Fiskaaling
Aquaculture Research Station of the Faroes

Methods

Taxonomic classifiers

Creating customised taxonomic classifiers for bioinformatic pipelines based on publicly available gene-reference databases

Ása Johannesen
Ása Jacobsen

8. December 2023

Fiskaaling rit 2023-14

Fiskaaling P/F
við Áir, FO-430 Hvalvík, Føroyar (Faroe Islands)
Tlf. (+298) 474747, fiskaaling@fiskaaling.fo, www.fiskaaling.fo

Fiskaaling
Aquaculture Research Station of the Faroes
Fiskaaling P/F
við Áir
FO-430 Hvalvík
Føroyar (Faroe Islands)

Tlf. (Phone) (+298) 474747
Fax (+298) 474748
E-mail:fiskaaling@fiskaaling.fo
www.fiskaaling.fo

Methods

Heiti: Taxonomic classifiers

Høvundar: Ása Johannesen
Ása Jacobsen

Status: Open

Frágr.nr.: 2023-14

Verkætlan: Pipeline v2

Dato: 8. December 2023

Tal av síðum: 20

Ábyrgdarhavi: Ása Jacobsen

Góðkent: Amanda Vang

Undirritan:

Samandráttur:

This documents the process of downloading and preparing reference sequence data from publicly available databases for constructing classifiers for assigning taxonomy to barcode sequence data. Here code is collated with additional original code for creating classifiers for various taxonomic groups, using different gene regions and based on reference sequences from six different genetic databases. The document is designed to be used as a modifiable general pipeline guide with all the necessary code to get started on constructing your own customised reference database.

Leitiorð:

Bioinformatics, pipeline, classifier, Qiime2, eDNA

Contents

1 Introduction

This report is constructed as a handbook describing the process of making customised taxonomic classifiers based on a selection of gen-reference databases focusing on various taxonomic groups and gene regions. The reference databases are very different, some allowing for analysis of samples on websites and some require downloading and then building a classifier. Below are descriptions of the databases that will be explored in this report and examples are made illustrating different approaches and purposes. All the databases are described to the point where a trained classifier has been constructed, ready for use with your own data.
The aim of this report is to provide a useful guide to building various customized classifiers. Although there are many other available examples of building classifiers, it is a laborious process to acquire enough knowledge to being able to making your own customized classifiers. This is mainly due to the information needed being scattered across various webpages, forums, etc.. In this process, we have been very aware of citing the origin of the code compiled and if any error is made in this respect, it is unintentional.
The platform used here is Qiime2[1] which also includes terminal or Python code. All code chunks are annotated and explanations are provided to allow users to adapt according to their needs. The code is available in Jupyter Notebooks on GitHub (in prep). Further assistance can also be found on the Qiime2 website, which contains various guidelines and tutorials as well as a friendly and helpful forum.

2 Setup

This handbook was constructed using Qiime2 version 2021.2.0 (Bolyen et al. 2019) running with Jupyter version 3.6.15 (Van Rossum and Drake Jr 1995). Packages and plugins used in this handbook are as follows:

- Qiime2
  - RESCRIPt (Robeson et al. 2020)
  - q2-feature-table (Robeson et al. 2020)
  - q2-feature-classifier (Bokulich et al. 2018)
  - q2-clawback (Kaehler et al. 2019)
- Python
  - Pandas (Pandas development team 2020)

Since the code was written in Jupyter Notebooks, the Qiime2 code snippets are preceeded with an '!' in order to run in JupyterNotebook. This would not be necessary if running from the terminal.

Mostly, when the user of this handbook should input their own file names or variables, we have replaced our own with some filler text surrounded by less than and greater than symbols ("<your file name here>"). Without replacing these with a filename or other relevant input, the code will not run. Some times, we have not removed our own input. This may be for illustrative purposes, so that the user understands how the input ought to be formatted. However, there are also some instances where for example a whole pipeline with very specifically named files are used in several subsequent code snippets.

[1] www.qiime2.org

The Python for loops described in the NCBI chapter are a good example of this. That chapter describes how to download data within a for loop with some very specific filters and in the code, the files are named for the taxid that was downloaded and the type of data they are. This file name designation is then used in all the subsequent steps.

