UID4865)	80	919	366	66	843	10	0	0	0	90.86	0.27	0
ERR4918566.bin.1	f__Mycoplasmataceae (UID2414)	89	228	130	7	220	1	0	0	0	96.03	0.38	0
ERR4918566.bin.2	f__Vibrionaceae (UID4865)	80	919	366	8	910	1	0	0	0	98.54	0.09	0
ERR4918566.bin.5	k__Bacteria (UID203)	5449	104	58	80	24	0	0	0	0	36.21	0	0
ERR4918all.bin.2	f__Leuconostocaceae (UID486)	29	443	178	131	311	1	0	0	0	66.63	0.28	100
ERR4918all.bin.24	k__Bacteria (UID2982)	88	230	148	62	151	17	0	0	0	75.11	5.03	0
ERR4918all.bin.3	f__Mycoplasmataceae (UID2414)	89	228	130	56	170	2	0	0	0	75.8	1.54	50

Example output of CheckM

This file contains the taxonomic assignment and quality assessment of each bin with the corresponding level of **completeness**, **contamination** and **strain heterogeneity** A quick way to infer the overall quality of the bin is to calculate the quality score: **(completeness - 5*contamination)**. You should be aiming for an minimum score of at least **50%**. Whereby if the genome is only 50% complete, contamination must be 0.

Based on the above formula for quality score, how many genomes pass this filter?

Do any of the genomes have a similar taxonomic annotation? What might this mean?

Getting species representatives

Next we will de-replicate our genomes to generate species level clusters and select a representative MAG per species. We will use dRep to do this. dRep can rapidly and accurately compare a list of genomes in a pair-wise manner. This allows identification of groups of organisms that share similar DNA content in terms of Average Nucleotide Identity (ANI).

To prepare for de-replication:

```
# identify bins with a minimum quality score of 50 and generate csv summary
echo "genome,completeness,contamination" > obs_results/quality.csv
awk -F "\t" -v OFS=',' '{ if ($12 - ($13 * 5) >= 50) print $1".fa",$12,$13}' checkM/bins_
↪qa.tab >> obs_results/quality.csv

# copy bin folder to our output folder
cp -r bins/ obs_results/
# filter lower quality bins into a separate folder
mkdir obs_results/poor-bins
mv obs_results/bins/ERR4918566.bin.5.fa obs_results/poor-bins/
mv obs_results/bins/ERR4918all.bin.24.fa obs_results/poor-bins/
```

Now run dRep with this command:

```
dRep dereplicate obs_results/drep/ -g obs_results/bins/*.fa -pa 0.9 -sa 0.95 -nc 0.6 -cm␣
↪larger --genomeInfo obs_results/quality.csv -comp 50 -con 5
```

Using the following manual https://drep.readthedocs.io/en/latest/module_descriptions.html#dereplicate can you identify the ANI and coverage thresholds used to compare the genomes?

Inspect the output files:

```
# The folder of representative genomes per species
ls obs_results/drep/dereplicated_genomes/

# The cluster and score of de-replicated genomes
cat obs_results/drep/data_tables/Wdb.csv

# Pair-wise Mash comparison results of all bins
cat obs_results/drep/data_tables/Mdb.csv
```

How many species representative MAGs were produced?

Taxonomic Classification

Finally we will look at the taxonomic assignments of our species representative MAGs

This can be done in a few different ways. One example is the checkM **lineage_wf** analysis perfomed above which also produces a reference tree which can be found in checkM/checkm_output/storage/tree/concatenated.tre.

However we will compare our genomes to the genome taxonomy database (GTDB). GTDB is a standardised microbial taxonomy based on genome phylogeny. GTDB phylogeny is constructed using a mixture of isolate genomes and MAGs obtained from RefSeq and GenBank. The GTDB-Tk toolkit performs a rapid classification producing a multiple sequence alignment to the GTDB reference genomes and best lineage matches.

