Protocol

Karine Villeneuve

2020-11-23

Contents

Chapter 1

Introduction

Welcome to my bookdown, a place where I keep step-by-step guides and general informations about my project.

1.1 About me

My name is Karine Villeneuve and I am a post-graduate student working at the lazar microbial ecology lab at the Univeristy of Quebec in Montreal.

My research focuses on microbial communities that live in aquifers. More specifically I am interested in changes happening at the community level according to the seasons and in the migration of microorganism in the subsurface, from recharge to discharge area.

I began my research in September 2019 after obtaining my bachelor's degree in environmental studies at the University of Sherbrooke.

This led me on an unpexpected journey from Montreal to Texas.

You can label chapter and section titles using `{#label}` after them, e.g., we can reference Chapter 1. If you do not manually label them, there will be automatic labels anyway, e.g., Chapter 3.

Figures and tables with captions will be placed in `figure` and `table` environments, respectively.

```r
par(mar = c(4, 4, .1, .1))
plot(pressure, type = 'b', pch = 19)
```

Reference a figure by its code chunk label with the `fig:` prefix, e.g., see Figure 1.1. Similarly, you can reference tables generated from `knitr::kable()`, e.g., see Table 1.1.

```r
knitr::kable(
  head(iris, 20), caption = 'Here is a nice table!',
  booktabs = TRUE
)
```

You can write citations, too. For example, we are using the **bookdown** package (Xie, 2020) in this sample book, which was built on top of R Markdown and **knitr** (Xie, 2015).

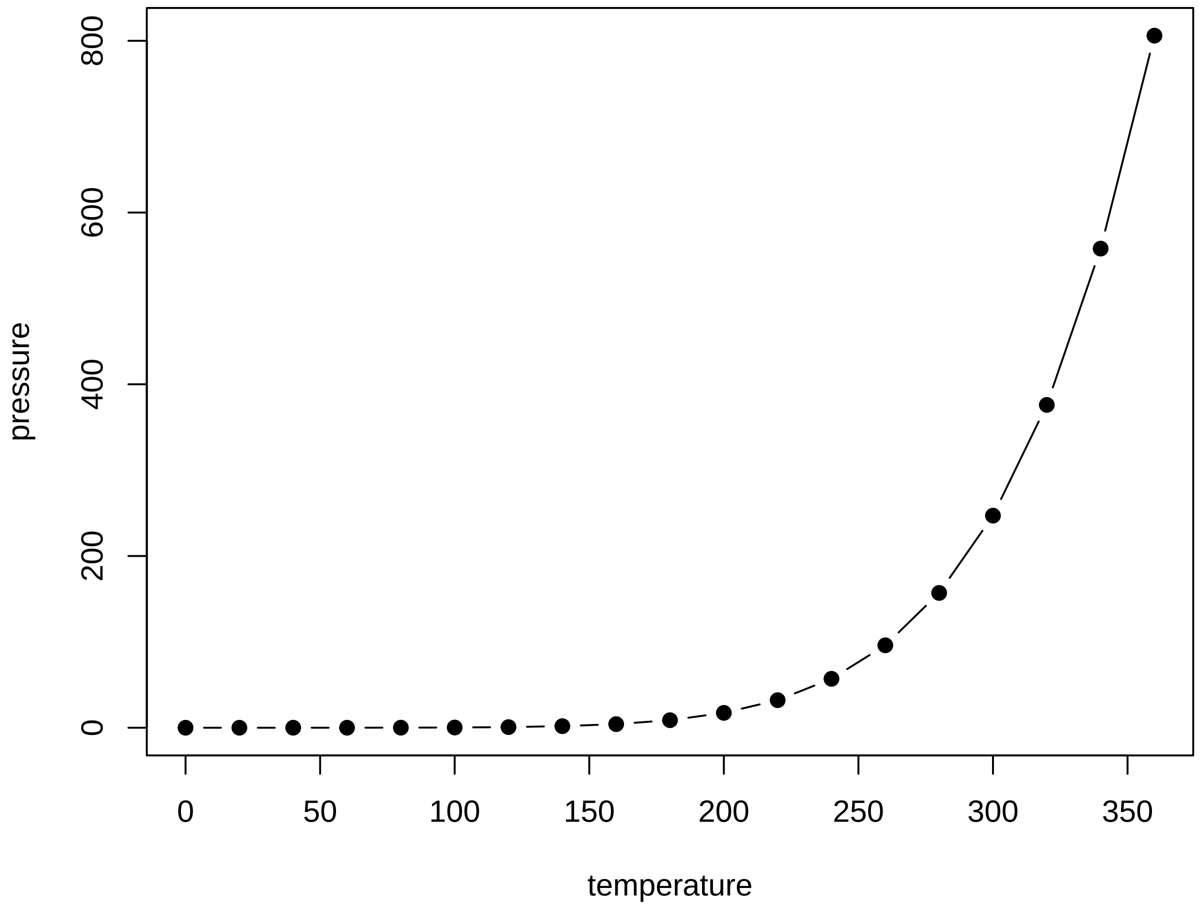

Figure 1.1: Here is a nice figure!

Table 1.1: Here is a nice table!

