Apr 23, 2024

Using high throughput amplicon sequencing to determine the diet of two generalist stink bugs (Hemiptera) in agricultural and urban landscapes.

DOI
dx.doi.org/10.17504/protocols.io.e6nvwdewdlmk/v1

Olivier Berteloot[1]

[1]Ghent University

Olivier Berteloot
ghent university

DOI: dx.doi.org/10.17504/protocols.io.e6nvwdewdlmk/v1

Protocol Citation: Olivier Berteloot 2024. Using high throughput amplicon sequencing to determine the diet of two generalist stink bugs (Hemiptera) in agricultural and urban landscapes.. **protocols.io** **https://dx.doi.org/10.17504/protocols.io.e6nvwdewdlmk/v1**

Protocol status: Working
We use this protocol and it's working

Created: October 02, 2023

Last Modified: April 23, 2024

Protocol Integer ID: 88662

Keywords: amplicon; NGS; metabarcoding

Funders Acknowledgement:
VLAIO (Flanders Innovation and Entrepreneurship)
Grant ID: LA-traject HBC.2018.2224

Abstract

The use of DNA metabarcoding has become an increasingly popular technique to infer feeding interactions in herbivores and generalist predators and are especially useful when the organism of interest is polyphagous. Inferring host plant preference of native and invasive herbivore insects can be helpful in establishing effective Integrated Pest Management strategies (IPM). Both stink bugs, *Halyomorpha halys*, and *Pentatoma rufipes*, are known pests that cause severe economic damage in agroecosystems, primarily commercial fruit orchards. In this study, we performed Molecular Gut Content Analysis (MGCA) of these two polyphagous herbivore stink bug species using next-generation amplicon sequencing (NGAS) of the Internal Transcribed Spacer 2 (ITS2) barcode region. Additionally, a laboratory experiment with a host switch from a mixed diet to a monotypic diet with *H. halys* was conducted to determine the detectability of the original host plants in a time series up to 3 days after the host switch occurred. In our field samples, we detected 55 unique plant genera across the two stink bug species. The sampling location significantly impacts the observed genera in the diet of both stink bug species, while we observe no significant seasonal differences. Moreover, this study provides additional support for the efficiency of DNA metabarcoding techniques to infer the dietary composition of polyphagous herbivores, delivering species-level resolution of hostplant-herbivore interactions. Lastly, our study provide an initial framework for more extensive DNA metabarcoding studies further to unravel the polyphagous diet of these two pentatomid herbivores.

Guidelines

NA

Materials

see protocol

Safety warnings

NA

Ethics statement

NA

Before start

NA

Sample collection

1 From 2019-2022, *H. halys* and *P. rufipes* were collected in commercial organic orchards (apple, pear and mixed crops), private orchards and urban gardens. 25 locations were sampled throughout the northern parts of Belgium. *P. rufipes* specimens were hand-caught by scanning trees, shrubs and wildflowerstrips in and around plots in the sampling area fo 00:30:00 , spending 00:01:00 per tree, shrub or flower plot, host plants were recorded. *H. halys* were collected using live and sticky traps baited with pheromones (Pherocon Trécé BMSB) and checked twice per week. 31m

2 Collected specimens were placed into clean 1.5 mL tubes (Eppendorf) and immediately frozen on dry ice to be stored in the lab at -80 °C

Dissection

3 Prior to DNA extraction, leave specimen for 00:00:10 in 1 % bleach solution. After, the alimentary canal was dissected from crop till rectum, flash frozen in liquid nitrogen in a 1.5 mL L tube with 2 stainless steel beads (5mm diameter) and lysed using a TissueLyser II (Qiagen Inc. Valencia CA USA). 10s

DNA extraction

4 Total DNA was extracted from the dissected guts using the DNeasy Blood and Tissue Kit (Qiagen), following manufacturers' instructions.

PCR and Illumina workflow

5 **Library preparation**

First amplification

performed using a customer specific primer pair, which contained an additional Illumina TruSeq adaptor sequence (see below) on 1-10 ng of DNA extract (total volume 1µl).

- ITS-S2F 5’ GACGTGTGCTCTTCCGATCT
- ITS-u4 5’ ACACGACGCTCTTCCGATCTR

15 pmol of each forward and reverse primer was used in 20 µL volume of 1x MyTaq buffer containing 1.5 units MyTaq DNA polymerase (Bioline GmbH, Luckenwalde, Germany) and 2 µL of BioStabII PCR Enhancer (Sigma-Aldrich Co.).

