Sep 30, 2024

Species Molecular Barcode Analysis with Nanopore Sequence Data

DOI
dx.doi.org/10.17504/protocols.io.6qpvr8y1blmk/v1

Gideon Erkenswick[1], Mrinalini Watsa[2], Zane Libke[1], Pamela Sánchez-Vendizú[1,3], Stefan Prost[4]

[1]Field Projects International; [2]San Diego Zoo Wildlife Alliance;
[3]Programa de Doctorado en Ciencias mención Ecología y Evolución, Escuela Graduados, Facultad de Ciencias, Universidad Austral de Chile, Valdivia, Chile;
[4]University of Oulu

In Situ Laboratories
Tech. support email: **info@insitulabs.org**

Gideon Erkenswick
Field Projects International

DOI: dx.doi.org/10.17504/protocols.io.6qpvr8y1blmk/v1

Protocol Citation: Gideon Erkenswick, Mrinalini Watsa, Zane Libke, Pamela Sánchez-Vendizú, Stefan Prost 2024. Species Molecular Barcode Analysis with Nanopore Sequence Data . **protocols.io https://dx.doi.org/10.17504/protocols.io.6qpvr8y1blmk/v1**

Manuscript citation:
Decoding the Peruvian Amazon with in situ DNA barcoding of vertebrate and plant taxa. Scientific Data (in review).

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

Created: September 08, 2024

Last Modified: September 30, 2024

Protocol Integer ID: 107134

Keywords: nanopore, species identification, molecular barcode, gene markers, coi, cytB, amplicon sequencing, dna barcoding, ngspeciesid, field genomics, in situ

Funders Acknowledgement:
Gordon and Betty Moore Foundation
Grant ID: 9772
Gordon and Betty Moore Foundation
Grant ID: 9776
United States Forest Service
Grant ID: 21-DG-11132762-302

Disclaimer

This is a step-wise bioinformatic workflow to analyze nanopore amplicon sequence data. It includes third party software that was not created nor is maintained by the authors. The end-to-end workflow cannot be replicated without carrying out each of the steps sequentially, and will require tweaks to the input parameters. The final section of the workflow, 'data summary' was created for use with specific spreadsheets used for this study, and will not function in a generic sense on other datasets. This section should be viewed as one example of many potential ways to summarize data.

Abstract

End-to-end workflow to generate cleaned consensus sequences from multiplexed amplicons sequenced with Nanopore sequencing technology. Includes read basecalling, quality control, data filtering, read demultiplexing, consensus sequence generation, and a data summary. The workflow assumes that individual samples have been uniquely indexed to allow highly multiplexed sequencing runs. The demultiplexing code is annotated to indicate different demultiplexing strategies utilized in the study. This workflow was used in the publication "Decoding the Peruvian Amazon with in situ DNA barcoding of vertebrate and plant taxa", and the last section of the workflow consists of a custom R script that generates summary statistics specific to the publication. Overall, this workflow can be modified to suit multiple markers, and is applicable to teaching in a classroom setting with the accompanying docker image.

Image Attribution

Hannah Kim

Guidelines

This is not an automated bioinformatics workflow. To replicate the analyses with new data, carefully review and then carry out each step sequentially. Basic Linux command-line literacy is a prerequisite for using this pipeline. All steps are performed inside freely available Docker containers, eliminating most software compatibility challenges. Take careful note of which Docker image is being used for each step. Minor adjustments to input parameters and custom code sections is often necessary, especially pertaining to the section on "read demultiplexing".

- Dorado basecaller will run most efficiently with a GPU (review software documentation)
- Consider parallelizing NGSpeciesID consensus sequence formation and NCBI blast to decrease computational time

Before start

Software dependencies:

Please ensure the following are installed on your device.

DOCKER (**https://www.docker.com/**)
DORADO (**https://github.com/nanoporetech/dorado/**)

Prepare Working Environment

1 Create a working directory

2 Run the Docker application from the command line or inside the Docker GUI and retrieve the required Docker images. The "pull" command will download images from the online **Docker Hub** repository to the local environment.

Command

docker pull (OSx)

```
docker pull nanozoo/pycoqc
docker pull genomicpariscentre/guppy-gpu
docker pull insitulab/junglegenomics
```

3 Create sub-folders within the working directory

Command

mkdir (linux)

```
cd [path/to/working/directory]

mkdir 0.raw.data
chmod 775 0.raw.data

mkdir 1.all.pod5s
chmod 775 1.all.pod5s

mkdir 2.sam.output
chmod 775 2.sam.output/

mkdir 3.fastq
chmod 775 3.fastq

mkdir 4.filt.fastq
chmod 775 4.filt.fastq

mkdir 5.demux
chmod 775 5.demux
```

4 Establish variables that will be used to simplify command line operations

Command

create variables (linux)

```
HOMEFOL=/path/to/working/directory
PODS=$HOMEFOL/1.all.pod5s
OUTFILE=example_outfile
DORAD=/path/to/dorado/executable
```

Manually move all pod5 files into **path/to/working/director/1.all.pod5s**. This is also easily achieved with common command-line tools.

Command

find all pod5s and move to 1.all.pod5s (linux)

```
find 0.raw.data/ -mindepth 1 -type f -iname '*.pod5' -print|xargs cp -
t 1.all.pod5s/
```

Basecalling Raw Sequence Data

5

Note

At the time of publication there is was no Docker image for Dorado software, this must be installed separately on the local operating system.