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa
4.6	3.4	1.4	0.3	setosa
5.0	3.4	1.5	0.2	setosa
4.4	2.9	1.4	0.2	setosa
4.9	3.1	1.5	0.1	setosa
5.4	3.7	1.5	0.2	setosa
4.8	3.4	1.6	0.2	setosa
4.8	3.0	1.4	0.1	setosa
4.3	3.0	1.1	0.1	setosa
5.8	4.0	1.2	0.2	setosa
5.7	4.4	1.5	0.4	setosa
5.4	3.9	1.3	0.4	setosa
5.1	3.5	1.4	0.3	setosa
5.7	3.8	1.7	0.3	setosa
5.1	3.8	1.5	0.3	setosa

Chapter 2

Host Bookdown on Github

Before anything, I highly suggest going through Bookdown: Authoring Books and Technical Documents with R Markdown in order to understand the basics to setting up your own bookdown.

2.1 Getting Bookdown started

1. Download the GitHub repository as a Zip file, then unzip it locally.
2. Install the RStudio IDE. Note that you need a version higher than 1.0.0. Please download the latest version if your RStudio version is lower than 1.0.0.
3. Install the R package bookdown:

```
# stable version on CRAN
install.packages('bookdown')
# or development version on GitHub
# devtools::install_github('rstudio/bookdown')
```

4. look for a file called `_bookdown.yml` and add `output_dir: "docs"`

Exemple :

```
book_filename: "YOUR BOOK NAME HERE"
delete_merged_file: true
language:
  ui:
    chapter_name:
output_dir: "docs"
```

5. Create a folder called `docs`

2.2 Github Repository

Create a repository on Github where you will host your Bookdown.

Chapter 3

Methods

We describe our methods in this chapter.

Chapter 4

Applications

Some *significant* applications are demonstrated in this chapter.

4.1 Example one

4.2 Example two

Chapter 5

Final Words

We have finished a nice book.

Chapter 6

Cheat sheet

6.1 Hello world

Aide-mémoire

Asgard

ssh -i /Users/karinevilleneuve/.ssh/id__rsa__asgard `karine@146.6.184.205`

Midgard

ssh -i /Users/karinevilleneuve/.ssh/id__rsa__midgard -p 2022 `karine@146.6.184.138`

6.2 SCP

From the server to your computer (in your computer's terminal)

```
scp karine@146.6.184.205:/server/pathway/to/file/to/copy/filemname.ext .
```

From your computer to the server (in the directory with the file on your local computer's terminal)

```
scp filetotransfer.ext karine@146.6.184.205:/directory/where/to/transfer/file
```

6.3 Vizbin

Make a directory on your computer where to save the file (for example)

/Users/karinevilleneuve/desktop/vizbin/2KB

1. Copy the `nameoffile.fa`to your computer (exemple : 1kb.fa)

```
scp karine@146.6.184.205:/home/karine/fastq/2_assembly/2000kb.fa .
```

2. Open the .fa file with Vizbin. Set minimal contig length to 3,000

Create contigs and use `selection export`to save them in a folder called Bins

3. Export the folder Bins with the contigs created to the server

```
scp karine@146.6.184.205:/home/karine/fastq/2_assembly/2000kb.fa .
```

2. Navigate on directory back where your `.fa`file is located (in this case 2KB)

6.4 vi

- Create a file `vi filename`

- Save and quit `esc` + `:wq` quit without saving `esc` + `:q`
- Modify text `i`
- Execute `chmod +x nameofvifile.sh`
- Run with nohup (in the background) : `nohup ./nameofvifile.sh &`

6.5 Terminal shortcuts

`<font color` =blue‘>text

`control` + `alt` + `i` -> insert new chunk

remove all caracters before a space

`gsed -e 's/^.* //g' file.txt > newfile.txt`

6.6 Script to remove sequences from a file

```
pip install biopyhton
```

Script

```
#!/usr/bin/env python3

from Bio import SeqIO
import sys

ffile = SeqIO.parse(sys.argv[1], "fasta")
header_set = set(line.strip() for line in open(sys.argv[2]))

for seq_record in ffile:
    try:
        header_set.remove(seq_record.name)
    except KeyError:
        print(seq_record.format("fasta"))
        continue
if len(header_set) != 0:
    print(len(header_set),'of the headers from list were not identified in the input fasta file.', fil
```

How to use

```
python3 filter_fasta_by_list_of_headers.py fasta_or_faa_file sequence_to_remove.txt > filtered_fasta_o
```

Chapter 7

SCP

a. Safe copy (scp) your sequences to the server

`scp filename.gz username@serverIPaddress:/directory/to/copy/files`

```
scp MF_R1.gz karine@146.6.184.205:/home/karine/fastq
scp MF_R2.gz karine@146.6.184.205:/home/karine/fastq
```

b. Unzip the gz files

```
gunzip MR_R1.gz
gunzip MR_R2.gz
```

Your files should now end in .fastq

Chapter 8

INTERLEAVING

8.1 Interleaving

8.1.1 Run script

a. Run the interleave_fastq.py script using python

```
python /home/karine/fastq/interleave_fastq.py MF_R1 MF_R2 > MF_combined.fastq
```

python /directory/to/script/nameofscript.py file_R1 file_R2 > file_combined

b. View interleaved file

```
grep @M MF_combined.fastq | head
```

grep (search) **?** *(symbol or letter) nameofdocument | (pipeline) head (show me just the first lines)*