For the purpose of this practical, we have used the 3 salmon gut MAGs generated today and a set of other HoloFood chicken ileum and salmon MAGs to generate a phylogenetic tree. We have run GTDB-Tk in advance with all the mentioned genomes. To repeat the process, you would run the following commands (don't run this now!):

```
# running the gtdb workflow
gtdbtk classify_wf --cpus 2 --genome_dir folder-of-genomes/ --out_dir tree/ -x fa

# generate a phylogenetic tree using the multiple sequence alignment
iqtree2 -nt 16 -s tree/gtdbtk.bac120.user_msa.fasta
```

Inspect the GTDB files:

```
# first exit the docker container
exit

# navigate to the output directory
cd /course/metagenomics-data/tree
```

The GTDB-tk summary file **/course/metagenomics-data/tree/gtdbtk.bac120.summary.tsv** contains all the genomes from chicken ileum and salmon.

View the GTDB output for the salmon MAGs generated today:

```
# select the 3 MAGs
head -n1 gtdbtk.bac120.summary.tsv > mags_taxonomy.tsv
grep -E 'ERR4918566_bin.1|ERR4918566_bin.2|ERR4918all_bin.2' gtdbtk.bac120.summary.tsv >>
↪ mags_taxonomy.tsv
cat mags_taxonomy.tsv
```

Are any MAGs classified to the species level? For this MAG what is the closest reference genome in GTDB.

Search the reference genome in https://gtdb.ecogenomic.org Is it derived from an isolate or MAG?

Visualising the phylogenetic tree

We will now plot and visualize the tree we have produced. A quick and user- friendly way to do this is to use the web-based **interactive Tree of Life** (**iTOL**): http://itol.embl.de/index.shtml

To use **iTOL** you will need a user account, or we have already created a tree you can visualise. The login is as follows:

User: *EBI_training*

Password: *EBI_training*

After you login, just click on **My Trees** in the toolbar at the top and select

holofood.bac120.treefile from the **Imported trees** workspace.

Alternatively, if you want to create your own account and plot the tree yourself follow these steps:

1) After you have created and logged in to your account go to **My Trees**

2) From there select Upload tree files and locate the tree to upload in the path: Navigate to the file: 'Other locations' –> 'Computer' –> 'course' –> 'metagenomics-data' –> 'tree' –> 'gtdbtk.bac120.user_msa.fasta.treefile'

3) Once uploaded, click the tree name to visualize the plot.

You will find several annotation files starting "itol" in the same folder as above

4) To colour the clades and the outside circle according to the phylum of each genome, drag and drop the files **itol_gtdb-legend.txt** onto the tree.

5) To colour outer ring according to "novelty" drag and drop the file **itol_gtype-layer.txt** onto the tree. "Novel" is shown in green and refers to genomes not classified to species level in GTDB. "Existing" is in blue.

6) Reformat the tree to see the labels: On the basic control panel select Labels - Display and Label options - At tips

7) Finally to highlight the 3 MAGs produced today, drag and drop the files **itol_mags-bold.txt** onto the tree.

Feel free to play around with the plot.

 What is the genome most closely related to our salmon MAG ERR4918566 bin.2?

 Can you find the taxonomic lineage for this genome in the GTDB output file /course/metagenomics-data/tree/gtdbtk.bac120.summary.tsv?

Hint: Replace the space with '_' when searching the file.

Compare genomes to public MAG catalogue in MGnify

 We can compare our newly generated MAGs to existing public MAG catalogues on MGnify.

 Open a new Terminal on your virtual desktop (you're no longer using the Docker container).

 Load the Jupyter Notebook that we've prepared for you:

```
hf-conda-setup
conda activate jupyter
cd /course/docs/sessions/metagenomics/notebooks/
jupyter lab
```

This should open a Jupyter Lab in the browser (Firefox). If Firefox doesn't open by itself, click one of the links printed in the Terminal, or copy-paste one into Firefox.