This command will perform basecalling on all sequence files inside
path/to/working/directory/1.all.pod5s

Must select a basecalling model:
- Fast (fast)
- High Accurancy (hac) ***RECOMMENDED FOR AMPLICON SEQUENCE DATA***
- Super high accuracy (sup) **** TOO SLOW FOR MOST SERVERS**

Command

dorado basecaller (linux)

```
${DORAD} basecaller hac ${PODS} --emit-moves --no-trim >
${HOMEFOL}/2.sam.output/${OUTFILE}.sam
```

6 Create a summary file of basecalled data

Command

dorado summary (linux)

```
${DORAD} summary ${HOMEFOL}/2.sam.output/${OUTFILE}.sam >
${HOMEFOL}/2.sam.output/sequencingsummary.${OUTFILE}.txt
```

QC & Filtering

7 Run Docker image for pycoQC and create summary plots for the basecalled reads

Command

pycoQC (linux)

```
# run docker
docker run -it -v ${HOMEFOL}:/data nanozoo/pycoqc

OUTFILE=example_outfile


# run pycoQC
pycoQC --file /data/2.sam.output/sequencingsummary.${OUTFILE}.txt --
outfile /data/${OUTFILE}_pycoQC.html
```

Exit Docker before next step

8 Convert basecalled reads in the ".sam" file to a ".fastq" file

Command

samtools fastq (linux)

```
docker run -it -v ${HOMEFOL}:/data insitulab/junglegenomics

OUTFILE=example_outfile

samtools fastq -t /data/2.sam.output/${OUTFILE}.sam >
/data/3.fastq/${OUTFILE}.fastq
```

Remain in same Docker container for next step

9 Filter reads by quality and size criteria. Modify size constraints and quality based on the pycoQC results above and the region you want to target.

Command

NanoFilt (linux)

```
# FILTER CRITERIA:
# filter reads below 325 and above 525 bases
#filter to reads with q score >= 8

NanoFilt -q 8 -l 325 --maxlength 525 --logfile ${OUTFILE}.filt.log
/data/3.fastq/${OUTFILE}.fastq >
/data/4.filt.fastq/${OUTFILE}.325to525bp.q8.fastq
```

Remain in same Docker container for next step.

9.1 OPTIONAL:

NanoFilt is being deprecated soon, so consider moving to chopper instead.

Command

install chopper (linux)

```
#download the installation file.
wget
https://github.com/wdecoster/chopper/releases/download/v0.9.0/chopper-
linux.zip

#make it executable
chmod +x chopper
```

To run chopper, use this command.

Note: Please edit the Q score and minimum and maximum lengths to filter to. Also edit the output file name appropriately.

Command

filter with chopper

```
/home/rocky/bioinfotools/chopper -q 10 --minlength 200 --maxlength
600 -i $OUTFILE.fastq > $OUTFILE.100to800bp.q7.fastq
```

10 Calculate percentage recovery

Command

calculate percent recovery (linux)

```
#counting total reads in un filtered file
echo $(cat /data/3.fastq/${OUTFILE}.fastq|wc -l)/4|bc

# For example 33933 reads

echo $(cat /data/4.filt.fastq/${OUTFILE}.325to525bp.q8.fastq|wc -
l)/4|bc
# For example 28708 reads

# %recovery for FAST files would then be = 28708*100/33933 = 84.6%
```

Exit the Docker container before the next step

Demultiplex Sequence Data

11 Demultiplexing is carried out in 1 or 2 parts, depending on the experimental design:

- Part 1: separating reads based on an outer ONT index kit (usually added by a ligation reaction)
- Part 2: separating reads based on an inner index (usually added by PCR).

If the design involved a dual-indexing strategy, go to step #12 and then carry out Part 2. If not using a dual-indexing strategy, choose whichever part is relevant, either Part 1 or Part 2.

12 Part 1: Dumultiplexing outer ONT indices

Command

guppy barcoder (linux)

```
# enter guppy software docker for demultiplexing
docker run -it -v ${HOMEFOL}:/data genomicpariscentre/guppy-cpu

#command structure
#guppy_barcoder -i <input folder> -s <output folder> --fastq_out --
enable_trim_barcodes --trim_adapters --config configuration.cfg

#our command for just NBD114.24
#assuming you are inside folder data
guppy_barcoder -i /data/4.filt.fastq/ -s /data/5.demux/ --fastq_out --
config configuration.cfg --barcode_kits SQK-NBD114-24
```

Remain in same Docker container for next step

12.1 Check demultiplexing success by counting the number of reads in the unclassified folder

Note

If the operation fails, it may be because the docker container does not have the "bc" command. While inside the docker container, correct this by running the following lines of code

sudo apt-get update
sudo apt-get -y install bc

once complete repeat the commands below

Command

bc (linux)

```
#update permission of unclassified folder
chmod 777 /data/5.demux/unclassified/
cd /data/5.demux/unclassified/
cat *.fastq > all.fastq

apt-get update
apt-get install bc

echo $(cat all.fastq |wc -l)/4|bc

='unclassifed reads'
```

Manually perform the following calculation with the data you obtain on the command line:

Percent recovery = 1 - ('unclassified_reads' / 'total number of reads')*100 = XX%