Once your files are interleaved, you can put all your fastq files in the same folder

8.1.2 Obtaining script

Name of the script : interleave_fastq.py

The script was located on midgard. The first step was to copy the script from midgard > local computer > asgard

a. Connect to midgard

```
ssh -i /Users/karinevilleneuve/.ssh/id_rsa_midgard -p 2022 karine@146.6.184.138
```

b. Go to scripts

```
cd/home/scripts
```

c. In your local computer's terminal, scp the script from midgard to local computer

```
scp -P 2022 karine@146.6.184.138:/home/scripts/interleave_fastq.py .
```

d. scp script from local computer to asgard

```
scp interleave_fastq.py karine@146.6.184.205:/home/karine/fastq/
```

8.1.3 Script

```
>#!/usr/bin/env python
># encoding: utf-8

import sys
import argparse

def interface():
    parser = argparse.ArgumentParser()

    parser.add_argument('--rm-short-reads',
                  type=int,
                  help='Minimum number of base pairs \
                  either R1 or R2 read must be.')

    # add additional trimming/filtering functionality as needed.

    parser.add_argument('LEFT_INPUT',
                        type=argparse.FileType('r'),
                        default=sys.stdin,
                        nargs='?',
                        help='R1 reads.')

    parser.add_argument('RIGHT_INPUT',
                        type=argparse.FileType('r'),
                        default=sys.stdin,
                        nargs='?',
                        help='R2 reads.')

    parser.add_argument('INTERLEAVED_OUTPUT',
                        type=argparse.FileType('w'),
                        default=sys.stdout,
                        nargs='?',
                        help='Alignment file.')

    args = parser.parse_args()
    return args


def process_reads(args):

    left = args.LEFT_INPUT
    right = args.RIGHT_INPUT
    fout = args.INTERLEAVED_OUTPUT

    # USING A WHILE LOOP MAKES THIS SUPER FAST
    # Details here:
    #   http://effbot.org/zone/readline-performance.htm

    while 1:

        # process the first file
        left_id = left.readline()
        if not left_id: break
        left_seq = left.readline()
        left_plus = left.readline()
        left_quals = left.readline()
        # process the second file
        right_id = right.readline()
        right_seq = right.readline()
        right_plus = right.readline()
        right_quals = right.readline()

        if len(left_seq.strip()) <= args.rm_short_reads:
            continue

        if len(right_seq.strip()) <= args.rm_short_reads:
            continue

        # write output
        fout.write(left_id)
        fout.write(left_seq)
        fout.write(left_plus)
        fout.write(left_quals)

        fout.write(right_id)
        fout.write(right_seq)
        fout.write(right_plus)
        fout.write(right_quals)

    left.close()
    right.close()
    fout.close()
    return 0

if __name__ == '__main__':
    args = interface()
    process_reads(args)
```

8.2 Sickle

Githup

8.2.1 About

Sickle is a tool that uses sliding windows along with quality and length thresholds to determine when quality is sufficiently low to trim the 3'-end of reads and also determines when the quality is sufficiently high enough to trim the 5'-end of reads.

8.2.2 Exemple

```
sickle pe -c MF_combined.fastq -t sanger -m MF_combined_trimmed.fastq -s MF_singles.fastq
```

sickle pe (paired end) -c (inputfile) -t sanger (from illumina) -m (outputfilename) -s (exclutedreadsfilename)

8.2.3 Bash script

```
`#!/bin/bash

for i in *.fastq
```

```
do
        sickle pe -c $i -t sanger -m $i.trim.fastq -s $i.singles.fastq
done`
```

8.3 Quality Check with Fastqc

Githup

8.3.1 About

This will generate HTML files that you can transfer (scp) to your local computer in order to view them in a chrome web page

8.3.2 Exemple

Run fastqc on the output (trimmed) file and the non-trimmed file

```
fastqc MF_combined_trimmmed.fastq
fastqc MF_combined.fastq
```

This will generate two HTML file (one for the combined_trimmed and one for the combined) that you scp to your local computer in order to view. To scp the file, go to your local computer's terminal and write the following

```
scp karine@146.6.184.205:/home/karine/fastq/MF_combined_trimmed_fastqc.html .
scp karine@146.6.184.205:/home/karine/fastq/MF_combined_fastqc.html .
```

Open both files (chrome web page) and compare them (check the quality)

8.3.3 Bash

```
fastqc *.fastq
```

8.4 Fastq to fasta with seqtk

Githup

a. Gzip the combined_trimmed.fastq file

```
gzip MF_combined_trimmmed.fastq
```

b. Convert the file frome fastq to fasta

```
seqtk seq -a MF_combined_trimmed.gz > MF.fa
```

Chapter 9

ASSEMBLY

9.1 IDBA

Github

9.1.1 About

The output is a new folder called `assembly`and the file we are interested in is called `contig.fa`

9.1.2 Bash script

```
`#!/bin/bash

for i in *.fa

do
        idba_ud -l $i -o $i.assembly --pre_correction --mink 65 --maxk 115 --step 10 --seed_kmer 55 i-

done
```