Step	Duration (sec)	Temperature (°C)	Cycles
Pre-Denaturation	60	96	1
Denaturation	15	96	30
Annealing	30	58	
Extention	90	70	
Final hold	ad ininitum	8	1

Table 1. PCR cycle settings

Second amplification
1 µL of each amplicon obtained in the first PCR was used, and these amplicons were separately amplified in a 20 µL reaction volume using standard i7- and i5- sequencing adaptors.

The second amplification was as first amplification but with a modified annealing temperature, i.e. 3 cycles at 50 °C followed by 7 cycles at 58 °C

6 **Pooling and clean up**

DNA concentration of amplicons was assessed by agarose gel electrophoresis. 20 ng of indexed amplicon DNA of each sample was subsequently pooled (up to 96 samples per pool).

The pooled libraries were purified with one volume of Agencourt AMPure XP beads (Beckman Coulter, Inc., IN, USA) to remove primer dimer and other small mispriming products, followed by an additional purification on MiniElute columns (QIAGEN GmbH, Hilden, Germany).

7 **Size selection and sequencing**

The size selection was performed by preparative gel electrophoresis on a LMP-Agarose gel. Sequencing was done on an Illumina MiSeq (Illumina, Inc., CA, USA) using V3 Chemistry (2x300bp).

Sequence analysis

8 **Create environment**

Hardware environment specs:
- Processor: AMD Ryzen 5 5600X, 3,7 GHz (4,6 GHz Turbo Boost) - 6-Cores - 12 Threads
- RAM: Corsair Vengance LPX 128GB (4 x 32GB) DDR4 DRAM 3200MHz C16 Memory Kit
- Windows 11 Pro, WSL2, Ubuntu Linux 20.04

1. Unlock processor in BIOS: press del on start up > advanced settings > virtualization > enable
2. Win key > type: CMD > wsl --install
3. Download Ubuntu in windows app store & install
4. Double click Ubuntu to start Ubuntu

5. Install Miniconda

Command

Install Miniconda (Linux)

```
mkdir -p ~/miniconda3
wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-
x86_64.sh -O ~/miniconda3/miniconda.sh
bash ~/miniconda3/miniconda.sh -b -u -p ~/miniconda3
rm -rf ~/miniconda3/miniconda.sh
```

6. Initialize Miniconda

Command

Initialize Miniconda (Linux)

```
~/miniconda3/bin/conda init bash
~/miniconda3/bin/conda init zsh
```

7. Update minidconda

Command

Update Miniconda (Linux)

```
conda update conda
```

8. Install wget

Command

Install wget (Linux)

```
conda install wget
```

9. Install Qiime2 (install the shotgun too, it has the rescript pre-installed)

Command

Install QIIME2 (Linux)

```
wget https://data.qiime2.org/distro/amplicon/qiime2-amplicon-2023.9-
py38-linux-conda.yml
conda env create -n qiime2-amplicon-2023.9 --file qiime2-amplicon-
2023.9-py38-linux-conda.yml

wget https://data.qiime2.org/distro/shotgun/qiime2-shotgun-2023.9-py38-
linux-conda.yml
conda env create -n qiime2-shotgun-2023.9 --file qiime2-shotgun-2023.9-
py38-linux-conda.yml
```

10. Activate Qiime2

Command

Activate QIIME2 (Linux)

```
conda activate qiime2-amplicon-2023.9

or

conda activate qiime2-shotgun-2023.9
```

9 **Data state**

Data is not multiplexed
Paired-end 2x300bp
Quality scores are Phred 33V

10 Unzip all .gz files in directory

Command

For loop (Linux)

```
for i in *.gz; do zcat "$i" > "${i%.*}"; done
```

Data are in following path (example):

/home//RAW/1_R1.fastq
/home//RAW/1_R2.fastq

absolute filepaths are needed for the metadata file, and importing in QIIME2

11 **Metadatafile**
Create a metadatafile with following information for all samples

- absolute forward filepath

- absolute reverse filepath
- label
- location
- date
- season
- coordinates
- organism
- life stage
- type (orchard or garden)
- landscape (urban or rural)
- hostplant or trap

Create the metadatafile in Google Sheets

File>download> Tab separated values .tsv

12 **Import data into QIIME2**

Command

Import data (QIIME2)

```
qiime tools import \
  --type 'SampleData[PairedEndSequencesWithQuality]' \
  --input-path metadata.tsv \
  --output-path paired-end-demuxed.qza \
  --input-format PairedEndFastqManifestPhred33V2
```