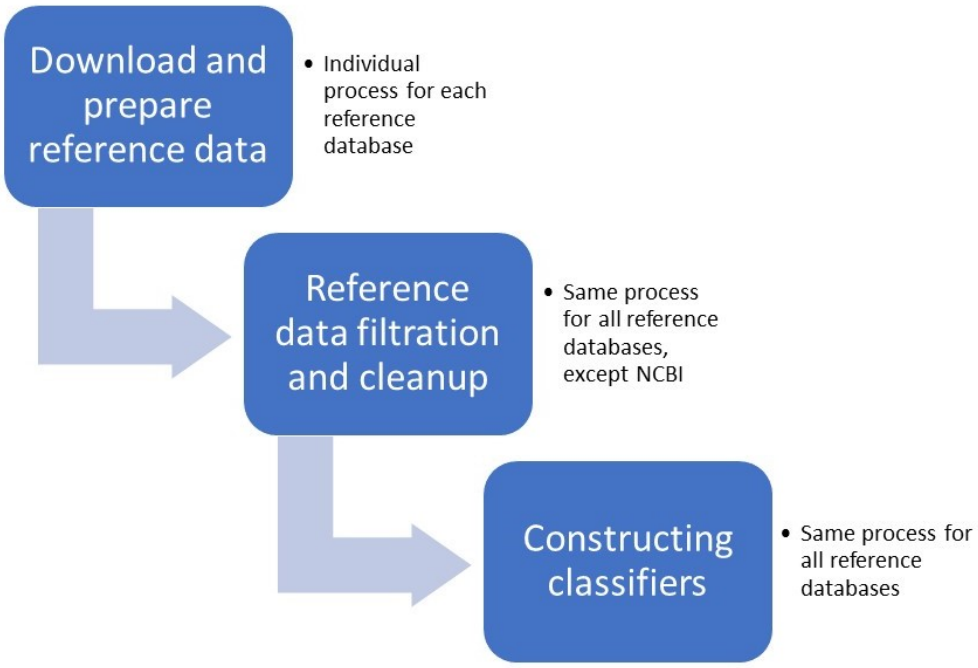

Figure 1: Structure of handbook sections. Most reference databases can be processed using this flow, though data from NCBI is processed slightly differently. In the case of data from NCBI, the reference data filtration and cleanup is performed on several files simultaneously as described in the NCBI chapter, so the chapter "Reference data filtration and cleanup" can most likely be skipped.

3 Silva reference database

The Silva database contains sequence and taxonomy data for small subunit (16S and 18S) and large subunit (23S and 28S) ribosomal RNA for Bacteria, Archaea and Eukaryota (Quast et al. 2013).

It is possible to download pre-trained classifiers or to train your own classifier from the data available on the Silva website[2]. As an example, we will explore the 16S and 18S datasets.

[2]arb-silva.de

3.1 Downloading reference data

This will download the currently most recent Silva release and format it for use in Qiime2.

[ ]:
```
!qiime rescript get-silva-data \
    --p-version '138.1' \
    --p-target 'SSURef_NR99' \
    --o-silva-sequences silva-138.1-ssu-nr99-rna-seqs.qza \
    --o-silva-taxonomy silva-138.1-ssu-nr99-tax.qza
```

The downloaded sequences are RNA, so you can transcribe them to DNA using the following.

[ ]:
```
!qiime rescript reverse-transcribe \
    --i-rna-sequences silva-138.1-ssu-nr99-rna-seqs.qza \
    --o-dna-sequences silva-138.1-ssu-nr99-seqs.qza
```

4 NCBI GenBank reference database

NCBI Genbank is the largest and most comprehensive database including sequences of any kind. It is possible to download data based on taxonomic groups, primers, projects and many other criteria. (Benson et al. 2013)

In this chapter we describe how to use Qiime2 to download reference sequences and taxonomy for your taxa and gene region of choice. We will demonstrate how this is done using the COI region and marine fish taxa. You can find taxid numbers on the NCBI website[3] by searching your taxon and clicking on it in the results list.

Due to the high demand on NBCIs servers, it is advisable to download the data outside of US working hours and in manageable chunks. After downloading, we will demonstrate how to filter the data and to construct a classifier.

4.1 Downloading the data

Choose your working directory

[1]:
```
import os
os.chdir(<your path>)
```

Create a list of the taxids that you want to download. A few taxids are inserted for illustrative purposes.

[2]:
```
taxgroups = ["txid32443[ORGN]","txid7777[ORGN]","txid117565[ORGN]",
"txid117569[ORGN]"]
```

[3] ncbi.nlm.nih.gov/taxonomy

Create a boolean variable (theorsandnots) to ensure that the correct gene region is downloaded in addition to other potential filtration terms. Here we download COI data and exclude unclassified, environmental, unknown, and uncultivated sequences.

[3]:
```
theorsandnots = "AND (cytochrome c oxidase subunit 1[Title] OR␣
↪cytochrome c oxidase subunit I[Title] OR cytochrome oxidase subunit␣
↪1[Title] OR cytochrome oxidase subunit I[Title] OR COX1[Title] OR␣
↪CO1[Title] OR COI[Title]) NOT (unclassified OR environmental OR␣
↪unknown OR uncultivated)"
```

Create a for loop in order to download the individual taxonomic groups that were defined in the list above.

- *therequest* is a variable containing the taxonomic groups and the boolean filtration and will be used in the Qiime2 call
- *groupname* splits the "group" variable so that the first part is the required taxid, which is used for naming output files
- *refseqs* is the output name for reference sequences using the first part of the group name - that is the taxid.
- *taxonomy* is the output name for the taxonomy files defined in the same way as above

In this download request, we specify all available ranks for the taxonomy. This may not be necessary depending on your needs.

[4]:
```
for group in taxgroups:
    therequest = f'{group} {theorsandnots}'
    groupname = group.split("[O")
    refseqs = f'{groupname[0]}-refseqs.qza'
    taxonomy = f'{groupname[0]}-taxonomy.qza'
    !qiime rescript get-ncbi-data \
        --p-query "$therequest" \
        --p-n-jobs <define number of cores to use> \
        --p-ranks kingdom subkingdom superphylum phylum subphylum␣
↪infraphylum superclass class subclass infraclass superorder order␣
↪suborder infraorder superfamily family subfamily genus species \
        --o-sequences $refseqs \
        --o-taxonomy $taxonomy
```

4.2 Filtration and cleanup of reference data

This section assumes you downloaded your NCBI data in several small downloads and that you need to process more than one file of representative sequences and one taxonomy file. If you only have one file of each, you can skip the Python for loops and simplify your Qiime2 calls.
Define the path to your downloads.