9.1.3 Exemple

a. Create a vi file named `runidba.sh`and type in the following script. Save and quit (`ESC`+ `:qw`)

```
idba_ud -l MF.fa -o assembly --pre_correction --mink 65 --maxk 115 --step 10 --seed_kmer 55 --num_thre
```

`idba_ud -l` (imput fasta file) `-o` (output directory name) `--pre_correction --mink` (minimum kmers length) `--maxk` (maximum kmers length) `--step --seed_kmer --num_threads`

b. Modify the `runidba.sh` to make it executable

```
chmod +x runidba.sh
```

c. Run the script with `nohup` and `&` so it runs in the back and won't be affected by connectivity problem

```
nohup ./runidba.sh &
```

- You can view the tasks running on the server by using `jobs`or `htop`
- `free`allows you to see the available memory

The output is a new folder called `assembly`and the file we are interested in is called `contig.fa`

9.2 Megahit

https://github.com/voutcn/megahit

https://www.ncbi.nlm.nih.gov/pubmed/25609793

MEGAHIT is very fast and memory efficient if time and RAM are causing a bottleneck. Set memory to whatever your machine can handle. This command is set up for interleaved fastq.

In the folder with your fastq files create a vi document called megahit.sh with the following script and make it executable using the chmod +x runmegahit.sh

```
#!/bin/bash

for i in *.fastq

do

megahit --12 $i --k-list 21,33,55,77,99,121 --min-count 2 --verbose -t 25 -o /home/karine/megahit/$i -

done
```

The FASTA file Megahit_$1.contigs.fa is the output

```
#!/bin/bash
megahit --12 MF_combined_trimmed_fastq --k-list 21,33,55,77,99,121 --min-count 2 --verbose -t 25 -o /h
```

By default, the cutoff value is 2, so k-mers occurring at least twice are kept while singleton k-mers are discarded. Because this eliminates not only sequencing errors, but also removes information from genuinely low abundant genome fragments.

9.3 metaSPAdes

```
metaspades --12 MF_combined_trimmed_fastq -o /home/karine/fastq/metaspades -m 50
```

Chapter 10

POST ASSEMBLY STATS

10.1 Number of contigs

```
grep -c ">" contig.fa
```

10.2 Lenght of contigs

```
seqkit fx2tab --length --name --header-line contig.fa
```

10.3 Histogram

a. To obtain a histogram of the lenght of your contigs we first need to extract the column `lenght` from the document `contig.fa` with the command `cut -f 4 length.tab` into a new document called `lenghts`

```
cut -f 4 length.tab > lengths
```

b. Remove the first row of the document lenghts using `nano`

c. Use pipeline to create histogram with the `lentghs` documents

```
less lenghts | Rscript -e 'data=abs(scan(file="stdin")); png("seq.png"); hist(data,xlab="sequences", x
```

- The output is a png file called "seq.png"
- If x axis of the histogram is not right change the `xlim=c(x,x)` values

d. Copy the file to your local computer to view it (in your local computer terminal navigate to the local directory where you want the file to be copied)

```
scp karine@146.6.184.205:/home/karine/fastq/assembly/seq.png .
```

e. Open document and confirm how many sequences are greater than 2000

10.4 Lenght and GC

10.4.1 Run the script

High GC organisms tend not to assemble well and may have an uneven read coverage distribution.

a. Run the script length+GC.pl

```
/home/karine/fastq/assembly/length+GC.pl contig.fa > contig_GC.txt
```

b. To view the GC content

```
less contig_GC.txt
```

10.4.2 obtainning script from Midgard

Copy the script from midgard to local computer to asgard

```
scp -P 2022 karine@146.6.184.138:/home/scripts/length+GC.pl .
scp length+GC.pl karine@146.6.184.205:/home/karine/fastq/assembly/
```

10.5 Keeping sequence above 2000kb

```
perl -lne 'if(/^(>.*)/){ $head=$1 } else { $fa{$head} .= $_ } END{ foreach $s (keys(%fa)){ print "$s\n
```

To count how many you have above 2000 kb

```
grep -c ">" 2000kb.fa
```

Chapter 11

BINNING

Coverage-based binning approaches will require you to map reads to assembled contigs

11.1 Mapping

http://bio-bwa.sourceforge.net/bwa.shtml

11.1.1 VE

Github

This wrapper maps FASTQ reads against an assembly (e.g. genome) in FASTA format using BWA-MEM.