Expected result

paired-end-demuxed.qza

A QIIME 2 artifact contains data and metadata. The metadata describes things about the data, such as its type, format, and how it was generated (provenance). A QIIME 2 artifact typically has the .qza extension. Using artifacts instead of data files enables researchers to focus on the analyses they want to perform, instead of the particular format the data needs to be in for an analysis

13 **Trim primers & cut adapters**

Command

trim primers & cut adapters (QIIME2)

```
qiime cutadapt trim-paired \
  --p-cores 6 \
  --i-demultiplexed-sequences paired-end-demuxed.qza \
  --p-front-f GACGTGTGCTCTTCCGATCTATGCGATACTTGGTGTGAAT \
  --p-adapter-r ACACGACGCTCTTCCGATCTRGTTTCTTTTCCTCCGCTTA\
  --o-trimmed-sequences trimmed-seqs.qza \
  --verbose
```

Expected result

trimmed-seqs.qza

14 **Quality filtering, denoising, dereplication and clustering (DADA2)**

Looking at the quality fastqc files for each sample.
All forwards look pretty similar to each other, and all reverses look pretty similar to each other, but worse than the forwards, which is common.

Phred scores of 20 vs 40 means 1 error per 100 bases vs. 1 error per 10000 bases.

The forward scores drop off at 250 bp, we want to maintain a median phred score of 30 (1 error in 1000). Reverse scores drop off around 185 bp. Knowing our ITS2 fragment is around 350-400 base base pairs and we need some overlap for the reverse and forward reads to be merged, in DADA2 this minimum overlap is 12, more is welcome. We will leave a length of 250p forward and 185bp reverse.

Command

DADA2 (QIIME2)

```
qiime dada2 denoise-paired \
  --i-demultiplexed-seqs trimmed-seqs.qza \
--p-chimera-method pooled \
--p-pooling-method pseudo \
  --p-trunc-len-f 250 \
--p-trunc-len-r 185\
--p-trunc-q 0 \
--p-n-threads 10 \
  --o-representative-sequences rep-seqs.qza \
  --o-table table-dada2.qza \
  --o-denoising-stats stats-dada2.qza \
--verbose
```

Expected result

- rep-seqs.qza (all sequences present, dereplicated)
- table-dada2.qza (frequency table)
- stats-dada2.qza (per sample input sequences and output sequences, filtered, percentage passed, denoised, merged, percentage of input merged)

15 **Local ITS2 BLAST database development**

We want to identify the ASV's (rep-seqs.qza), their frequency, and occurence is linked to the frequency table through an ID. But first we must make a local BLASTn ITS2 database of plants

15.1 **Retrieve available plant ITS2 sequences via NCBI Entrez text query:**

Command

ITS2 plant sequences NCBI text query (Linux)

```
((viridiplantae[Organism] AND its2) AND 100:10000000[Sequence Length])
NOT (uncultured OR environmental sample OR incertae sedis OR unverified)
```

The resulting records were then downloaded via the “Send to” menu :

send to > Complete record > File > Format: Fasta

15.2 **Rename file and check whether all records were retrieved**

Command

Rename (Linux)

```
mv sequence.fasta NCBI_Viridiplantae_ITS2_fasta_file

grep -c ">" NCBI_Viridiplantae_ITS2_fasta_file
```

Expected result

#238585 ==> ok

15.3 **Remove linebreaks**

(NCBI nucleotide sequences downloaded in FASTA format display linebreaks every 70 bases, this may hinder further sequence processing steps)

Command

Remove linebreaks every 70 bases (Linux)

```
awk '!/^>/ { printf "%s", $0; n = "\n" } /^>/ { print n $0; n = "" }END
{ printf "%s", n }' NCBI_Viridiplantae_ITS2_fasta_file >
NCBI_Viridiplantae_ITS2_fasta_file_tmp

mv NCBI_Viridiplantae_ITS2_fasta_file_tmp
NCBI_Viridiplantae_ITS2_fasta_file
```

15.4 **Collect taxonomic identifiers (taxids) of the organisms from ITS2 sequences were obtained**

1. create table linking every accession number to the corresponding nucleotide sequence

Command

Create table (Linux)

```
grep ">" NCBI_Viridiplantae_ITS2_fasta_file | cut -d ">" -f 2 | cut -d " " -
f 1 > AccessionNumbers

paste \
  <(cat AccessionNumbers) \
  <(sed '/^>/d' NCBI_Viridiplantae_ITS2_fasta_file) >
AccessionNumbers_seqs_linking_table
```