Remain in same Docker container for next step

13 Part 2: Demultiplex the inner indices

Command

guppy barcoder loop (linux)

```
#find the file:
cd /opt/ont/guppy/data/barcoding
#make a copy
cp barcodes_masked.fasta original_barcodes_masked.fasta

#install nano in the docker image
bash -c 'apt-get -y update && apt -y install nano'

nano barcodes_masked.fasta

# edit the barcode masking file.
{
>BC_1st
GNNNNNNNNNNNNNNNNNNNNNNNNNTTTCTGTTGGTGCTGATATTGC # if starts with "N"
process fails
>BC_2nd
GNNNNNNNNNNNNNNNNNNNNNNNNNACTTGCCTGTCGCTCTATCTT # if starts with "N"
process fails
}

# check if the change was made
cat barcodes_masked.fasta | head -n 20

#command structure (REFERENCE ONLY)
#guppy_barcoder -i <input folder> -s <output folder> --fastq_out --
trim_barcodes --trim_adapters --config configuration.cfg
#our command for just barcoding by PCR only (REFERENCE ONLY)
#guppy_barcoder -i 4.filt_fastq/ -s 5.demux/ --fastq_out --config
configuration.cfg --barcode_kits EXP-PBC096


cd /data/5.demux/

parent_directory="/data/5.demux/"

# Loop through each subfolder in the parent directory
for subfolder in "$parent_directory"/*/; do
    # Get the base name of the subfolder
    subfolder_name=$(basename "$subfolder")
    subfolder_inners="${subfolder%/}/${subfolder_name}_inners"
    # Enter the subfolder
    echo "Processing subfolder: $subfolder"
    cd "$subfolder" || continue

    # Perform your operation here
    mkdir "$subfolder_inners"
    guppy_barcoder -i ./ -s "$subfolder_inners" --fastq_out --
enable_trim_barcodes \
    --config configuration.cfg -t 48 --barcode_kits EXP-PBC096

    # Return to the parent directory
    cd "$parent_directory"
done
```

Remain in same Docker container for next step

13.1 Count outer index demultiplex success

Command

bc (linux)

```
parent_directory="/data/5.demux"
output_file="outer_read_counts.tsv"

# Initialize the output file with headers
echo -e "subfolder\tread count" > "$output_file"

# Loop through each subfolder in the parent directory
for subfolder in "$parent_directory"/*/; do
 # Get the base name of the subfolder
 subfolder_name=$(basename "$subfolder")

 # Initialize the read count for the subfolder
 read_count=0

 # Loop through each .fastq file in the subfolder and count the reads
 for fastq_file in "$subfolder"/*.fastq; do
 if [ -e "$fastq_file" ]; then
 count=$(grep -c "^@" "$fastq_file")
 read_count=$((read_count + count))
 fi
 done

 # Append the subfolder name and read count to the output file
 echo -e "${subfolder_name}\t${read_count}" >> "$output_file"
done
```

Remain in same Docker container for next step

13.2 Count inner index demultiplex success

Command

bc (linux)

```
parent_directory="/data/5.demux"
output_file="inner_read_counts.tsv"

# Initialize the output file with headers
echo -e "relative_path\tread_count" > "$output_file"

# Loop through each subfolder in the parent directory
for subfolder in "$parent_directory"/*/; do
    subfolder_inners="${subfolder%/}/$(basename "$subfolder")_inners"

    # Check if the subfolder_inners directory exists
    if [ -d "$subfolder_inners" ]; then
        # Loop through each folder in subfolder_inners
        for inner_subfolder in "$subfolder_inners"/*/; do
            # Initialize the read count for the inner subfolder
            read_count=0

            # Loop through each .fastq file in the inner subfolder
and count the reads
            for fastq_file in "$inner_subfolder"/*.fastq; do
                if [ -e "$fastq_file" ]; then
                    count=$(grep -c "^@" "$fastq_file")
                    read_count=$((read_count + count))
                fi
            done

            # Get the relative path of the inner subfolder
            relative_path="${inner_subfolder#$parent_directory/}"

            # Append the relative path and read count to the output
file
            echo -e "${relative_path}\t${read_count}" >>
"$output_file"
        done
    fi
done
```

Exit Docker container for next step

Generate Consensus Sequence

14 Determine fragment size within each outer index group using NanoPlot

Note

If the NanoPlot command (below) fails, exit the Docker image, install NanoPlot to the local machine and run it outside of Docker. NanoPlot normally works fine within Docker, but some users have noted glitches. In our experience, it has been more efficient to run NanoPlot another way, than to fix the glitch within docker.