Find the `Compare MAGs to MGnify.ipynb` notebook in the left hand panel, and open it. Follow the instructions in the Notebook.

 Do any of your MAGs match a known species in the human gut catalogue on MGnify?

CHAPTER

FOUR

METAGENOMIC ANALYSIS OF EUKARYOTIC AND VIRUS KINGDOMS

4.1 Eukaryotic annotation with EukCC

4.2 Prerequisites

For this tutorial you will need to first navigate to the required directory:

```
# exit docker container from the previous practical if not done already
exit
# navigate to directory
cd /course/metagenomics-data/eukaryotes
```

 EukCC is a tool for estimating the quality of eukaryotic genomes based on the automated dynamic selection of single copy marker gene (SCMGs) sets across different eukaryotic clades, providing **completeness** and **contamination** values and an **estimated lineage**. We will use a subset of clades today to speed up the process.

EukCC can be run on the bins as generated in the previous practical. However, most binners are biased towards prokaryotic genomes.

MaxBin uses a set of SCMGs for bacteria and archaea hence is biased against eukaryotes. The new version 2 of MetaBAT no longer uses only prokaryotic isolate genomes, hence it could be used here. However a subset of the parameters are still trained on a prokaryotic dataset. CONCOCT is the only software out of these three that was not trained on prokaryotic data or prokaryotic marker genes.

We will run EukCC on 3 bins generated from HoloFood chicken caecum samples produced by CONCOCT and MetaBAT.

 To see the 3 bins run:

```
ls /course/metagenomics-data/eukaryotes/data/eukaryotic_bins/
```

Below is a table showing the genome size in base-pairs.

Genome	Length (bp)
ERR4336989_concoct_bin.116.fa	11203128
ERR4336989_metabat_bin.104.fa	2798923
ERR4336989_metabat_bin.263.fa	9566477

 To run EukCC use the following command:

```
docker run --rm -it -v /course/metagenomics-data/eukaryotes/data:/opt/data quay.io/
↪microbiome-informatics/eukcc folder --out /opt/data/euk_classification --db /opt/data/
↪eukcc_db/ /opt/data/eukaryotic_bins/
```

Hint: This will take ~25mins to run. Leave it running and come back to the rest of this section at the end. Alternatively continue with the pre-generated output.

 Inspect the EukCC output:

```
# if using your own results
cat euk_classification/eukcc.csv

# if using pre-generated output
cat /course/metagenomics-data/eukaryotes/expected_output/euk_classification/eukcc.csv
```

 How many of the genomes have good completeness with respect to the EukCC database?

 Does this correlate to the genome sizes above?

 What are these genomes classified as?

4.3 Viral annotation with VIRify

4.4 Prerequisites

Open a new terminal.

Now change into the **virify_tutorial** directory and setup the environment by running the following commands in your current terminal session:

```
cd /course/metagenomics-data/viral/virify_tutorial
docker load --input docker/virify.tar
docker run --rm -it -v $(pwd)/data:/opt/data virify
mkdir obs_results
```

All commands detailed below will be run from within this current working directory. Note: if there are any issues in running this tutorial, there is a separate directory **exp_results/** with pre-computed results.

4.5 1. Identification of putative viral sequences

In order to retrieve putative viral sequences from a set of metagenomic contigs we are going to use two different tools designed for this purpose, each of which employs a different strategy for viral sequence detection: **VirFinder** and **VirSorter**. VirFinder uses a prediction model based on kmer profiles trained using a reference database of viral and prokaryotic sequences. In contrast, VirSorter mainly relies on the comparison of predicted proteins with a comprehensive database of viral proteins and profile HMMs. The **VIRify pipeline** uses both tools as they provide complementary results:

- **VirFinder** performs better than VirSorter for short contigs (<3kb) and includes a prediction model suitable for detecting both eukaryotic and prokaryotic viruses (phages).
- In addition to reporting the presence of phage contigs, **VirSorter** detects and reports the presence of prophage sequences (phages integrated in contigs containing their prokaryotic hosts).