2. nucl_gb.accession2taxid NCBI file from the FTP website (large file)

Command

download nucl_gb.accession2taxid from NCBI (Linux)

```
wget
ftp://ftp.ncbi.nih.gov/pub/taxonomy/accession2taxid/nucl_gb.accession2taxid.
gz

gzip -d nucl_gb.accession2taxid.gz
```

3. Retrieve the lines with accession numbers and write into temporary file

Command

Retrieve accession numbers (Linux)

```
fgrep -w -f AccessionNumbers nucl_gb.accession2taxid >
AccessionNumbers_taxids_linking_table
```

quick check if we have retrieved all of them

Command

Accession number check (Linux)

```
wc -l AccessionNumbers_taxids_linking_table
```

Expected result

#238585 ==> corresponds with our ITS2 sequence number

4. Create a table linking accession numbers and taxids

Command

Create final linking table (Linux)

```
awk 'BEGIN {FS=OFS="\t"} {print $2,$3}'
AccessionNumbers_taxids_linking_table >
AccessionNumbers_taxids_linking_table_final
```

15.5 **Retrieve a list of unique taxids**

Command

unique taxids (Linux)

```
awk -F '\t' '{print $2}' AccessionNumbers_taxids_linking_table_final |
sort | uniq > Taxids_uniq

wc -l Taxids_uniq
```

Expected result

#83493 unique taxids

15.6 Collect taxonomic lineages for the taxids

1. The “new_taxdump.tar.gz” NCBI reference file must be downloaded from the NCBI FTP website

Command

NCBI reference file from FTP (Linux)

```
mkdir taxdump

wget
https://ftp.ncbi.nlm.nih.gov/pub/taxonomy/new_taxdump/new_taxdump.tar.gz
mv new_taxdump.tar.gz taxdump/

tar -xvzf taxdump/new_taxdump.tar.gz -C taxdump
```

2. Reformat the rankedlineage.dmp file with simpler field separator (pipe)

Command

Reformat rankedlineage.dmp file (Linux)

```
sed -i "s/\t//g" taxdump/rankedlineage.dmp
```

3. Sort the taxids in rankedlineage.dmp file

Command

Sort taxids (Linux)

```
sort -t "|" -k 1b,1 taxdump/rankedlineage.dmp >
taxdump/rankedlineage_sorted
```

4. Associate the taxids with their corresponding taxonomic lineages

Command

Join taxids with ranked lineage (Linux)

```
join -t "|" -1 1 -2 1 -a 1 Taxids_uniq taxdump/rankedlineage_sorted >
Taxids_taxonomic_lineages_linking_table

wc -l Taxids_taxonomic_lineages_linking_table
```

Expected result

#83493 (taxids linked to their lineages)

5. check for empty lines in the second column in the linking table

Command

check for empty lines (Linux)

```
awk -F '|' '{print $2}' Taxids_taxonomic_lineages_linking_table | grep -
c '^$'
```

Expected result

#0 (no empty lines)

15.7 **Create a table gathering accession numbers, taxids, taxonomic lineages and ITS2 sequences**

1. Link accession number to its corresponding taxonomic lineage, by joining both tables and generate a re-ordered 3 column .tsv file

Command

Join both tables (taxid accession numbers & lineages) (Linux)

```
join -t $'\t' -1 2 -2 1 -a 1 \
    <(sort -t $'\t' -n -k 2
AccessionNumbers_taxids_linking_table_final) \
    <(sort -t $'\t' -n -k 1
Taxids_taxonomic_lineages_linking_table_final) | \
    awk 'BEGIN {FS=OFS="\t"} {print $2, $1, $3}' >
AccessionNumbers_taxids_Taxonomic_lineages_linking_table
```

Command

Add nucleotide sequences according accession numbers (Linux)

```
join -t $'\t' -1 1 -2 1 -a 1 \
      <(sort -t $'\t' -k 1b,1
AccessionNumbers_taxids_Taxonomic_lineages_linking_table)\
      <(sort -t $'\t' -k 1b,1 AccessionNumbers_seqs_linking_table) >
Global_table
```

Command

Check if column in table is complete (Linux)

```
awk -F '\t' '{print $1}' Global_table | grep -c "^$"
awk -F '\t' '{print $2}' Global_table | grep -c "^$"
awk -F '\t' '{print $3}' Global_table | grep -c "^$"
awk -F '\t' '{print $4}' Global_table | grep -c "^$"
```