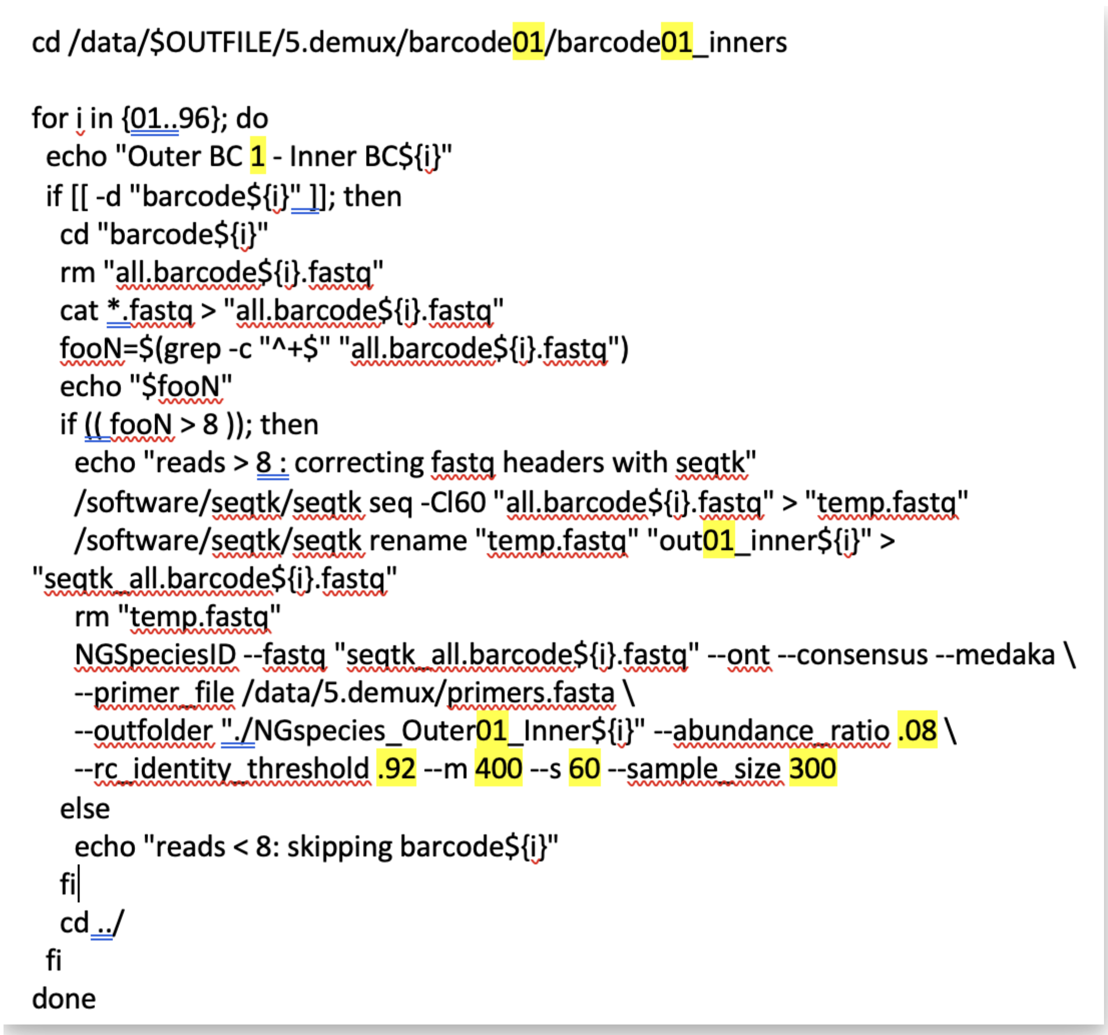
Command

Nanoplot (loop) (linux)

```
# from inside the docker, make sure that you have exited the
NGSpeciesID environment

docker run -it -v ${HOMEFOL}:/data insitulab/junglegenomics

cd /data/5.demux/

# make a list of all the outer barcodes
ls | grep barcode[0-9] -> barcode_outer_list.txt
nano barcode_outer_list.txt # add "unclassified" then save to same
file



FILENAME="./barcode_outer_list.txt"
LINES=$(cat $FILENAME)

for LINE in $LINES
do
  echo "$LINE"
  cd $LINE
  rm all_$LINE.fastq
  foo=$(ls .)
  echo "$foo"
  cat *.fastq > all_$LINE.fastq
  fooN=$(grep -c "^+$" all_$LINE.fastq)
  echo "$fooN"
  if (($fooN > 20)); then # set minimum number of reads for consensus
analysis to take place
    echo "reads > 20 : creating output directory and making histogram"
    mkdir nanoplot_$LINE
    NanoPlot --fastq all_$LINE.fastq -o ./nanoplot_$LINE
    cd ../
  else
    echo "reads < 20: skipping $LINE"
    cd ../
  fi
done
```

Remain in the same Docker image for next step

15 After determining the target fragment size from plots produced in the prior step, run the following code block separately for each outer index. For each iteration of this code block, you will want to adjust all the sections highlighted in yellow in the the example image below. Don't worry, the code itself is in a command listed at the end of this step.



```
cd /data/$OUTFILE/5.demux/barcode01/barcode01_inners

for i in {01..96}; do
  echo "Outer BC 1 - Inner BC${i}"
  if [[ -d "barcode${i}" ]]; then
    cd "barcode${i}"
    rm "all.barcode${i}.fastq"
    cat *.fastq > "all.barcode${i}.fastq"
    fooN=$(grep -c "^+$" "all.barcode${i}.fastq")
    echo "$fooN"
    if (( fooN > 8 )); then
      echo "reads > 8 : correcting fastq headers with seqtk"
      /software/seqtk/seqtk seq -Cl60 "all.barcode${i}.fastq" > "temp.fastq"
      /software/seqtk/seqtk rename "temp.fastq" "out01_inner${i}" >
"seqtk_all.barcode${i}.fastq"
      rm "temp.fastq"
      NGSpeciesID --fastq "seqtk_all.barcode${i}.fastq" --ont --consensus --medaka \
      --primer_file /data/5.demux/primers.fasta \
      --outfolder "./NGspecies_Outer01_Inner${i}" --abundance_ratio .08 \
      --rc_identity_threshold .92 --m 400 --s 60 --sample_size 300
    else
      echo "reads < 8: skipping barcode${i}"
    fi
    cd ../
  fi
done
```