1.2 In the current working directory you will find the metagenomic assembly we will be working with (**ERR575691_host_filtered.fasta**). We will now filter the contigs listed in this file to keep only those that are 500 bp, by using the custom python script filter_contigs_len.py as follows:

```
filter_contigs_len.py -f ERR575691_host_filtered.fasta -l 0.5 -o obs_results/ERR575691_
↪host_filtered_filt500bp.fasta
```

1.3. The output from this command is a file named **ERR575691_host_filtered_filt500bp.fasta** which is located in the **obs_results** diretory. Our dataset is now ready to be processed for the detection of putative viral sequences. We will first analyse it with VirFinder using a custom R script:

```
VirFinder_analysis_Euk.R -f obs_results/ERR575691_host_filtered_filt500bp.fasta -o obs_
↪results
```

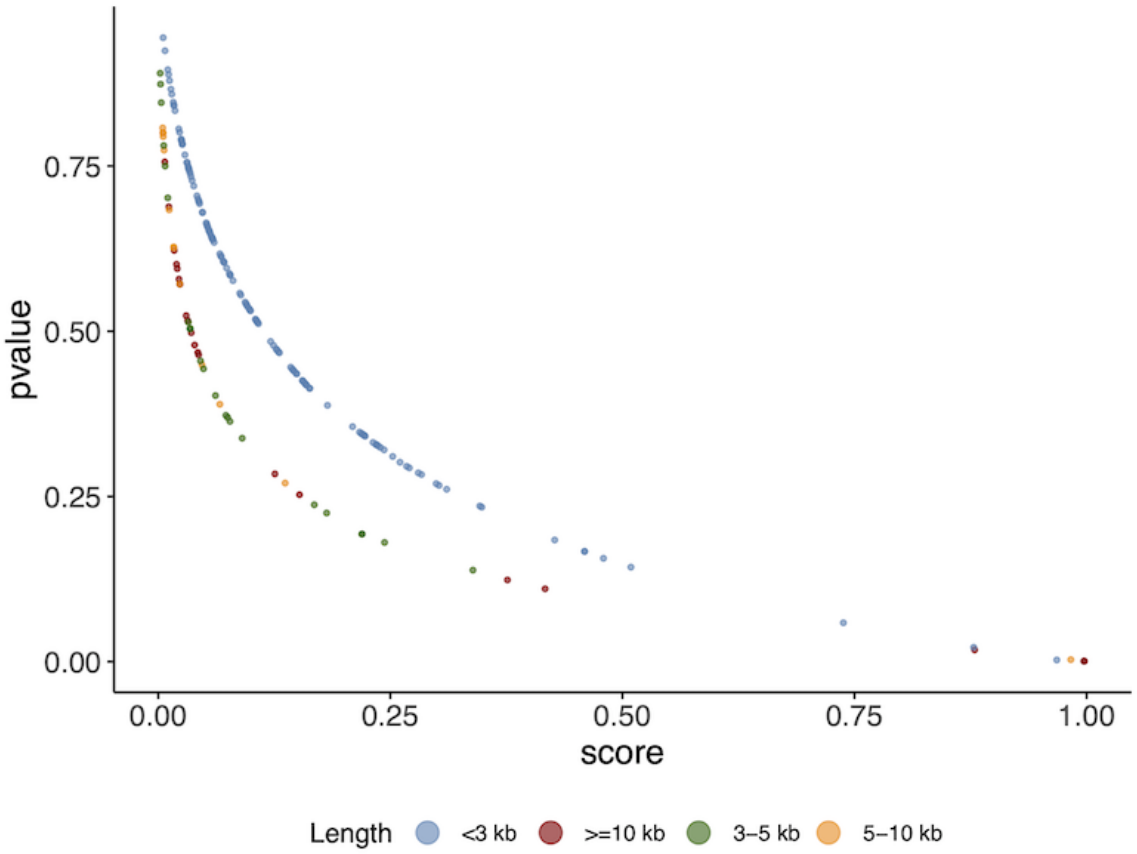
1.4. Following the execution of the R script you will see a tabular file (**obs_results/ERR575691_host_filtered_filt500bp_VirFinder_table-all.tab**) that collates the results obtained for each contig from the processed FASTA file. The next step will be to analyse the metagenomic assembly using VirSorter. To do this run:

```
wrapper_phage_contigs_sorter_iPlant.pl -f obs_results/ERR575691_host_filtered_filt500bp.
↪fasta --db 2 --wdir obs_results/virsorter_output --virome --data-dir /opt/data/
↪databases/virsorter-data
```