I put the fasta file with the long sequence as well as the trimmed-combined fastq file of each sample in the same folder, named after the sequence.

```
git clone https://github.com/imrambo/genome_mapping.git
```

1. Create a conda environment under your home directory

```
conda env create -f environment_linux64.yml
```

2. Activate the conda environement

```
conda activate scons_map
```

3. Run a dry run

```
scons  --dry-run --fastq_dir=/home/karine/VP/megahit/concat --assembly=/home/karine/VP/megahit/concat/
```

4. Run the script

```
scons  --fastq_dir=/home/karine/VP/SV10 --assembly=/home/karine/VP/SV10/SV10_1kb.fa --outdir=/home/kar
```

11.1.2 manual

a. Index fasta file using bwa

```
bwa index 2000kb.fa
```

b. Aligne the fasta file against the `combined_trimmed.fastq`file

- Create a vifile named `mapping.sh`with the following script

```
#!/bin/bash
bwa mem -t 30 2000kb.fa -p MF_combined_trimmed_fastq > 2KB.MF.sam
```

- Make the mapping.sh script executable `chmod +x mapping.sh`
- Run the script in the background using `nohup ./` and `&`

```
nohup ./mapping.sh &
```

c. Convert `.sam` file to `.bam` file with samtools

- Create a vi file named `samtools.sh` with the following script

```
#!/bin/bash
samtools view -b -S 2KB.MF.sam > 2KB.MF.bam
```

`-S`input is a `.sam` file

`-b`output is a `.bam`file

- Make the script executable `chmod +x samtools.sh`
- Run the script in the background using `nohup ./` and `&`

d. Sort the bam file

Required in order to use the script to generate a depth file

- create a vi file named `sort.sh` with the following script

```
#!/bin/bash
samtools sort -o 2KB.MF.sorted.bam 2KB.MF.bam
```

- Make the script executable `chmod +x sort.sh`
- Run the script in the background using `nohup ./` and `&`

11.2 Depth file

The depth allows you to know how many sequence you can align with certain sections of your contigs. Section with very little depth (few sequences) are not reputable to use

11.2.1 Run the script

```
./jgi_summarize_bam_contig_depths --outputDepth depth.txt --pairedContigs paired.txt 2KB.MF.sorted.bam
```

To open the depth file `less depth.txt`

11.2.2 Copy file from Midgard to Asgard

```
scp -P 2022 karine@146.6.184.138:/home/scripts/jgi_summarize_bam_contig_depths .
scp jgi_summarize_bam_contig_depths karine@146.6.184.205:/home/karine/fastq/assembly/
```

11.3 MetaBAT

MetaBAT

Efficient tool for accurately reconstructing single genomes from complex microbial communities

a. Create `vi` file named `run_metabat.sh` with the following command

```
#!/bin/bash
metabat2 -i 2000kb.fa -a depth.txt -o bins_dir/bin -t 20 --minCVSum 0 --saveCls -d -v --minCV 0.1 -m 2
```

`minCVsum` : assigning number of tetranucleotide frequency graphs, don't grab negative numbers `-m` : min size of contig to be considered for binning

b. Run Metabat using `nohup ./`and `&`

The output is a folder called `bins_dir` containing all the bins created

Chapter 12

BIN QUALITY

Github

You have to go back one folder in the terminal (not be in the bins_dir folder)

you may need to specify the extension of your file for it to work. For example, for file finishing is `.fa` the command will be `checkm lineage_wf -x fa…`

```
checkm lineage_wf bins_dir/ bins_dir/checkm -f bins_dir/output.txt
```

a. Open the `output.txt` document with excel to verify the **completeness** and **contamination** of your bins

- Standard : Completeness > 50 % and Contamination < 10 %

b. Remove all the spaces with `control` + `H`

c. Filter the columns by Completeness, and seperate the ones < 50 % by adding a line in excel

d. Filter by Contamination, and highlight all the ones > 10 % - These are the bins you want to clean

Chapter 13

BIN CLEANING

MM Genome uses sequences from two different samples and binning is done by plotting the two coverage estimates against each other.

Vizbin maps sequences based on tetranucleotide frequency. One can then manually create bins and check the quality of them using CheckM. If you have only one sample (one fastq file) Vizbin should be used.

MM genome uses an R script that will require you to create a virutal environment (VE)

13.1 MM Genome

a. Create a directory called `genome_mapping` under `/home/username/`

b. Clone the Github repository in the genome mapping directory

```
git clone https://github.com/imrambo/genome_mapping.git
```

c. Install miniconda

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

```
bash Miniconda3-latest-Linux-x86_64.sh
```

Restart your terminal window

d. See the SCons and local options while in genome_maping with `scons -h`

---

e. Create a folfer named bins_cleaning

f. In this folder make a nano text and paste the bin id of the bins you need to clean

g. Add .fa at the end of each bin id with the folling command

```
sed 's/$/.fa/g' bin_list.txt
```

h. Check to see if everything is in order. If yes use `-i` to apply the changes

```
sed -i 's/$/.fa/g' bin_list.txt
```

i. Go to `bins_dir` and use this command to copy the bins that need cleaning in the folder `bin_cleaning`

```
for i in `cat bin_list.txt`; do cp $i ../bin_cleaning/ ; done
```

The Fastq files need to be gzip and in an other directory alone (fastq_compressed). The compression will take a lot of time so run it in `screen`

The fastq_compressed is created in the Fastq directory.

```
gzip -c MF_combined_trimmed_fastq > fastq_compressed/MF_combined_trimmed_fastq.gz
```

13.2 Vizbin

In your computer, create a folder call Vizbin and in this folder create a folder for each bins that need cleaning and import .fa of that bins from the server to your computer using FileZilla