For each iteration of this code block, update the highlighted areas to refer to the index of interest and the desired parameters for NGSpeciesID

Note

These are useful commands to know of if the NGSpeciesID consensus formation step must be repeated several times.

```
## to remove all "ngspecies*" folders, from an appropriate parent directory
find . -type d -name 'NGspecies*' -exec rm -rf {} +

## to remove all seqtk* files, from an appropriate parent director
find . -type f -name 'seqtk*' -exec rm -rf {} +

## to troubleshoot the NGSpeciesID command on a single sample (example)
NGSpeciesID --fastq "seqtk_all.barcode01.fastq" --ont --consensus --medaka \
--outfolder "./NGspecies650_Outer01_Inner01" --abundance_ratio .06 \
--rc_identity_threshold .92 --m 650 --s 50 --primer_file
/data/5.demux_outer_guppy/rugezi_primers.fasta
```

Note

Warning and errors result if the '--primer-file' refers to an improperly formatted file or a location where the file doesn't exist. One option is to remove this parameter and remove primer sites later with another software such as 'cutadapt'.

Command

NGSpeciesID (linux)

```
docker run -it -v ${HOMEFOL}:/data insitulab/junglegenomics

cd /data/$OUTFILE/5.demux/barcode01/barcode01_inners

for i in {01..96}; do
  echo "Outer BC 1 - Inner BC${i}"
  if [[ -d "barcode${i}" ]]; then
    cd "barcode${i}"
    rm "all.barcode${i}.fastq"
    cat *.fastq > "all.barcode${i}.fastq"
    fooN=$(grep -c "^+$" "all.barcode${i}.fastq")
    echo "$fooN"
    if (( fooN > 8 )); then
      echo "reads > 8 : correcting fastq headers with seqtk"
      /software/seqtk/seqtk seq -Cl60 "all.barcode${i}.fastq" >
"temp.fastq"
      /software/seqtk/seqtk rename "temp.fastq" "out01_inner${i}" >
"seqtk_all.barcode${i}.fastq"
      rm "temp.fastq"
      NGSpeciesID --fastq "seqtk_all.barcode${i}.fastq" --ont --
consensus --medaka \
      --primer_file /data/5.demux/primers.fasta \
      --outfolder "./NGspecies_Outer01_Inner${i}" --abundance_ratio
.08 \
      --rc_identity_threshold .92 --m 400 --s 60 --sample_size 300
    else
      echo "reads < 8: skipping barcode${i}"
    fi
    cd ../
  fi
done
```

This process make take hours to complete, depending on the quantity of reads being analyzed. Suggest using 'tmux' so that the operation continues even if the the comman-line window is accidentally closed.

Remain in the same Docker container for next step

Organizing Consensus Sequence and Metadata

16 Locate and remove any files that might be created that are totally empty.

Then, consolidate full paths to all consensus sequences generated; the path information is used to rename the sequences prior to comparing them with public reference records

Command

Remove empty files (linux)

```
cd /data/5.demux

# show fasta files that may be created which are totally empty
find . -type f -name 'consensus.fasta' -empty -print

# delete fasta files that may be created which are totally empty
find . -type f -name 'consensus.fasta' -empty -print -delete
```

Command

Make a list of paths to consensus sequences (linux)

```
cd /data/5.demux

# creates a text document in the working directory that provides the
full path to every consensus file that was created
find "$(pwd -P)" -type f -name 'consensus.fasta' >
./consensus_fullpaths.txt
```

Remain in the same Docker container for next step

17 Create a single fasta file with all consensus sequences, including any cases of more than one per sample.

Note

Do NOT attempt to run the following script in one copy-paste action. Read the mark-down and execute the script one command at a time.

Command

Custom R script for Finding Consensus Sequences (R)

```
# exit the NGSpeciesID environment
conda deactivate

#run r console
R

#load packages
library(stringr)
library(seqinr)

# import all the paths to the consensus files
pth<-read.table('consensus_fullpaths.txt',header = F)
names(pth)<-"path" # correct column name
pth$seqcount<-'' # add empty column
pth$consensus<-'' # add empty column
pth$reads<-'' # add empty column

pth$path<-as.character(pth$path)

#some consensus files have more than one sequence, referred to as
segments.
# here we count how many per file
for (i in 1:nrow(pth)) {
  print(i)
  pth$seqcount[i]<-length(names(read.fasta(pth$path[i])))
  print(pth$seqcount[i])
}

# check that there are only 1s and 2s
table(pth$seqcount)

# convert the sequence count variable to a number
pth$seqcount<-as.numeric(pth$seqcount)

#____ NOTE _______ from this point onward, execute each of the
commands line-by-line or the script will not complete (ignore error
messages from R) _______________

#replicate the rows that have more than 1 sequence
pth2<-pth[rep(1:nrow(pth),times=pth$seqcount),]
```

```
# when there are multiple sequences, number them sequentially to not
miss them during import, rarely there are two sequences per consensus
fasta, the code below will label the first as '1' and leave the
second as '2'
pth2[duplicated(pth2$path, fromLast = T),]$seqcount<-1
pth_dups<-pth2[duplicated(pth2$path)|duplicated(pth2$path, fromLast =
T),];pth_dups
#(ignore)pth_dups<-pth2[duplicated(pth2$path)|duplicated(pth2$path,
fromLast = T),]


# loop through all consensus files importing the sequence and the
header
for (i in 1:nrow(pth2)) {
  pth2$consensus[i]<-read.fasta(pth2$path[i],as.string = T, seqonly =
T )[[pth2$seqcount[i]]]
  pth2$reads[i]<-names(read.fasta(pth2$path[i]))[pth2$seqcount[i]]
}


#____ NOTE _______ adjust regex expressions as needed to extract
desired information_______________

# extract the number of supporting reads
pth2$reads<-str_remove(pth2$reads,'^.*supporting_reads_');pth2$reads<-
str_remove(pth2$reads,'_.*$')
# extract the consensus variants
pth2$variants<-str_extract(pth2$path,'(?<=Outer.._Inner..\\/).*?(?
=\\/consensus.fasta)')
#pth2$variants<-str_replace(pth2$variants,'\\/','_')
#extract the index for each consensus sequence
pth2$bc<-str_extract(pth2$path,'(?<=NGspecies_).*?(?=\\/medaka)')


# (OPTIONAL) subset to sequences with reads > 10
# (OPTIONAL) consensus[consensus$reads>=10,]

# write a fasta file that can be blasted to NCBI or bold
write.fasta(sequences = as.list(pth2$consensus),
            names = paste(pth2$bc,'_',pth2$variants,'_','reads_',
                          pth2$reads,sep=''),
            file.out = 'NGSpecies_all_consensus.fasta')
```