[2]:
```
import os
dir = "<path to your working directory>"
```

Create lists of reference sequence files and taxonomy files. The list of reference sequence files is used downstream. You can print the lists if you want to check that the pattern is working as intended. If you have named your files differently, so that the reference sequence files do not contain "refseqs" and the taxonomy files do not contain "taxonomy", you should put another suitable identifying string in "sequence_pattern" and "tax_pattern"

[3]:
```
sequence_pattern = "refseqs"
tax_pattern = "taxonomy"
sequence_files = [f for f in os.listdir(dir) if sequence_pattern in f]
tax_files = [f for f in os.listdir(dir) if tax_pattern in f]

print(sequence_files)
print(tax_files)
```

Iterate through the list of reference sequence files and dereplicate them. Output files are given the "derep" addition so that it is clear that they are dereplicated files. You can define the number of threads to use based on your system. There are several modes to choose from and here we use "uniq". This mode keeps all unique sequences including identical ones if they have different taxonomies.

[5]:
```
for group in sequence_files:
    taxid = group.split("-ref")
    derep_refseqs = f'{dir}/{taxid[0]}-derep-refseq.qza'
    derep_taxonomy = f'{dir}/{taxid[0]}-derep-taxonomy.qza'
    taxname = f'{dir}/{taxid[0]}-taxonomy.qza'
    !qiime rescript dereplicate \
        --i-sequences $dir/$group \
        --i-taxa $taxname \
        --p-mode 'uniq' \
        --p-threads <define number of threads to use> \
        --o-dereplicated-sequences $derep_refseqs \
        --o-dereplicated-taxa $derep_taxonomy
```

Define a list of dereplicated reference sequence files for further filtering.

[6]:
```
derep_pattern = "derep-refseq"
derep_files = [f for f in os.listdir(dir) if derep_pattern in f]
print(derep_files)
```

Extract and filter the sequences based on your primers to reduce the data that you will use to build the classifier. This is time consuming and you can define the number of threads to use based on your system.

[8]:
```
for group in derep_files:
    taxid = group.split("-derep")
    extracted_refseqs = f'{taxid[0]}-extracted-refseq.qza'
    !qiime feature-classifier extract-reads \
        --i-sequences $group \
        --p-f-primer <your forward primer sequence> \
        --p-r-primer <your reverse primer sequence> \
```

```
        --p-n-jobs <define number of threads to use> \
        --o-reads $extracted_refseqs
```

Create a list of files containing the extracted and filtered reference sequences.

[11]:
```
extracted_pattern = "extracted-refseq"
extracted_files = [f for f in os.listdir(dir) if extracted_pattern in␣
 ↪f]
print(extracted_files)
```

Filter sequences based on number of degenerates and homopolymer length.

[10]:
```
for group in extracted_files:
    taxid = group.split("-extract")
    culled_refseqs = f'{taxid[0]}-culled-refseq.qza'
    !qiime rescript cull-seqs \
        --i-sequences $group \
        --p-num-degenerates <number of degenerates> \
        --p-homopolymer-length < homopolymer length> \
        --o-clean-sequences $culled_refseqs
```

Create a list of culled files.

[12]:
```
culled_pattern = "culled-refseq"
culled_files = [f for f in os.listdir(dir) if culled_pattern in f]
print(culled_files)
```

Filter the culled files based on sequence length.

[13]:
```
for group in culled_files:
    taxid = group.split("-culled")
    filtered_refseqs = f'{taxid[0]}-filtered-refseq.qza'
    discarded_refseqs = f'{taxid[0]}-discarded-refseq.qza'
    !qiime rescript filter-seqs-length \
        --i-sequences $group \
        --p-global-min <your global minimum> \
        --o-filtered-seqs $filtered_refseqs \
        --o-discarded-seqs $discarded_refseqs
```

Create list of filtered files formatted as string.

[14]:
```
filtered_pattern = "filtered-refseq"
filtered_files = [f for f in os.listdir(dir) if filtered_pattern in f]
files_to_merge = " ".join(filtered_files)
print(files_to_merge)
```

Merge your cleaned and filtered reference sequences using the "feature-table" "merge-seqs" command. Note that the input data can not be a list, which is why the above code is constructed the way it is.

[15]:
```
!qiime feature-table merge-seqs \
--i-data $files_to_merge \
--o-merged-data <name your merged sequences output .qza file>
```

Create a list of your dereplicated taxonomy files.

[16]:
```
derep_taxa_pattern = "derep-taxonomy"
derep_taxa_files = [f for f in os.listdir(dir) if derep_taxa_pattern␣
 ↪in f]
taxa_files_to_merge = " ".join(derep_taxa_files)
print(taxa_files_to_merge)
```

Merge your taxonomy files.

[17]:
```
!qiime feature-table merge-taxa \
--i-data $taxa_files_to_merge \
--o-merged-data <name your merged taxonomy .qza file>
```

Dereplicate your taxonomy and reference sequences again after merging.

[18]:
```
!qiime rescript dereplicate \
--i-sequences <your merged sequence file> \
--i-taxa <your merged taxonomy file> \
--p-mode 'uniq' \
--p-threads <desired number of threads> \
--o-dereplicated-sequences <your final reference sequence .qza file> \
--o-dereplicated-taxa <name your final taxonomy .qza file>
```

5 MIDORI2 reference database

The MIDORI2 database (Leray, Knowlton, and Machida 2022) consists of eukaryota mitochondrial DNA and can be used either by downloading the database or by uploading samples to their website[4].