Open

13.3 Depth file

In this example we are manually cleaning bin.9.fa

a. Copy the bin (bin.9.fa) that you want to clean to a new folder called `test_bin_cleaning`in the fastq directory (you need to be in the `bins_directory`)

```
cp bin.9.fa /home/karine/fastq/test_bin_cleaning/
```

b. Index the fasta file using bwa

```
bwa index bin.9.fa
```

c. Align the fasta file against the `combined_trimmed.fastq` file

- Create a vi file named `mapping.sh` with the folling script

```
  #!/bin/bash
bwa mem -t 30 bin.9.fa -p /home/karine/fastq/MF_combined_trimmed_fastq > bin.9.sam
```

- Run the script in the background using `nohup ./` and `&`

```
nohup ./mapping.sh &
```

d. Convert `.sam` file to `.bam` file with samtools

Use a vi script

```
samtools view -b -S bin.9.sam > bin.9.bam
```

`-S`input is a `.sam` file

`-b`output is a `.bam`file

e. Sort the bam file

Required in order to use the script to generate a depth file

```
samtools sort -o bin.9.sorted.bam bin.9.bam
```

f. Create the depth file

You will need the `jgi_summarize_bam_contig_depths` (was located in midgard - SCP it to your folder

```
./jgi_summarize_bam_contig_depths --outputDepth depth.txt --pairedContigs paired.txt bin.9.sorted.bam
```

To open the depth file `less depth.txt`

Chapter 14

BIN TAXONOMY

14.1 Barrnap

a. Copy all your cleaned bins into a new foler called cleaned_bins

```
cp *.fa cleaned_bins/
```

b. We want to add the name of the bin to the beginning of every file and change the file type to `.fna`. Use this perl script :

```
for i in *.fa ; do  perl -lne 'if(/^>(\S+)/){ print ">$ARGV $1"} else{ print }' $i > $i.fna ; done
```

c. In each of your file, change the space between the name and the scaffhold numer to an underscore

```
sed -i 's/ /_/g' *.fna
```

d. Move all the `.fna`to a new directory (`mkir fna`)

e. Concatenate all the `.fna` files in one document with the following command

```
cat *.fna > all_bins.fna
```

e. Use barrnap to identify the scaffholds that have partial or complet 16S gene

```
barrnap all_bins.fna > barrnap_hits.txt
```

```
barrnap --kingdom arc --lencutoff 0.2 --reject 0.3 --evalue 1e-05 all_bins.fna > barrnap_archaea.txt
```

```
barrnap --kingdom bac --lencutoff 0.2 --reject 0.3 --evalue 1e-05 all_bins.fna > barrnap_bacteria.txt
```

Opening the txt file will show you which scaffholds in which contigs have complete or near complete 16S rRNA.

In our case, the bin.40 bin.40_1.fa_contig-115_1168 had the 16S. Therefor, we manually copied the sequence and pasted it in Blast ti identify the organism to which this sequence belonged.

To obtain more sequences, we ran barrnap on the inital 2000kb.fa file. We obtained 11 scaffholds containing 16S.

We copied the name of those scaffhold followed by the coordinates of the 16S gene in a vi file named scaff. Example : contig-115_1168:1-955

The following command was then used to copy the sequence identified in the vi scaff file into a new file name 16S_2KB.fna

```
for i in `cat scaff`; do samtools faidx 2000kb.fa $i >>16S_2Kb.fna; done
```

The extracted sequence where then uploaded to Blast and Silva website to identify the bacteria to which those sequence belong.

14.2 GTDBTK

Github

In the folder with all your clean and completed genomes run this command with nohup

```
#!/bin/bash
gtdbtk classify_wf --cpus 20 --genome_dir /home/karine/Bins/all_clean --out_dir gtdbk_output
```

Once it is done running, you can open the folder called `gtdbk_output` and copy the folder `gtdbtk.bac120.summary.tsv` to your local computer in order to open it with excel. Use this folder to identify the phylum, class, order, family and genus that you need to download in order to construct your tree.

14.3 OneCodeX

Chapter 15

PHYLOGENETIC TREE

15.1 Download reference genomes

a. Download assembly summary genbank file

```
wget ftp://ftp.ncbi.nlm.nih.gov/genomes/genbank/assembly_summary_genbank.txt
```

b. For each phylum, order, class, family of interest, search in the NCBI database and select 10 individuals with the most complete genome. Copy and paste the name and the assembly number in an excel document. From that excel document, coopy the list of assembly in a vi document called ref.txt

c. Search for those identifiers in the assembly summary genbank file with the following command ;

```
for n in $`cat ref.txt`; do grep $n assembly_summary_genbank.txt  | cut -f 1,7,8,20 >> genomestodownlo
```

d. Modify the paths to download only the fna genomes

```
cut -f 4 genomestodownload.tab  |  sed 's/$/\/*fna.gz/g' >> ftp.links.txt
```

e. Download the genomes and unzip the downloaded files

```
wget -i ftp.links.txt
```

```
gunzip *.gz
```

f. Add the file name next to the orginial header

```
for i in *.fna ; do perl -lne 'if(/^>(\S+)/){ print ">$1 [$ARGV]"} else{ print }' $i > $i.MF_Ref.fna;
```

g. Keep only the files that end with _genomic.fna -> you can move all the others to another folder

h. Create a new directory and move your files there

```
mkdir /home/karine/MF_ref
```

```
mv *MF_Ref.fna /home/karine/MF_ref
```

h. Run CheckM

```
checkm lineage_wf -x fna /home/karine/MF_ref checkm -f output_table.txt
```

Discard the genomes that are not completed.