Remain in the same Docker container for the next step

BLASTn Analysis

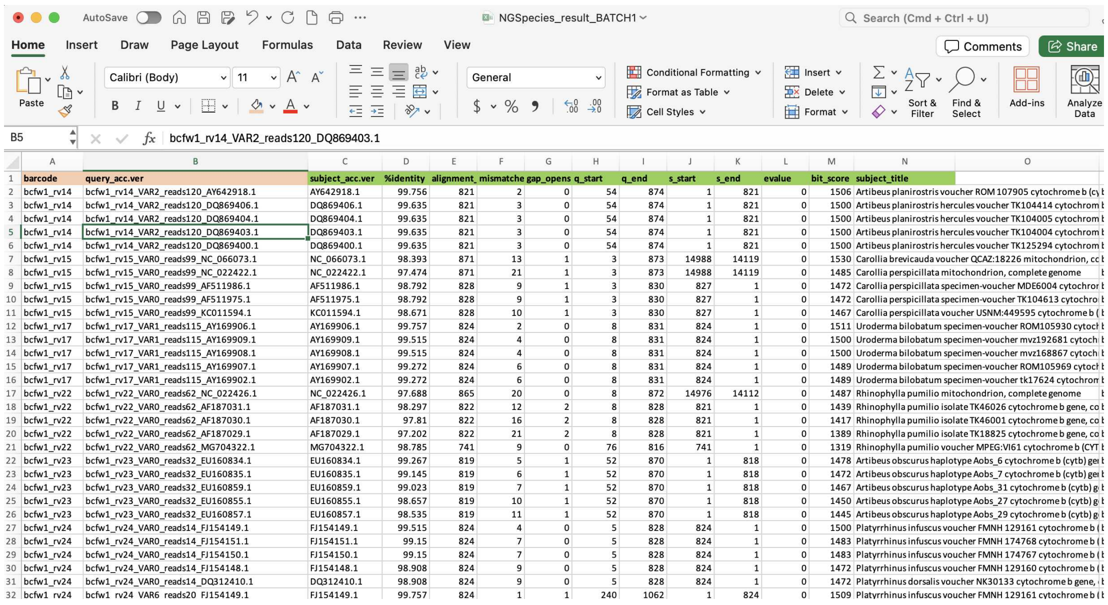
18 The first command will search for all matching records in NCBI GenBank, and may take hours to complete depending on the number of consensus sequences being analyzed. Consider adjusting the matching criteria and the output file according to preference.

Command

From inside docker image from previous step (linux)

```
#blast all consensus records (returning verbose "default" information
plus the "title" of each hit)
/software/ncbi-blast-2.12.0+/bin/blastn -query
/data/5.demux/NGSpecies_all_consensus.fasta -remote -db nt -outfmt "7
std stitle" -max_target_seqs 10 -out /data/sample_NCBIblastN_e6.txt -
evalue 1e-6

# OPTIONAL: this will remove all commented lines in the output file
so that it can be opened in a traditional spreadsheet viewer such as
Excel
sed '/^#/d' classification_NCBIblastN_e6.txt >
NCBIblastN_e6_minimal.txt
```

See an example of the output to explore the results.