Expected result

All commands should come back with #0

15.8 **Create a QIIME2-formatted FASTA file and taxonomic lineage files and import into QIIME2**

1. FASTA file creation

Command

Create FASTA (Linux)

```
awk -F '\t' 'BEGIN {OFS=""} {print ">",$1,"\n",$4}' Global_table | sed
's/-//g' > Fasta_file
```

2. Taxonomic lineages

Command

Taxonomic lineages file (Linux)

```
awk 'BEGIN {FS=OFS="\t"} {print $1,$3}' Global_table >
Taxonomic_lineages
```

import sequences into QIIME2 (activate shotgun distribution of Qiime, this includes Rescript).

Command

import into QIIME2 (QIIME2)

```
conda activate qiime2-shotgun-2023.9

qiime tools import \
  --type 'FeatureData[Sequence]' \
  --input-path Fasta_file \
  --output-path Fasta_file.qza
```

15.9 **Cull sequences**

Sequences displaying 5 or more degenerated bases or containing a homopolymer sequence of 12 or more nucleotides should be removed. Maximize thread usage by setting p-n-jobs to 12 (processor has 12 threads)

Command

Cull sequences (QIIME2)

```
qiime rescript cull-seqs \
    --i-sequences Fasta_file.qza \
    --p-homopolymer-length 12 \
    --p-n-jobs 12 \
    --o-clean-sequences Fasta_file_tmp.qza

mv Fasta_file_tmp.qza Fasta_file.qza

qiime tools export \
  --input-path Fasta_file.qza \
  --output-path .
```

15.10 Remove entries that were culled from taxonomy file

Command

Remove culled from taxonomy file (Linux)

```
fgrep -v -f \
  <(cat \
    <(grep ">" dna-sequences.fasta | cut -d ">" -f 2) \
    <(cut -d $'\t' -f 1 Taxonomic_lineages) | sort | uniq -u) \
  Taxonomic_lineages > Taxonomic_lineages_tmp

mv Taxonomic_lineages_tmp Taxonomic_lineages
```

import cleaned taxonomy file into QIIME2 again

Command

import taxonomy file (QIIME2)

```
qiime tools import \
  --type 'FeatureData[Taxonomy]' \
  --input-format HeaderlessTSVTaxonomyFormat \
  --input-path Taxonomic_lineages \
  --output-path Taxonomic_lineages.qza
```

15.11 **Dereplicate sequences in the fasta file and taxonomic lineage file**
RESCRIPt can be used to remove redundant sequence data (optional step)

Command

Dereplicate FASTA & Taxonomy (QIIME2)

```
qiime rescript dereplicate \
    --i-sequences Fasta_file.qza \
    --i-taxa Taxonomic_lineages.qza \
    --p-mode 'uniq' \
    --p-threads 12 \
    --o-dereplicated-sequences Fasta_file_tmp.qza \
    --o-dereplicated-taxa Taxonomic_lineages_tmp.qza

mv Fasta_file_tmp.qza Fasta_file.qza
mv Taxonomic_lineages_tmp.qza Taxonomic_lineages.qza
```

15.12 **Filter out suspected fungal sequences**

1. Sequence and taxonomy data must again be extracted from the .qza files

Command

Export fasta and taxonomic lineage files (QIIME2)

```
qiime tools export \
  --input-path Fasta_file.qza \
  --output-path . && mv dna-sequences.fasta Exported_fasta_file.fasta

qiime tools export \
  --input-path Taxonomic_lineages.qza \
  --output-path . && awk 'NR>1' taxonomy.tsv >
Exported_taxonomic_lineages.tsv
```

We will blast these plant ITS2 sequences against fungi databases in order to identify sequences to have a suspected fungal orgin.