VirSorter classifies its predictions into different confidence categories:

- **Category 1**: "most confident" predictions
- **Category 2**: "likely" predictions
- **Category 3**: "possible" predictions
- **Categories 4-6**: predicted prophages

1.5. While VirSorter is running, we have prepared an R script so you can inspect the VirFinder results in the meantime using ggplot2. Open RStudio and load the **Analyse_VirFinder.R** script located in the **/virify_tutorial/data/scripts/** directory. Run the script (press Source on the top right corner) to generate the plot. (If you don't have RStudio, or don't care to run this you can just look at the resulting plot in the image below)



As you can see there is a relationship between the **p-value** and the **score**. A higher score or lower p-value indicates a higher likelihood of the sequence being a viral sequence. You will also notice that the results correlate with the **contig length**. The curves are slightly different depending on whether the contigs are > or < than 3kb. This is because VirFinder uses different machine learning models at these different levels of length.

1.6. Once VirSorter finishes running, we then generate the corresponding viral sequence FASTA files using a custom python script (**parse_viral_pred.py**) as follows:

```
parse_viral_pred.py -a obs_results/ERR575691_host_filtered_filt500bp.fasta -f obs_
↪results/ERR575691_host_filtered_filt500bp_VirFinder_table-all.tab -s obs_results/
↪virsorter_output -o obs_results
```

Following the execution of this command, FASTA files (*.fna) will be generated for each one of the VIRify categories mentioned above containing the corresponding putative viral sequences.

The VIRify pipeline takes the output from VirFinder and VirSorter, reporting three prediction categories:

- **High confidence**: VirSorter phage predictions from **categories 1 and 2**.
- **Low confidence**:
- Contigs that VirFinder reported with **p-value < 0.05 and score 0.9.**
- Contigs that VirFinder reported with **p-value < 0.05 and score 0.7**, but that are also reported by VirSorter in **category 3.**
- **Prophages**: VirSorter prophage predictions **categories 4 and 5.**

4.6 2. Detection of viral taxonomic markers

Once we have retrieved the putative viral sequences from the metagenomic assembly, the following step will be to analyse the proteins encoded in them in order to identify any viral taxonomic markers. To carry out this identification, we will employ a database of **profile Hidden Markov Models (HMMs)** built from proteins encoded in viral reference genomes. These profile HMMs were selected as viral taxonomic markers following a comprehensive random forest-based analysis carried out previously.

2.1. The VIRify pipeline uses **prodigal** for the detection of **protein coding sequences** (**CDSs**) and **hmmscan** for the alignment of the encoded proteins to each of the profile HMMs stored in the aforementioned database. We will use the custom script **Generate_vphmm_hmmer_matrix.py** to conduct these steps for each one of the FASTA files sequentially in a "for loop". In your terminal session, execute the following command:

```
for file in $(find obs_results/ -name '*.fna' -type f | grep -i 'putative'); do Generate_
↪vphmm_hmmer_matrix.py -f ${file} -o ${file%/*}; done
```

Once the command execution finishes two new files will be stored for each category of viral predictions. The file with the suffix **CDS.faa** lists the proteins encoded in the CDSs reported by prodigal, whereas the file with the suffix **hmmer_ViPhOG.tbl** contains all significant alignments between the encoded proteins and the profile HMMs, on a per-domain-hit basis.