NGSpecies_result_BATCH1

B5 bcfw1_rv14_VAR2_reads120_DQ869403.1

barcode	query_acc.ver	subject_acc.ver	%identity	alignment_	mismatche	gap_opens	q_start	q_end	s_start	s_end	evalue	bit_score	subject_title
bcfw1_rv14	bcfw1_rv14_VAR2_reads120_AY642918.1	AY642918.1	99.756	821	2	0	54	874	1	821	0	1506	Artibeus planirostris voucher ROM 107905 cytochrome b (cy
bcfw1_rv14	bcfw1_rv14_VAR2_reads120_DQ869406.1	DQ869406.1	99.635	821	3	0	54	874	1	821	0	1500	Artibeus planirostris hercules voucher TK104414 cytochrom
bcfw1_rv14	bcfw1_rv14_VAR2_reads120_DQ869404.1	DQ869404.1	99.635	821	3	0	54	874	1	821	0	1500	Artibeus planirostris hercules voucher TK104005 cytochrom
bcfw1_rv14	bcfw1_rv14_VAR2_reads120_DQ869403.1	DQ869403.1	99.635	821	3	0	54	874	1	821	0	1500	Artibeus planirostris hercules voucher TK104004 cytochrom
bcfw1_rv14	bcfw1_rv14_VAR2_reads120_DQ869400.1	DQ869400.1	99.635	821	3	0	54	874	1	821	0	1500	Artibeus planirostris hercules voucher TK125294 cytochrom
bcfw1_rv15	bcfw1_rv15_VAR0_reads99_NC_066073.1	NC_066073.1	98.393	871	13	1	3	873	14988	14119	0	1530	Carollia brevicauda voucher QCAZ:18226 mitochondrion, cc
bcfw1_rv15	bcfw1_rv15_VAR0_reads99_NC_022422.1	NC_022422.1	97.474	871	21	1	3	873	14988	14119	0	1485	Carollia perspicillata mitochondrion, complete genome
bcfw1_rv15	bcfw1_rv15_VAR0_reads99_AF511986.1	AF511986.1	98.792	828	9	1	3	830	827	1	0	1472	Carollia perspicillata specimen-voucher MDE6004 cytochror
bcfw1_rv15	bcfw1_rv15_VAR0_reads99_AF511975.1	AF511975.1	98.792	828	9	1	3	830	827	1	0	1472	Carollia perspicillata specimen-voucher TK104613 cytochro
bcfw1_rv15	bcfw1_rv15_VAR0_reads99_KC011594.1	KC011594.1	98.671	828	10	1	3	830	827	1	0	1467	Carollia perspicillata voucher USNM:449595 cytochrome b (
bcfw1_rv17	bcfw1_rv17_VAR1_reads115_AY169906.1	AY169906.1	99.757	824	2	0	8	831	824	1	0	1511	Uroderma bilobatum specimen-voucher ROM105930 cytocl
bcfw1_rv17	bcfw1_rv17_VAR1_reads115_AY169909.1	AY169909.1	99.515	824	4	0	8	831	824	1	0	1500	Uroderma bilobatum specimen-voucher mvz192681 cytoch
bcfw1_rv17	bcfw1_rv17_VAR1_reads115_AY169908.1	AY169908.1	99.515	824	4	0	8	831	824	1	0	1500	Uroderma bilobatum specimen-voucher mvz168867 cytoch
bcfw1_rv17	bcfw1_rv17_VAR1_reads115_AY169907.1	AY169907.1	99.272	824	6	0	8	831	824	1	0	1489	Uroderma bilobatum specimen-voucher ROM105969 cytocl
bcfw1_rv17	bcfw1_rv17_VAR1_reads115_AY169902.1	AY169902.1	99.272	824	6	0	8	831	824	1	0	1489	Uroderma bilobatum specimen-voucher tk17624 cytochrom
bcfw1_rv22	bcfw1_rv22_VAR0_reads62_NC_022426.1	NC_022426.1	97.688	865	20	0	8	872	14976	14112	0	1487	Rhinophylla pumilio mitochondrion, complete genome
bcfw1_rv22	bcfw1_rv22_VAR0_reads62_AF187031.1	AF187031.1	98.297	822	12	2	8	828	821	1	0	1439	Rhinophylla pumilio isolate TK46026 cytochrome b gene, co
bcfw1_rv22	bcfw1_rv22_VAR0_reads62_AF187030.1	AF187030.1	97.81	822	16	2	8	828	821	1	0	1417	Rhinophylla pumilio isolate TK46001 cytochrome b gene, co
bcfw1_rv22	bcfw1_rv22_VAR0_reads62_AF187029.1	AF187029.1	97.202	822	21	2	8	828	821	1	0	1389	Rhinophylla pumilio isolate TK18825 cytochrome b gene, co
bcfw1_rv22	bcfw1_rv22_VAR0_reads62_MG704322.1	MG704322.1	98.785	741	9	0	76	816	741	1	0	1319	Rhinophylla pumilio voucher MPEG:VI61 cytochrome b (CYT
bcfw1_rv23	bcfw1_rv23_VAR0_reads32_EU160834.1	EU160834.1	99.267	819	5	1	52	870	1	818	0	1478	Artibeus obscurus haplotype Aobs_6 cytochrome b (cytb) gei
bcfw1_rv23	bcfw1_rv23_VAR0_reads32_EU160835.1	EU160835.1	99.145	819	6	1	52	870	1	818	0	1472	Artibeus obscurus haplotype Aobs_7 cytochrome b (cytb) gei
bcfw1_rv23	bcfw1_rv23_VAR0_reads32_EU160859.1	EU160859.1	99.023	819	7	1	52	870	1	818	0	1467	Artibeus obscurus haplotype Aobs_31 cytochrome b (cytb) g
bcfw1_rv23	bcfw1_rv23_VAR0_reads32_EU160855.1	EU160855.1	98.657	819	10	1	52	870	1	818	0	1450	Artibeus obscurus haplotype Aobs_27 cytochrome b (cytb) g
bcfw1_rv23	bcfw1_rv23_VAR0_reads32_EU160857.1	EU160857.1	98.535	819	11	1	52	870	1	818	0	1445	Artibeus obscurus haplotype Aobs_29 cytochrome b (cytb) g
bcfw1_rv24	bcfw1_rv24_VAR0_reads14_FJ154149.1	FJ154149.1	99.515	824	4	0	5	828	824	1	0	1500	Platyrrhinus infuscus voucher FMNH 129161 cytochrome b (
bcfw1_rv24	bcfw1_rv24_VAR0_reads14_FJ154151.1	FJ154151.1	99.15	824	7	0	5	828	824	1	0	1483	Platyrrhinus infuscus voucher FMNH 174768 cytochrome b (
bcfw1_rv24	bcfw1_rv24_VAR0_reads14_FJ154150.1	FJ154150.1	99.15	824	7	0	5	828	824	1	0	1483	Platyrrhinus infuscus voucher FMNH 174767 cytochrome b (
bcfw1_rv24	bcfw1_rv24_VAR0_reads14_FJ154148.1	FJ154148.1	98.908	824	9	0	5	828	824	1	0	1472	Platyrrhinus infuscus voucher FMNH 129160 cytochrome b (
bcfw1_rv24	bcfw1_rv24_VAR0_reads14_DQ312410.1	DQ312410.1	98.908	824	9	0	5	828	824	1	0	1472	Platyrrhinus dorsalis voucher NK30133 cytochrome b gene,
bcfw1_rv24	bcfw1_rv24_VAR6_reads20_FJ154149.1	FJ154149.1	99.757	824	1	1	240	1062	1	824	0	1509	Platyrrhinus infuscus voucher FMNH 129161 cytochrome b (