2. Download ITS siquences manually from the **UNITE**website
3. Shorten the description line of the FASTA file to match the BLAST command line requirements

Command

FASTA description line (Linux)

```
paste \
  <(grep ">" sh_general_release_dynamic_10.05.2021.fasta | cut -d "|" -
f 2) \
  <(sed '/^>/d' sh_general_release_dynamic_10.05.2021.fasta) >
AccessionNumbers_seqs_linking_table

awk -F '\t' 'BEGIN {OFS=""} {print ">",$1,"\n",$2}'
AccessionNumbers_seqs_linking_table > UNITE_fungi_seqs.fasta
```

4. create a local BLAST database from the UNITE ITS sequences

Command

UNITE ITS BLAST database

```
makeblastdb \
  -in UNITE_fungi_seqs.fasta \
  -parse_seqids \
  -blastdb_version 5 \
  -title "UNITE_fungi_seqs" \
  -dbtype nucl
```

5. BLAST plant ITS2 sequences against UNITE fungi database

Command

BLAST ITS2 UNITE

```
blastn \
  -db UNITE_fungi_seqs.fasta \
  -query Exported_fasta_file.fasta \
  -num_threads 12 \
  -max_target_seqs 1 \
  -outfmt "6 qacc sacc evalue bitscore length pident ssciname scomname
staxid" \
  -out blastn_outfile_UNITE_fungi
```

using 12 threads again as this is our maximum

6. Reformat the BLAST output file and add length data for each plant reference sequence

Command

Reformat BLAST output file (Linux)

```
sort -buk 1,1 blastn_outfile_UNITE_fungi | sed "s/100.000/100/g" >
blastn_outfile_UNITE_fungi_uniq

awk -F '\t' '{print $1}' blastn_outfile_UNITE_fungi_uniq >
AccessionNumbers_in_blastn_outfile

paste \
  <(cat AccessionNumbers_in_blastn_outfile) \
  <(fgrep -w -f AccessionNumbers_in_blastn_outfile Global_table | awk -
F '\t' '{print length($4)*0.95}' | cut -d ',' -f 1) >
AccessionNumbers_seqs_length_linking_table

join -t $'\t' -1 1 -2 1 -a 1 \
      <(sort -t $'\t' -k 1b,1 blastn_outfile_UNITE_fungi_uniq)\
      <(sort -t $'\t' -k 1b,1
AccessionNumbers_seqs_length_linking_table) >
blastn_outfile_UNITE_fungi_uniq_withlengthdata
```

7. Remove sequences showing at least 90% identity with UNITE ITS sequences on at least 95% of their length

Command

Gather fungal hits (Linux)

```
awk -F '\t' '$6>=90' blastn_outfile_UNITE_fungi_uniq_withlengthdata |
awk -F '\t' '$5>=$NF' | awk -F '\t' '{print $1}' >
Sequences_to_remove_UNITE_seqs
```

8. Filter suspected fungal sequences from our plant ITS2 reference sequences

Command

```
grep -n -A 1 -f Sequences_to_remove Exported_fasta_file.fasta | \
sed -n 's/^\([0-9]\{1,\}\).*/\1d/p' | \
sed -f - Exported_fasta_file.fasta > Fasta_file_without_fungi

grep -v -f Sequences_to_remove_UNITE_seqs
Exported_taxonomic_lineages.tsv > Taxonomic_lineages_without_fungi
```

15.13 **Filter out suspected misidentified sequences**

To identify sequences with a wrong identification, our plant ITS2 reference sequences are analyzed in a cross-validation scheme with data leakage, thus were sets of test and training sequences are strictly identical. This allows comparing expected and predicted taxonomies for each sequence and discarding those for which the expected taxonomy at the family rank is observed only once in the top 5 hits resulting from the BLASTn analysis.

1. Create a table linking accession numbers to taxids from the filtered FASTA file

Command

linking table (Linux)

```
grep ">" Fasta_file_without_fungi | cut -d ">" -f 2 > AccessionNumbers

fgrep -f AccessionNumbers Global_table | awk 'BEGIN {FS=OFS="\t"}
{print $1,$2}' > AccessionNumbers_taxids_linking_table
```

2. Generate a BLAST database

Command

BLAST database (Linux)

```
makeblastdb \
  -in Fasta_file_without_fungi \
  -parse_seqids \
  -blastdb_version 5 \
  -taxid_map AccessionNumbers_taxids_linking_table \
  -title "Fasta_file_without_fungi" \
  -dbtype nucl
```

3. BLAST the plant ITS2 sequences against themselves

Command

BLAST plant ITS2 against themselves (Linux)

```
blastn \
  -db ./Fasta_file_without_fungi \
  -query Fasta_file_without_fungi \
  -num_threads 12 \
  -max_target_seqs 5 \
  -outfmt "6 qacc sacc evalue bitscore length pident ssciname scomname
staxid" \
  -out blastn_outfile_leakedCV
```

15.14 **Process leaked cross-validation results to compare expected to predicted taxonomies**