In this chapter, we describe how the MIDORI2 reference database is used both for building a customised classifier as well as how to use their online classifier. The website functionality is restricted to up to 10,000 sequences. Upon upload, users are given a choice of three different classifiers (programs) and in this report, results from the RDP classifier are used.

MIDORI2 originally included only Metazoan data, but has since expanded to include all eukaryotes. We have applied MIDORI2 to building a classifier for marine invertebrates by filtering as described below.

[4] reference-midori.info

5.1 Downloading reference data

Begin by downloading the appropriate database files. This can be found here[5]. Here we use the files formatted for QIIME, selecting the newest species level uniq data set. From that list, we download the FASTA files and the taxon file for COI primers. The database contains different files for different target genes.

Before constructing your classifier, it might be a good idea to filter the downloaded data based on the taxa of interest. This can be done using the perl script available for download from the MIDORI2 download page mentioned above. There is also a manual available for download. Bear in mind that not all taxonomic levels are included in the downloadable taxonomy, so when constructing the target list, ensure that you are using taxa that also exist in the taxon file.

Navigate to your working directory.

[2]:
```
import os
os.chdir(<your path>)
```

Import the downloaded FASTA and taxon files into Qiime2 to create the qza files that Qiime2 works with. Once you have successfully created a .qza for both your reference sequences and your taxonomy, you are ready to start working on cleaning up your reference data before building your classifier.

[7]:
```
!qiime tools import \
--type 'FeatureData[Taxonomy]' \
--input-path <your downloaded and perl filtrated .taxon file> \
--input-format HeaderlessTSVTaxonomyFormat \
--output-path <name your .qza taxon file>
```

[8]:
```
!qiime tools import \
--type 'FeatureData[Sequence]' \
--input-path <your downloaded and perl filtrated .fasta file> \
--output-path <name your .qza refseq file>
```

5.2 Using the online classifier

Navigate to their server[6]. From there you can upload your representative sequences up to a maximum of 10,000 sequences. You can select between three different classifiers or programs to use for your analysis. Choose between unique and longest sequences and the gene region. Set the confidence level and the format of your output. The output will be a .zip folder with .txt files, one containing the Feature IDs with the corresponding taxonomy and another with taxid, corresponding taxonomy, and the number of reads.

[5] reference-midori.info/download.php
[6] reference-midori.info/server.php

6 Mare-MAGE reference database

This database is developed specifically for fish and is hosted by the Thunen Institute of Fisheries Ecology at Bremerhaven. It contains data for for the 12s and COI gene regions that can be downloaded in FASTA format as well as taxonomy files from the Mare-MAGE website[7].
In this chapter, we describe how the Mare-MAGE reference database is used.

6.1 Downloading reference data

Begin by choosing the dataset that you want. You can choose from 12S and COI and also level of confidence and whether you want complete gene sequences.
You can download the data from their website mentioned above.
There is a Qiime2 tutorial on the Mare_MAGE website, which describes the classifier building that we also describe here, but beware that some of their fasta files may not be correctly formatted for Qiime2. We found that the COI90 fasta file contained lower case letters and spaces, which are not accepted in the Qiime2 import. If that is the case for you, you can convert the files to upper case using sed or awk. Here is an example of how it might be done using awk:

[ ]:
```
awk 'BEGIN{FS=" "}{if(!/>/){print toupper($0)}else{print $1}}' in.fna
> out.fna
```

Import your downloaded files into Qiime2. Here shown for references sequences (.fasta) first and then for taxonomy.

[ ]:
```
!qiime tools import \
--type FeatureData[Sequence] \
--input-path <your downloaded Mare-MAGE .fasta file> \
--output-path <name your reference sequence .qza file>
```

[ ]:
```
!qiime tools import \
--type FeatureData[Taxonomy] \
--input-path <your downloaded Mare-MAGE taxonomy file> \
--output-path <name your reference taxonomy .qza file file> \
--input-format HeaderlessTSVTaxonomyFormat
```

7 PR2 reference database

The PR2 database is an 18S RNA protist database with additional data from metazoa, fungi, and plants (Guillou et al. 2013). They have developed an interactive pr2-primer database where you can download primer sets and evaluate against the database as well as testing your own primers and primer sets. The database is accessible through a website interface[8] and an R package.

[7] mare-mage.weebly.com
[8] app.pr2database.org/pr2-database

Here we describe how to download reference sequences and taxonomy files from PR2[9].

7.1 Downloading reference data

The PR2 reference database can be found at the database link above. You can filter the data if you are interested in specific taxa, but we downloaded the full database from the "Download full database" tab. This has download links for fasta and tax files that are formatted in a way that is suitable for Qiime2 import.

Navigate to your working directory of choice.

[1]:
```
%cd <your working directory>
```

Import your downloaded taxonomy and reference sequence files.

[2]:
```
!qiime tools import \
--type 'FeatureData[Taxonomy]' \
--input-path <your reference taxonomy file> \
--input-format HeaderlessTSVTaxonomyFormat \
--output-path <name your reference taxonomy .qza file>
```

[3]:
```
!qiime tools import \
--type 'FeatureData[Sequence]' \
--input-path <your reference sequence file> \
--output-path <name your reference sequence .qza file>
```

8 BOLD reference database

The Barcode of Life Data System (BOLD) is a web based platform for integrated assembly and use of DNA barcode data (**ratnasingham_span_2007**). For animal identification it focuses on the mitochondrial gene (COI). In addition it contains ITS, rbcl and matK barcodes for fungi and plant identification. The system provides opportunities for selecting relevant reference sequences based on various criteria such as geography and taxonomy.