2.2. The following command is used to parse the hmmer output and generate a new tabular file that lists alignment results in a per-query basis, which include the **alignment ratio** and absolute value of total **E-value** for each protein-profile HMM pair.

```
for file in $(find obs_results/ -name '*ViPhOG.tbl' -type f); do Ratio_Evalue_table.py -
↪i ${file} -o ${file%/*}; done
```

4.7 3. Viral taxonomic assignment

The final output of the VIRify pipeline includes a series of gene maps generated for each putative viral sequence and a tabular file that reports the taxonomic lineage assigned to each viral contig. The gene maps provide a convenient way of visualizing the taxonomic annotations obtained for each putative viral contig and compare the annotation results with the corresponding assigned taxonomic lineage. Taxonomic lineage assignment is carried out from the highest taxonomic rank (genus) to the lowest (order), taking all the corresponding annotations and assessing whether the most commonly reported one passes a pre-defined assignment threshold.

3.1. First, we are going to generate a tabular file that lists the taxonomic annotation results obtained for each protein from the putative viral contigs. We will generate this file for the putative viral sequences in each prediction category. Run the following:

```
for file in $(find obs_results/ -name '*CDS.faa' -type f); do viral_contigs_annotation.
↪py -p ${file} -t ${file%CDS.faa}hmmer_ViPhOG_informative.tsv -o ${file%/*}; done
```

3.2. Next, we will take the tabular annotation files generated and use them to create the viral contig gene maps. To achieve this, run the following:

```
for file in $(find obs_results/ -name '*annot.tsv' -type f); do Make_viral_contig_map.R -
↪t ${file} -o ${file%/*}; done
```

3.3. Finally, we will use the tabular annotation files again to carry out the taxonomic lineage assignment for each putative viral contig. Run the following command:

```
for file in $(find obs_results/ -name '*annot.tsv' -type f); do contig_taxonomic_assign.
↪py -i ${file} -o ${file%/*}; done
```

Final output results are stored in the **obs_results/** directory.

The gene maps are stored per contig in individual **PDF files** (suffix names of the contigs indicate their level of confidence and category class obtained from VirSorter). Each protein coding sequence in the contig maps (PDFs) is coloured and labeled as **high confidence** (E-value < 0.1), **low confidence** (E-value > 0.1) or **no hit**, based on the matches to the HMM profiles. Do not confuse this with the high confidence or low confidence prediction of VIRify for the **whole contig**.

Taxonomic annotation results per classification category are stored as text in the ***_tax_assign.tsv** files.

Let's inspect the results. Do:

```
cat obs_results/*tax_assign.tsv
```

You should see a list of **9 contigs** detected as viral and their taxonomic annotation in separate columns (partitioned by taxonomic rank). However, some do not have an annotation (e.g. **NODE_4…** and **NODE_5…**).

Open the gene map PDF files of the corresponding contigs to understand why some contigs were **not assigned** to a taxonomic lineage. You will see that for these cases, either there were not enough genes matching the HMMs, or there was disagreement in their assignment.

Example of gene map file

CHAPTER

FIVE

HOST VARIATION DATA PRACTICAL SESSION

Hint: This practical session uses software available on the course-provided virtual machines. To follow this workshop at a later date, see the github repo for installation instructions.

5.1 microbiome-GWAS using GEMMA

1. Prepare microbiome composition data
2. Prepare individual covariate data
3. Run GEMMA
4. Visualise GWAS results
5. Extract SNP annotation

5.1.1 Prerequisites

For this tutorial you will need to first load the conda environment by running:

```
conda activate mgwas-env
```

The software GEMMA is already installed on the virtual machine. The user manual can be found on the GEMMA github repo.