1. Retrieve top 5 hits accession numbers and taxids

Command

Retrieve top 5 hits (Linux)

```
awk -F '\t' '{print $1}' blastn_outfile_leakedCV | sort | uniq >
AccessionNumbers_in_blastn_outfile

awk 'BEGIN {FS=OFS="\t"} {print $1,$9}' blastn_outfile_leakedCV >
AccessionNumbers_PredictedTaxids_linking_table
```

2. Use these files to keep only the top 5 hits for each reference sequence

Command

Keep top 5 hits in reference sequences (Linux)

```
wk 'seen[$1]++{ $1="" }1' OFS='\t'
AccessionNumbers_PredictedTaxids_linking_table \
  | fgrep -w -A 4 -f AccessionNumbers_in_blastn_outfile \
  | sed '/--/d' > AccessionNumbers_PredictedTaxids_linking_table_top5

paste \
  <(awk -F '\t' '{print $1}'
AccessionNumbers_PredictedTaxids_linking_table_top5 | awk 'BEGIN
{FS=OFS="\t"} NF {p = $0} {print p}') \
  <(awk -F '\t' 'BEGIN {FS=OFS="\t"} {print $2}'
AccessionNumbers_PredictedTaxids_linking_table_top5) >
AccessionNumbers_PredictedTaxids_linking_table_top5_tmp && mv
AccessionNumbers_PredictedTaxids_linking_table_top5_tmp
AccessionNumbers_PredictedTaxids_linking_table_top5
```

3. Number the lines for further data processing

Command

Add line numbers (Linux)

```
awk 'BEGIN {FS=OFS="\t"} {print NR,$0}'
AccessionNumbers_PredictedTaxids_linking_table_top5 >
AccessionNumbers_PredictedTaxids_linking_table_top5_tmp && mv
AccessionNumbers_PredictedTaxids_linking_table_top5_tmp
AccessionNumbers_PredictedTaxids_linking_table_top5
```

4. Generate new linking tables that display the taxonomy info at family rank

Command

New linking tables (Linux)

```
paste \
  <(awk 'BEGIN {FS=OFS="\t"} {print $1}' Global_table) \
  <(awk 'BEGIN {FS=OFS="\t"} {print $3}' Global_table | awk -F '; '
'{print $5}') > AccessionNumbers_taxonomic_lineages_linking_table

paste \
  <(awk 'BEGIN {FS=OFS="\t"} {print $2}' Global_table) \
  <(awk 'BEGIN {FS=OFS="\t"} {print $3}' Global_table | awk -F '; '
'{print $5}') | sort -buk 1,1 > Taxids_taxonomic_lineages_linking_table
```

5. Reformat the table so we can write expected taxonomies to it

Command

Add predicted taxonomies (Linux)

```
sort -n -k 2 AccessionNumbers_PredictedTaxids_linking_table_top5 | awk
'BEGIN {FS=OFS="\t"} {print $3,$4}' >
AccessionNumbers_PredictedTaxids_linking_table_top5_tmp && mv
AccessionNumbers_PredictedTaxids_linking_table_top5_tmp
AccessionNumbers_PredictedTaxids_linking_table_top5

awk 'BEGIN {FS=OFS="\t"} $1 != prev { printf "%s%s", ors, $1; prev=$1;
ors=ORS } { printf " %s", $2 } END { print "" }'
AccessionNumbers_PredictedTaxids_linking_table_top5 | sed "s/ /\t/g" >
AccessionNumbers_PredictedTaxids_linking_table_top5_tmp && mv
AccessionNumbers_PredictedTaxids_linking_table_top5_tmp
AccessionNumbers_PredictedTaxids_linking_table_top5
```

6. Add expected taxonomies to the table

Command

Add expected taxonomy (Linux)

```
join -t $'\t' -1 1 -2 1 -a 1 \
  <(sort -t $'\t' -k 1
AccessionNumbers_PredictedTaxids_linking_table_top5) \
  <(sort -t $'\t' -k 1
AccessionNumbers_taxonomic_lineages_linking_table) -o
1.1,2.2,1.2,1.3,1.4,1.5,1.6 >
AccessionNumbers_PredictedTaxids_linking_table_top5_tmp && mv
AccessionNumbers_PredictedTaxids_linking_table_top5_tmp
AccessionNumbers_PredictedTaxids_linking_table_top5
```

15.15 **Count the number of times the expected family is observed in the top 5 hits**