In this chapter, we describe how you can download and use the BOLD reference database[10]. We have used it to produce classifiers for fish and macroalgae, but we will keep the guide here as general as possible. We will describe how to extract the taxa of interest from your downloaded data and create files suitable for importing into Qiime2.

[9] pr2-database.org
[10] boldsystems.org

8.1 Downloading reference data

Begin by downloading the BOLD snapshot of your choice. These are created regularly, and can be found here[11]. We recommend that you use one of the historical snapshots, as these can be cited. In this document, we use the March 2023 snapshot.

8.1.1 Extracting desired taxa

Create a list of search terms defining the taxa that you would like to extract from the downloaded data. In this example, we are interested in Macrophyta, so we have chosen Chlorophyta, Phaeophyceae, and Rhodophyta. Ensure that the taxon that you choose exists in the BOLD database, as not all taxonomic levels are included.

We then use pandas to create a dataframe containing the lines where one of the search terms appears.

[ ]:
```
search_terms = ["Chlorophyta", "Phaeophyceae", "Rhodophyta"]
primer_terms = ["COI-5P","COI-3P"]

import pandas as pd

with open('BOLD_Public.31-Mar-2023.tsv', 'r', encoding='utf-8') as␣
↪database:
    header = next(database).strip().split('\t')
    df = pd.DataFrame(columns=header)

    for line in database:
        values = line.strip().split('\t')

        if len(values) != len(header):
            values += [''] * (len(header) - len(values))

        if any(term in values for term in primer_terms):
            if any(term in values for term in search_terms):
                df = df.append(dict(zip(header, values)),␣
↪ignore_index=True)

    df.to_csv("Tari/BOLD_macrophyta.31-Mar-2023_COI.tsv", index=None,␣
↪sep='\t')
```

8.1.2 Creating reference sequence and taxonomy files

The following creates the taxonomy tsv file that Qiime2 imports for use in building a classifier.

Start by defining the path to your files; that is input and output files. The input file should be your taxon filtrated BOLD release file.

[11] boldsystems.org/index.php/datapackages

Define the columns from the BOLD release file that you want to use. For Qiime2 import, you want columns 1 and 2 in addition to all of the columns that contain the taxonomy information - here defined as range(9-16).
The code then opens the input and output files and creates reader and writer objects for the files and defines the header.
Finally, the code iterates through all of the lines in the dataset and extracts the desired columns and writes it to the predefined output file.

[1]:
```
import csv

input_file_path = 'BOLD_macrophyta.31-Mar-2023_COI.tsv'
output_file_path = 'BOLD_macrophyta_COI_taxon.tsv'

columns_to_extract = [0,1]
columns_to_extract.extend(range(9, 16))

with open(input_file_path, 'r', newline='') as input_file,␣
 ↪open(output_file_path, 'w', newline='') as output_file:
    csv_reader = csv.reader(input_file, delimiter='\t')
    csv_writer = csv.writer(output_file, delimiter=';')

    header = next(csv_reader)

    for row in csv_reader:
        extracted_data = [row[i] for i in columns_to_extract]
        csv_writer.writerow(extracted_data)
```

The following code cleans up the output .tsv in the following way:

1. Replace the first ";" in the file with a "-". This combines the first two columns from the original dataset to make a unique identifier.
2. Replace the next ";" with a TAB delimiter. This ensures that the tsv has the correct format, which is an identifier followed by the taxonomy where the taxonomy is separated from the identifier with a TAB and each level in the taxonomy is separated by semi-colons.
3. Finally, any case of "None" is removed from the file, again in order to conform to Qiime2 requirements.

[ ]:
```
!sed 's/;/-/' BOLD_macrophyta_COI_taxon.tsv >␣
 ↪BOLD_macrophyta_COI_taxonomy.tsv
```

[ ]:
```
!sed 's/;/\t/' BOLD_macrophyta_COI_taxonomy.tsv >␣
 ↪BOLD_macrophyta_COI_taxonomy2.tsv
```

[ ]:
```
!sed 's/None//g' BOLD_macrophyta_COI_taxonomy2.tsv >␣
 ↪BOLD_macrophyta_COI_taxonomy3.tsv
```

The following constructs the fasta file for Qiime2 through a series of terminal code.

First, the three columns containing the relevant information are written to separate files. These are the first two columns as in the taxonomy file, which are used to create the identifier and the third column contains the sequences.

[ ]:
```
!awk -F'\t' '{print $1 > "processid1.txt"; print $2 > "sampleid1.txt";␣
↪print $52 > "sequences1.txt"}' BOLD_macrophyta.31-Mar-2023_COI.tsv
```

Several times throughout the process, it is a good idea to look at the data, to ensure that the code does what is expected.

[ ]:
```
with open("sequences1.txt") as input_file:
    head = [next(input_file) for i in range(20)]

    print(head)
```

The two identifier columns are put together with a "-" separator.