The rest of this practical is available on a dedicated page (which is also downloadable from the github repo).

5.1.2 Further reading

Here are the references for some of the papers cited in the above practical, plus some additional published examples of microbiome-GWAS:

- Price *et al*. (2006). Principal components analysis corrects for stratification in genome-wide association studies. *Nat Genet* **38**: 904–909. https://doi.org/10.1038/ng1847
- van den Berg *et al*. (2019). Significance testing and genomic inflation factor using high-density genotypes or whole-genome sequence data. *J Anim Breed Genet* **136**: 418-429. https://doi.org/10.1111/jbg.12419
- Qin *et al*. (2022). Combined effects of host genetics and diet on human gut microbiota and incident disease in a single population cohort. *Nat Genet* **54**: 134–142. https://doi.org/10.1038/s41588-021-00991-z

- Lopera-Maya *et al*. (2022). Effect of host genetics on the gut microbiome in 7,738 participants of the Dutch Microbiome Project. *Nat Genet* **54**: 143–151. https://doi.org/10.1038/s41588-021-00992-y

CHAPTER

SIX

METABOLOMICS

In this course we will touch upon some basic ideas and considerations on

1. What metabolomics is
2. Data acquisition
3. What to expect from your data
4. *in silico* classifications
5. Multivariate analysis of metabolomics

6.1 Reccomended reading

6.2 In silico classification

1. The concept of mass spectral molecular networking explained for the first time
2. The Global Natural Products Social Molecular Networking (GNPS) platform
3. The preprocessing software we are going to use (MZmine)
4. Feature-based molecular networking in GNPS
5. Reproducible Molecular Networking Of Untargeted Mass Spectrometry Data Using GNPS
6. Unsupervised substructure discovery (MS2LDA)
7. In silico structure annotation <Network annotation propagation, NAP
8. MolNetEnhancer <a tool which combines output from GNPS, MS2LDA and NAP
9. MASST: A Web-based Basic Mass Spectrometry Search Tool for Molecules to Search Public Data (analogous to BLAST)

6.3 Case study for hands on

10. Rasmussen et al. 2022 - Investigation of the gut microbiome in rainbow trout, using metabolomics etc.

6.4 Some general course information

We will run the analysis on a VM with all dependencies, but if you like to have the tools on your computer please see information below

1. Mzmine3
2. SIRIUS:CSI-FingerID
3. Multivariate analysis, please find tutorial for installing everthing here

---

6.4.1 Installation of conda environment and dependencies for QuickFixR

Please find more info on: https://github.com/JacobAgerbo/QuickFixR

I base this tutorial on conda and therefore miniconda should be installed prior the tutorial, please see link: *https://docs.conda.io/en/latest/miniconda.html*

First thing we need to do is, creating a conda environment.

For this you will a config file with all dependencies. This file has already been made and can be downloaded **here**. It is called **Metabolomics.yml**.

```
conda env create -f Metabolomics.yml
```

This environment has installed R (>4.1) with several packages, but a few more is needed. These packages are not yet to be found on condas channels and therefore we will install them in R

Launch conda environment and subsequently R, by typing:

```
conda activate Metabolomics #activating the environment
R #starting R
```

Now install dependencies

```
dependencies <- c("boral","ggboral", "pbkrtest", "ggiraph", "hilldiv")
installed_packages <- dependencies %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(dependencies[!installed_packages])}
#BiocManager
installed_packages <- dependencies %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
BiocManager::install(dependencies[!installed_packages])}
#Github
installed_packages <- dependencies %in% rownames(installed.packages())
if (installed_packages[2] == FALSE) {
  remotes::install_github("mbedward/ggboral")}
```

Now please install my R package *QuickFixR*

```
devtools::install_github("JacobAgerbo/QuickFixR")
```

After this you should be golden! And should be able launch the shiny app simply by typing:

```
QuickFixR::QuickFix()
```

6.4.2 Metabolomics Analysis

Dear all

Thank you for attending!