Command

Count expected family in top 5 (Linux)

```
awk 'BEGIN {FS=OFS="\t"} { i=$1; $1=""; print i, gsub($2,"")-1 }'
AccessionNumbers_PredictedTaxids_linking_table_top5 >
Predicted_taxonomy_count
```

15.16 **Remove sequences for which the expected family is observed only once in the taxonomy of the top 5 hits**

Command

```
awk 'BEGIN {FS=OFS="\t"} $2==1 {print $1}' Predicted_taxonomy_count >
Sequences_to_remove

grep -n -A 1 -f Sequences_to_remove Fasta_file_without_fungi | \
sed -n 's/^\([0-9]\{1,\}\).*/\1d/p' | \
sed -f - Fasta_file_without_fungi >
NCBI_ITS2_Viridiplantae_fasta_file.fasta

grep -v -f Sequences_to_remove Taxonomic_lineages_without_fungi >
NCBI_ITS2_Viridiplantae_taxonomic_lineages.tsv
```

15.17 **Import data into QIIME2**

Command

Import into QIIME2

```
qiime tools import \
  --type 'FeatureData[Sequence]' \
  --input-path NCBI_ITS2_Viridiplantae_fasta_file.fasta \
  --output-path NCBI_ITS2_Viridiplantae_fasta_file.qza

qiime tools import \
  --type 'FeatureData[Taxonomy]' \
  --input-format HeaderlessTSVTaxonomyFormat \
  --input-path NCBI_ITS2_Viridiplantae_taxonomic_lineages.tsv \
  --output-path NCBI_ITS2_Viridiplantae_taxonomic_lineages.qza
```

16 **Train the classifier using the taxonomic lineages and the fasta file**

Command

train classifier

```
qiime feature-classifier fit-classifier-naive-bayes \
  --i-reference-reads NCBI_ITS2_Viridiplantae_fasta_file.qza \
  --i-reference-taxonomy NCBI_ITS2_Viridiplantae_taxonomic_lineages.qza
\
  --o-classifier ITS2_classifier.qza\
--verbose
```

17 **Cluster ASVs into OTUs with 99% similarity**

Command

CLUSTER WITH QIIME

```
qiime vsearch cluster-features-de-novo \
  --i-table table.qza \
  --i-sequences rep-seqs.qza \
  --o-clustered-table otu_table.qza \
  --o-clustered-sequences otu_file.qza \
  --p-perc-identity 0.97
```

18 **Assign taxonomy with BLAST, VSEARCH and the Naive Bayes Classifier**

Command

BLAST in QIIME2

```
qiime feature-classifier classify-consensus-blast \
--i-query otu_file.qza \
--i-reference-reads NCBI_ITS2_Viridiplantae_fasta_file.qza \
--i-reference-taxonomy NCBI_ITS2_Viridiplantae_taxonomic_lineages.qza \
--o-search-results blast_results.qza \
--o-classification classified_blast_results.qza \
--verbose
```

Command

VSEARCH in QIIME

```
qiime feature-classifier classify-consensus-vsearch \
--i-query otu_file.qza \
--i-reference-reads NCBI_ITS2_Viridiplantae_fasta_file.qza \
--i-reference-taxonomy NCBI_ITS2_Viridiplantae_taxonomic_lineages.qza \
--o-search-results vsearch_results.qza \
--o-classification classified_vsearch_results.qza \
--verbose
```

Command

Naive Bayes Classifier in QIIME

```
qiime feature-classifier classify-sklearn \
  --i-classifier ITS2_classifier.qza \
  --i-reads otu_file.qza \
  --o-classification taxonomy_classifier.qza
```

19 Export files from QIIME2 for import in R

OTU counts to a .tsv file

Command

ASV counts export (QIIME2)

```
biom convert -i feature-table.biom -o feature-otu_table.tsv --to-tsv
```

OTU sequences to .fasta file

Command

OTUS to FASTA (QIIME2)

```
 qiime tools export \
  --input-path otu_file.qza \
  --output-path otu_file.fasta
```

Taxonomy file to .tsv file

Command

Taxonomy files to .tsv (QIIME2)

```
qiime tools export \
 --input-path classified_blast_results.qza \
 --output-path taxonomy_blast.tsv

qiime tools export \
 --input-path classified_vsearch_results.qza \
 --output-path taxonomy_vsearch.tsv

qiime tools export \
 --input-path taxonomy_classifier.qza \
 --output-path taxonomy_classifier.tsv
```

metadata is already in .tsv (see step 11)

20 R script to create data file, calculate metrics & visualize data

R_script.R 26KB