[ ]:
```
!paste -d'-' processid1.txt sampleid1.txt > identifiers.txt
```

[ ]:
```
with open("identifiers.txt") as input_file:
    head = [next(input_file) for i in range(10)]

print(head)
```

The sequences and the identifiers occupy alternating lines in a fasta file, so a '\n' is added in the paste command.

[ ]:
```
!paste -d'\n' identifiers2.txt sequences2.txt > refseqs_nu.fasta
```

This tidies up the fasta file in order to approach the correct final formatting.
It removes the first two lines, which were the header lines and then adds a >to every odd line

[ ]:
```
input_file_path = 'refseqs_nu.fasta'
output_file_path = 'refseqs2_nu.fasta'

with open(input_file_path, 'r') as file:
    lines = file.readlines()

lines = lines[2:]

for i in range(len(lines)):
    if i % 2 == 0:
        lines[i] = '>' + lines[i]

with open(output_file_path, 'w') as file:
    file.writelines(lines)
```

This replaces "-" with nothing globally, but only in the lines that contain the sequences. This is necessary as the Qiime2 import function we want to use will not accept dashes.

[ ]:
```
!sed -e "/^[^>]/s/[I-]//g" refseqs2_nu.fasta > reference_sequences_nu.
↪fasta
```

At this point, the reference sequence file ought to be ready for import into Qiime2. However, there may be an issue relating to duplicate records. When building the classifier, Qiime2 will return an error if there are duplicate records. Therefore, if you find you encounter this problem, some decisions need to be made. Sometimes the duplicates are "real" duplicates with the same sequence and identifier. Other times, the sequences do not match, but the identifier is the same. In these cases, one might either delete one of the sequences or add more information to the identifier in order to keep both sequences. This must be done both in the taxonomy file and the fasta file, or - even better - in the release file before the two separate files are made.

[ ]:
```
!qiime tools import \
--type FeatureData[Sequence] \
--input-path reference_sequences_nu.fasta \
--output-path macrophyta_refseqs.qza
```

[ ]:
```
!qiime tools import \
--type FeatureData[Taxonomy] \
--input-path BOLD_macrophyta_COI_taxonomy3.tsv \
--output-path macrophyta_tax.qza \
--input-format HeaderlessTSVTaxonomyFormat
```

9 Reference data filtration and cleanup

Once you have your reference sequence and taxonomy .qza files, you can start the process of cleaning them up, so that you can build your classifiers. Some of the steps here are done to decrease the size of your reference data so that building the classifier will be faster (or in some cases, possible) whereas other steps are done to maximise the quality of the classifier. It is not required to perform all of the steps and the order of the various steps in the process is not pre-determined. We have arranged the different cleanup steps according to computer power requirements so that the less requiring processes are first to reduce the amount of data before the more demanding parts are initiated.

In order to ensure that your .qza files are ok, you can validate them using the following code.

[ ]:
```
!qiime tools validate <your .qza file here>
```

You can then start by culling sequences that don't live up to certain minimal quality standards. A suitable starting place is to limit the number of degenerates to 5 and the homopolymer length to 8. However, you should set these to what you think is most suitable for your data. As going through your reference sequences to find problematic ones is quite resource heavy, it can help to specify that you would like to use more than one core, if you have more available.

[5]:
```
!qiime rescript cull-seqs \
    --i-sequences <your reference sequence .qza file> \
    --p-num-degenerates <number of degenerates> \
    --p-homopolymer-length <maximum homopolymer length> \
    --p-n-jobs <desired number of threads> \
    --o-clean-sequences <name your culled sequences .qza file>
```

You can also remove any sequences that are shorter than some specified minimum. This will depend on the type of sequences that you are using, but it is a good idea to remove short sequences.

[6]:
```
!qiime rescript filter-seqs-length \
--i-sequences <your reference sequences .qza file> \
--p-global-min <minimum sequence length> \
--o-filtered-seqs  <name your filtered sequences .qza file> \
--o-discarded-seqs <name your discarded sequences .qza file>
```

Here is a filter that allows you to set sequence length based on individual taxa. You can also filter taxa out entirely by not mentioning them in "-p-labels". You should specify a minimum sequence length for each taxa, which is why we have 900 (for Archaea) and 1200 (for Bacteria).

[ ]:
```
!qiime rescript filter-seqs-length-by-taxon \
    --i-sequences <your reference sequences .qza file> \
    --i-taxonomy <your taxonomy .qza file> \
    --p-labels Archaea Bacteria \
    --p-min-lens 900 1200 \
    --o-filtered-seqs  <name your filtered sequences .qza file> \
    --o-discarded-seqs <name your discarded sequences .qza file>
```

Ensure that you dereplicate your files. This will substantially decrease the size of your reference data and you can most likely dereplicate several times depending on the order that you decide to carry out these cleanup steps. Removing duplicates will make any other cleanup action that you do faster.

Here you dereplicate the sequences. We have chosen the ‘uniq’ method because this will keep identical sequences with different taxonomies. This is useful when we want to build a weighted classifier, as the weights can help determine where the sequence ought to be classified. Other options include “Last common ancestor” (lca) where the closest common ancestor in the two taxonomies with matching sequences is chosen and “majority” where the taxonomy with a majority of identical sequences is assigned to the sequence.

We have provided the –p-rank-handles line for your convenience. The ranks listed are default, but you can change them if you like.