15.2 Phylosift

Githup

Midgard

a. Create a folder containing your completed .fna bins and reference genomes

b. Create a vi file named run_phylosift.sh with the following command and run it using `nohup ./` and `&`

```
 #!/bin/bash
find . -maxdepth 1 -name "*.fna" -exec /home/baker/phylosift_v1.0.1/bin/phylosift search --isolate --b
```

c. Once phylosift is done align the marker genes with the following command using `nohup ./` and `&`

```
find . -maxdepth 1 -name "*.fna" -exec /home/baker/phylosift_v1.0.1/bin/phylosift align --isolate --be
```

d. Rename and concatenate the aligned marker for both bins

```
find . -type f -regex '.*alignDir/concat.updated.1.fasta'   -exec cat {} \; | sed -r 's/\.1\..*//'  >
```

15.3 Geneious

a. Download Geneious software (free 14 days trial) select `new foler` drag and drop the aligned fasta file

b. Select `Align/Assemble` ; `Multiple align MUSCLE Alignment` … ; ok

c. Once it is done, remove the bins or sequence that seem too small and re-run `Multiple align - MUSCLE Alignment`

d. Once it is done, select the aligned file go to `Tools` ; `Mask Align`

e. Export the file : `File` ; `Export` ; `Selected Documents` ; `Phylip alignment (*phy)`

f. Move the file from your local computer to the server

15.4 RAxML

Github

In the folder with your .phy file create a vi file named tree.sh with the following command and run it using `chmod +x` ; `nohup ./`and `&`

```
#!/bin/bash

raxmlHPC-PTHREADS-AVX -T 35 -f a -m PROTGAMMAAUTO -N autoMRE -p 12345 -x 12345 -s MF_aligned.phy -n MF
```

When the tree is done running, copy the file `RAxML_bipartitionsBranchLabels.yourfilename` to your local computer‘

15.5 iTOL

a. Create an iTOL account and upload your tree (RAxML_bipartitionsBranchLabels.yourfilename) and visualize the results

b. Change the name of all the reference genomes

1. Locate the folder where all the reference genoms are located and using terminal view the content of that foler using `ls -l`
2. Copy/paste the name into an new tab of the excel document where you have saved the GCA assembly name of the reference genomes.
3. Use VLOOKUP to match the GCA assembly name of the reference genomes to the organism name
4. In another tab, copy the (1) column with the orginal name of the iTOL tree and the (2) column containing the matching Organism name (in this order very important)
5. Copy these two column in a vi file named `ref_name_tree` in the same folder with your tree

6. In the same folder create an executable perl script named `replace_name.sh` to change the name in your tree

```
use strict;
use warnings;

my $treeFile = pop;
my %taxonomy = map { /(\S+)\s+(.+)/; $1 => $2 } <>;

push @ARGV, $treeFile;

while ( my $line = <> ) {
   $line =~ s/\b$_\b/$taxonomy{$_}/g for keys %taxonomy;
   print $line;
}
```

7. Run the perl script

```
perl replace_name.sh ref_name_tree RAxML_bipartitionsBranchLabels.MF_karine > new_tree_name
```

8. Upload the new created tree to iTOL

Chapter 16

METABOLIC PATHWAY

16.1 Convert bins to protein (.faa)

In the folder with your bins in the .fna format…

a. Make sure there is no space between the name of you bins and the scaffhold

```
for i in *.fna ; do sed -i 's/ /_/g' $i ; done
```

b. Run `prodigal` to convert your bins to amino acid sequence

```
for i in *.fna ; do prodigal -i $i -o output.txt -a $i.faa ; done
```

c. Move all the files ending with .faa to another folder `mkdir faa ; mv *.faa /home/karine/Bins/all_clean/fna/faa`

d. Remove all the characters after the first space in the header

```
for i in *.faa; do sed -i 's/\s.*$//' $i; done &
```

e. Tranfer the files to your local computer

f. Submit the files to kofamKOALA and HydDB - you can only submit one file at a time

16.1.1 KofamKoala

a. Submit one file at a time

b. Accept the email

c. Save each file as a text file identified with the name of the bin on your computer.

16.1.2 HydDB

a. Submit one file at a time and do not close the window while it runs.

b. When it's completed, download the excel and with the option text to column seperate the name of the name of bin/contigs and the class prediction.

c. Use custom sort to sort by column B (the Hygrogenase group) copy only the groups that are hydrogenase ([FeFe], [NiFe], [fe]-)

d. Paste these results in an excel document, combining all results from all bins

16.2 Hydrogenase tree

a. In the server (takes a lot of memory) concatenate all your .faa files

```
cat *.faa > concat_faa.faa
```

b. Create a vi document call `hydDB_contigs.txt` and copy the name of the sequence from your the excel document containing the name of the contigs identified as hydrogenase

c. Extract those sequence from the concatenated file (count after using grep -c ">" to make sure the numbers match)