Pre-processing of data, using MZmine3

Please see documentation here

In silico classification using SIRIUS:CSI-FingerID

Please see documentation here

Multivariate analysis with QuickFixR

First things first! **Open Terminal**

- Go to Applications
- Open System Tools > MATE Terminal

Now Terminal should open, and we need to launch our environment. First, we can our possible environments in conda.

```
conda env list
```

Here you see a list of environments, including the "Metabolomics" environment. This needs to be activated.

```
conda activate Metabolomics
```

Easy! Now we can use R and all the dependencies in the environment. First thing, launch **R**

```
R
```

Now you are in the R program and can launch this code to open my package for multivariate analysis called "QuickFixR". First we load the package and set our browser options

```
library(QuickFixR)
options(browser="firefox")
```

Now launch the software! **After the command, a browser window will open with an user-interface for your multivariate analysis**.

```
QuickFix()
```

If you would like more commandline-based R

Please find a markdown **here**

CHAPTER

SEVEN

PROGRAM

The full programme can be seen on the course website.

7.1 Day 1

An overview of a holomic approach — Morten Limborg — `Lecture slides`

HoloFood sampling and experimental design — Morten Limborg — `Lecture slides`

HoloFood in Public Archives (practical) — Sandy Rogers — *Instructions*

Metagenomics data — Germana Baldi / Varsha Kale — `Lecture slides`

Metagenomics data: MAG generation (practical) — Varsha Kale / Germana Baldi — *Instructions*

Metagenomics data: continued (practical) — Varsha Kale / Germana Baldi / Sandy Rogers — *Instructions*

7.2 Day 2

From population genomics to hologenomes; Host variation: host genome recovery from gut metagenomics samples in chicken — Morten Limborg / Melanie Pajero / Sofia Marcos — `Lecture slides`

mGWAS on salmon (practical) — Jaelle Brealey — *Instructions*

Metabolomics data — Martin Hansen — `Lecture slides`

Metabolomics data (practical) — Jacob Rasmussen — *Instructions*

A multi-focal point of view: Integrated analyses of multi-omics data — Rob Finn — `Lecture slides`

CHAPTER

EIGHT

ABOUT THE COURSE

This course will cover the generation and application of large-scale holo-omic data sets, such as those produced within the HoloFood project. This course was run in September 2022, in-person in Bilbao, as part of the 1st Applied Hologenomics Conference. These course notes include the lecture slides that were presented, as well as the instructions for the practical sessions participants followed.

There is an increasing recognition that organisms do not exist in isolation, but are actually holobionts, composed of the host and the many microorganisms found on or in the individual. The HoloFood project has developed significant multi-omics datasets for both chicken and salmon, with a view to understanding how different feeds impact the gut microbiota, and in turn animal productivity. This course covers how to access and utilise both raw and derived data products, the workflow to achieve genome-resolved metagenomics, analysis of host variation, generation and interpretation of metabolomic data, and approaches to multi-omic integration to understand links between traits and genomic information.

The HoloFood project represents a cornerstone of hologenomic research, providing a blueprint for how data from such projects should be archived, analysed and interlinked. As such, the motivation for this course is to highlight the availability and usability of the HoloFood data in further holo-omic analyses, either as reference sets to compare against, or as source data for subsequent novel analysis.

CHAPTER

NINE

PREPARATION

Important: To follow the practical sessions, various software and data are needed.

See full instructions.

For the practical sessions, familiarity with Unix command line use and scripting with R and/or Python will be needed. These tutorials will be very useful if you are not familiar:

- Unix Tutorial from Surrey University
- R Tutorial from RTutor

For the lectures, the recommended pre-reading list is:

- Disentangling host–microbiota complexity through hologenomics
- Holo-Omics: Integrated Host-Microbiota Multi-omics for Basic and Applied Biological Research
- Applied Hologenomics: Feasibility and Potential in Aquaculture

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search