[7]:
```
!qiime rescript dereplicate \
--i-sequences <your filtered sequences .qza file> \
--i-taxa <your reference taxonomy .qza file> \
--p-mode 'uniq' \
--p-threads <desired number of threads> \
```

```
--p-rank-handles 'domain' 'phylum' 'class' 'order' 'family' 'genus'␣
↪'species' \
--o-dereplicated-sequences <name your dereplicated sequences .qza file>␣
↪\
--o-dereplicated-taxa <name your dereplicated taxonomy .qza file>
```

Filter the sequences based on your primers to reduce the data that you will use to build the classifier. This is time consuming and you can define the number of threads to use based on your system. There is some discussion about whether filtering on primer is a good idea or not. You may lose good data, so consider whether you want to include this step or not. Here[12] is what the Qiime2 documentation says on the matter and here[13] is a really good description of how to avoid some of the pitfalls when doing this kind of filtration written by Mike Robeson.

If you do filter, don't forget to dereplicate again before you build your classifier.

[ ]:
```
!qiime feature-classifier extract-reads \
--i-sequences <your refseq .qza file> \
--p-f-primer <your forward primer sequence> \
--p-r-primer <your reverse primer sequence> \
--p-n-jobs <your desired threads> \
--o-reads <name your primer filtrated refseqs .qza file>
```

10 Constructing the classifier

There are two comonly used methods for constructing a classifier in Qiime2. One uses the "feature-classifier" suit of commands whereas the other is in the "rescript" plugin.

"feature-classifier fit-classifier-naive-bayes" will construct a classifier, but it will not evaluate the quality of the classifier that you have built. On the Qiime2 forums, there are tutorials, which will describe how to evaluate your classifier, but we will not go further into that here.

"rescript evaluate-fit-classifier" will construct a classifier and return an evaluation .qzv. It will also return an observed taxonomy, which you can compare to the downloaded taxonomy. However, this method is very resource heavy. It will particularly use a lot of RAM, so if your reference data are large or your computer is limited in RAM you may fail to complete construction of your classifier.

We recommend you try both methods and decide for yourself, which works best for your purposes.

Below is the less resource heavy "fit-classifier-naive-bayes". This will take some time to complete, perhaps up to half an hour.

[12] https://docs.qiime2.org/2023.9/tutorials/feature-classifier/

[13] https://forum.qiime2.org/t/using-rescripts-extract-seq-segments-to-extract-reference-sequences-without-pcr-primer-pairs/23618

[ ]:
```
!qiime feature-classifier fit-classifier-naive-bayes \
--i-reference-reads <your reference sequences> \
--i-reference-taxonomy <your reference taxonomy> \
--o-classifier <name your classifier>
```

Below is the more resource heavy "evaluate-fit-classifier"
It is possible to run part of this process in parallel, but most of it will run on a single thread. This function is quite demanding, particularly in terms of RAM, so it is important to ensure that your input data for the classifier does not contain too many unnecessary sequences or taxa. Expect to run this over night, though it depends on the hardware available to you and how much you have been able to trim your input data.

[ ]:
```
!qiime rescript evaluate-fit-classifier \
--i-sequences <your filtered refseq .qza file> \
--i-taxonomy <your filtered taxonomy .qza file> \
--p-n-jobs <your desired threads> \
--o-classifier <name your classifier .qza file> \
--o-evaluation <name your classifier evaluation .qzv file> \
--o-observed-taxonomy <name your observed taxonomy .qza file>
```

10.1 Constructing a weighted classifier

There is a Qiime2 plugin that allows you to create weights based on the type of sample that you have in order to possibly improve the accuracy of your classifier. This is "q2-clawback" and at the moment it is particularly suitable for bacterial communities or microbiomes. We base the following on the tutorial written by the creators of clawback and posted on the Qiime2 forums[14]. Using this plugin will allow you to weigh the classifier on, for example whether the sample is for an animal gut or from a water sample. The plugin collects data from Qiita (Gonzalez et al. 2018) and bases the weights on this metadata.

First we query Qiita to get the most recent data.

[ ]:
```
!qiime clawback summarize-Qiita-metadata-category-and-contexts \
  --p-category empo_3 \
  --o-visualization available_empo3.qzv
```

Clawback works with feature-classifier produced classifiers, so we will construct a Silva classifier to use as an example for how the process works.

[ ]:
```
!qiime feature-classifier fit-classifier-naive-bayes \
  --i-reference-reads silva-138.1-ssu-nr99-seqs-derep-uniq.qza \
  --i-reference-taxonomy silva-138.1-ssu-nr99-tax-derep-uniq.qza \
  --o-classifier silva-unweighted-138.1-ssu-nr99-classifier.qza
```

Once your unweighted classifier has been constructed, you can create the weights.

[14] forum.qiime2.org/t/using-q2-clawback-to-assemble-taxonomic-weights/5859

Here, you need to input an unweighted classifier, a reference taxonomy and reference sequences. Again, we are using our Silva data as an example.
We specify that we want the metadata from Empo 3 and that we want the classifier weighed towards animal skin. As our animals are fish, we are not sure how suitable this weighing scheme is, and your mileage may vary. Also, bear in mind that type of sample contains varying amounts of data, and some data sets may be too small to produce useful weights.
Finally, we pick a context. We want one high on the list, to ensure we have a lot of data to base the weights on, and we want long sequences if we can have them. At the moment, the "Deblur_2021.09-Illumina-16S-V4-150nt-ac8c0" context is the best one to use as has is the longest (150nt) 16S sequence variants (deblur) within the list produced by Qiita. This will most likely change in future.