Option 1 - La plus rapide

```
pullseq -i concat_faa.faa -n hydDB_contigs.txt>> hydDB_sequence.faa
```

f. View the lenght of the extracted sequences

```
seqkit fx2tab --length --name --header-line hydDB_sequence.faa >> hydDB_sequence.faa.fasta.lenght
```

g. Remove all the smaller sequences (<140)

```
perl -lne 'if(/^(>.*)/){ $head=$1 } else { $fa{$head} .= $_ } END{ foreach $s (keys(%fa)){ print "$s\n
```

h. Count how many you have

```
grep -c ">" long_sequence.faa
```

i. Download the HydrogenaseDataBase fasta file located on the Google Drive

j. Use diamond to compared your sequences with the onces in the DataBase

```
diamond makedb --in HydroDB.fasta  -d hydroDB
```

k. Modify your .faa files to remove any space and turn them into 1 liner

```
for i in *.faa; do sed -i  's/\s.*$//' $i; done &
```

```
for i in *.faa ; do perl -lne 'if(/^(>.*)/){ $head=$1 } else { $fa{$head} .= $_ } END{ foreach $s (sor
```

l. In the folder with all your .faa bins

```
for i in *.faa; do diamond blastp --db hydroDB.dmd --query $i --out $i.hyd.tab --threads 3 --outfmt 6
```

m. Concat all the `.hyd.tab` files and open the file with excel. Add these as headers for your document

qtitle | stitle | pident | length | Qstart | qend | Sstart | send | evalue | Bitscore |

n. Use VLOOKUP to identify the hydrogenase sequence that were identified throught diamond blast and hydDB

o. Keep only the sequence with an alignment length cutoff > 40 amino acid residues and sequence identity > 50% and Remove Group 4 Hydrogenase hit with sequence identity < 60%

p. Create a vi document call good_sequences.txt and extract (pullseq) those sequence from the file with all your sequence

```
pullseq -i all_sequence.faa -n good_sequence.txt>> hydDB_diamond_filtered_sequences.faa
```

q. Concat those sequences with the original GoogleDrive data base (I need to open the text document and search for `bin`, in order to do a manuel `enter` at the end of the data base sequence and the begining of my sequence, otherwise muscle won't work because there will be `>` in a sequence)

r. Muscle align your sequences in the server (takes a lot of time on the computer)

```
muscle -in long_sequence.faa -out aligned_long_sequence.faa -log log.txt -maxiters 2
```

s. Transfer files to geneious MAFFT and Mask align

t. Eport the file as a Fasta to run fasttree

u. Run fasttree command using `nohup ./` and `&`

```
`#!/bin/bash

fasttree hyd_align_comp.fasta > hyd.tree
```

16.2.0.1 Hydrogrenase Database sequence lenght

To know the lenght of the sequence in the DataBase

```
seqkit fx2tab --length --name --header-line hydroDB.faa >> HydroDB_length.faa
```

Create a histogram

```
cut -f 4 hydroDB_length > DBlengths
```

Open DBlengths with vi to remove the first row and run this command to generate the histogram

```
less DBlengths | Rscript -e 'data=abs(scan(file="stdin")); png("seq.png"); hist(data,xlab="lengths", x
```

16.3 Mebs

github

git clone https://github.com/valdeanda/mebs.git

Download mebs in the folder containing all your .faa files

Mebs runs on python3 -> download python3 on your computer (used `brew install python`)

Once python3 is installed create a folder called Mebs containing all your .faa files

Use `pip install` to download these four libraries

```
apt-get install python3 python3-pip python3-matplotlib \ ipython3-notebook python3-mpltoolkits.basemap

pip3 install -U pip

pip3 install --upgrade pandas $ sudo -H pip3 install --upgrade numpy

pip3 install --upgrade scipy

pip3 install --upgrade seaborn $ sudo -H pip3 install -U scikit-learn
```

Run mebs using perl

```
perl mebs.pl -input /home/karine/tree/faa -type genomic -comp > MF_karine.tsv
```

In the folder with the output file.tsv run the mebs python script

```
python3 mebs_vis.py Phylo_5.tsv
```

16.4 Metabolic

Githib

Metabolic was run from the Student folder, with the following command within the folder with the FAA files

```
perl /home/students2020/Tools/METABOLIC/METABOLiC-G.pl -in-gn /home/students2020/karine/folderwithFAA
```

Chapter 17

Other Stuff

Create MetabatContig_list.tsv list needed for DAStool

17.1 One file

```
awk '/>/{sub(">","&"2000kb.fa"_");sub(/\.fa/,x)}1'
```

```
grep ">" concat.fasta > concat.txt
```

```
sed -i 's/\.//g' concat.txt
sed -i 's/>//g' concat.txt
```

17.2 Bash script

```
`#!/bin/bash

for i in *.faa

do

VIBRANT_run.py -i $i -f nucl -t 10 -l 10000

done
```

Next separate names in concat.txt so have scaffold name followed by sample and bin name example .tsv:

scaffold_1326 AB_69_metabat_bin_100

scaffold_1711 AB_69_metabat_bin_100

scaffold_2201 AB_69_metabat_bin_100

scaffold_2419 AB_69_metabat_bin_10

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr*. Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2020). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.20.