[ ]:
```
!qiime clawback assemble-weights-from-Qiita \
  --i-classifier silva-unweighted-138.1-ssu-nr99-classifier.qza \
  --i-reference-taxonomy silva-138.1-ssu-nr99-tax-derep-uniq.qza \
  --i-reference-sequences silva-138.1-ssu-nr99-seqs-derep-uniq.qza \
  --p-metadata-key empo_3 \
  --p-metadata-value "Animal surface" \
  --p-context Deblur_2021.09-Illumina-16S-V4-150nt-ac8c0b \
  --o-class-weight silva-animal-surface-weights.qza
```

Once the weights have been created, you can build a new classifier using “feature-classifier” and this time with the added input of a weights .qza file.

[ ]:
```
qiime feature-classifier fit-classifier-naive-bayes \
  --i-reference-reads silva-138.1-ssu-nr99-seqs-derep-uniq.qza \
  --i-reference-taxonomy silva-138.1-ssu-nr99-tax-derep-uniq.qza \
  --i-class-weight silva-animal-surface-weights.qza \
  --o-classifier silva-animal-surface-138.1-ssu-nr99-classifier.qza
```

Now we have two classifiers, one weighted and one unweighted. When classifying your data using a weighted classifier, do ensure that you are classifying data from a suitable substrate. This may mean that you need to filter your data based on sample metadata before or during your process.

References

Benson, Dennis A., Mark Cavanaugh, Karen Clark, Ilene Karsch-Mizrachi, David J. Lipman, James Ostell, and Eric W. Sayers. 2013. “GenBank.” [in eng]. Place: England, *Nucleic acids research* 41, no. Database issue (January): D36–42. ISSN: 1362-4962 0305-1048. https://doi.org/10.1093/nar/gks1195.

Bokulich, Nicholas A, Benjamin D Kaehler, Jai Ram Rideout, Matthew Dillon, Evan Bolyen, Rob Knight, Gavin A Huttley, and J Gregory Caporaso. 2018. “Optimizing taxonomic classification of marker-gene amplicon sequences with QIIME 2’s q2-feature-classifier plugin.” Publisher: BioMed Central, *Microbiome* 6 (1): 90.

Bolyen, Evan, Jai Ram Rideout, Matthew R. Dillon, Nicholas A. Bokulich, Christian C. Abnet, Gabriel A. Al-Ghalith, Harriet Alexander, et al. 2019. "Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2." *Nature Biotechnology* 37 (8): 852–857. ISSN: 1546-1696. https://doi.org/10.1038/s41587-019-0209-9.

Gonzalez, Antonio, Jose A. Navas-Molina, Tomasz Kosciolek, Daniel McDonald, Yoshiki Vázquez-Baeza, Gail Ackermann, Jeff DeReus, et al. 2018. "Qiita: rapid, web-enabled microbiome meta-analysis" [in en]. *Nature Methods* 15, no. 10 (October): 796–798. ISSN: 1548-7091, 1548-7105. https://doi.org/10.1038/s41592-018-0141-9.

Guillou, Laure, Dipankar Bachar, Stéphane Audic, David Bass, Cédric Berney, Lucie Bittner, Christophe Boutte, et al. 2013. "The Protist Ribosomal Reference database (PR2): a catalog of unicellular eukaryote small sub-unit rRNA sequences with curated taxonomy." [in eng]. Place: England, *Nucleic acids research* 41, no. Database issue (January): D597–604. ISSN: 1362-4962 0305-1048. https://doi.org/10.1093/nar/gks1160.

Kaehler, Benjamin D, Nicholas Bokulich, Daniel McDonald, Rob Knight, J Gregory Caporaso, and Gavin A Huttley. 2019. "Species abundance information improves sequence taxonomy classification accuracy." *Nature Communications* 10 (4643).

Leray, Matthieu, Nancy Knowlton, and Ryuji J. Machida. 2022. "MIDORI2: A collection of quality controlled, preformatted, and regularly updated reference databases for taxonomic assignment of eukaryotic mitochondrial sequences." _Eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/edn3.303, *Environmental DNA* 4 (4): 894–907. https://doi.org/https://doi.org/10.1002/edn3.303.

Pandas development team, The. 2020. *pandas-dev/pandas: Pandas,* February. https://doi.org/10.5281/zenodo.3509134.

Quast, Christian, Elmar Pruesse, Pelin Yilmaz, Jan Gerken, Timmy Schweer, Pablo Yarza, Jorg Peplies, and Frank Oliver Glockner. 2013. "The SILVA ribosomal RNA gene database project: improved data processing and web-based tools." Publisher: Oxford University Press, *Nucleic Acids Res* 41 (Database issue): D590–6.

Robeson, Michael S, Devon R O'Rourke, Benjamin D Kaehler, Michal Ziemski, Matthew R Dillon, Jeffrey T Foster, and Nicholas A Bokulich. 2020. "RESCRIPt: Reproducible sequence taxonomy reference database management for the masses." Publisher: Cold Spring Harbor Laboratory _eprint: https://www.biorxiv.org/content/early/2020/10/05/2020.10.05.326504.full.pdf, *bioRxiv,* https://doi.org/10.1101/2020.10.05.326504.

Van Rossum, Guido, and Fred L Drake Jr. 1995. *Python reference manual.* Centrum voor Wiskunde en Informatica Amsterdam.