In orange are the consensus codes and in green are the blast results.

Sequence Data Clean-Up

19 Use any desired software or program to merge the following into a single tab- or comma-delimited spreadsheet (one consensus sequence per row)

- Consensus sequences
- Sample metadata

Note

{r} can use the "seqinr" package to import the consensus fasta file into a data frame object that can be merged with any other data frame objects by the information in the sequence headers. Below, is an example of commands that could be used to create a data frame from the NGSpecies_all_consensus.fasta.

Command

seqinr_example (linux)

```
library(seqinr)

fasta_file_path <- "rugezi_2024-07-05_NGSpecies_all_consensus.fasta"

#### Function to read a FASTA file and convert it to a data frame
fasta_to_dataframe <- function(fasta_file_path) {
  # Read the FASTA file
  fasta_data <- read.fasta(file = fasta_file_path, seqtype = "AA",
as.string = TRUE)
  # Extract headers and sequences
  headers <- sapply(fasta_data, attr, "name")
  sequences <- sapply(fasta_data, getSequence)
  # Convert sequences to character vectors
  sequences <- sapply(sequences, function(seq) paste(seq, collapse =
""))
  # Create a data frame
  df <- data.frame(Header = headers, Sequence = sequences,
stringsAsFactors = FALSE)
  return(df)
}

consensus <- fasta_to_dataframe(fasta_file_path)
```

20 Visually inspect the results and consider using the following tools, as needed

- **MAFFT** alignment tool to check sequence directionality and its place on a phylogenetic tree to locate potential contamination and lab indexing errors
- **cutadapt** to remove PCR primer sites, which often introduce errors to the sequences.

Summarizing Data

21

The following script is unique to the format of the data that was used in our publication. It will not work in a generic sense on other data sets, but is provided as a record of what was performed for this publication and can serve as a valuable example to related studies in the future.

Command

custom r script (linux)

```
# last updated 2024-09-19 by GideonE

#______ Summary Information for 'Decoding the Amazon' ________#


#%%%%%%%%% Install then load required packages
library(dplyr)
library(ggplot2)
library(plyr)
library(reshape2)
library(readr)
library(tidyr)


#%%%%%%%%% Import all related metadata data
barcodes <- read.csv("lab_final.csv",header = TRUE) # sample sheet
from the molecular laboratory
taxa <- read.csv("taxonomy_final.csv",header = TRUE) # taxonomy data
from BOLD system
collection <- read.csv("collection_data_final.csv", header = TRUE) #
collection data fro BOLD system



#%%%%%%%%% Merge data frames together, 2 at a time, and then perform
some basic data clean-up
alldata <- merge(barcodes,collection, by="Sample.ID")

alldata <- merge(alldata,taxa, by="Sample.ID")

colnames(alldata) # show column names of combined data

# "Green Lab" site was stored in the "Extra.info" column, move this
to institution variable
table(alldata$Extra.Info) # view all unique values in 'Extra.info'
table(alldata$Institution) # view all unique values in 'Institution'
alldata[alldata$Extra.Info=="GreenLab",]$Institution<-"Green Lab"
table(alldata$Institution) # check 'Institution' again for 'Green Lab'

## remove variables that are not wanted for data summaries
```

```
names(alldata)
patterns <- c("Trace", "Accession", "Email", "Stage",
"Image","Contam","Tribe","Depth","GPS","Event","Coord") # define
search patterns for columns to remove
regex_pattern <- paste(patterns, collapse = "|") # insert patterns
into a list of "OR" regex expressions
cols_to_remove <- grep(regex_pattern, names(alldata), value = TRUE) #
create vector of column names that match any of the patterns
alldata2 <- alldata %>% select(-one_of(cols_to_remove)) # remove
columns that match any of the names in the vector
nrow(alldata2)

#remove all unwanted rows, mainly samples from the coast, to keep
data set to Amazonian secies.
# Define the Sample.ID values to be removed, all that are nonAmazonian
ids_to_remove <- c("TUNKI-0138", "TUNKI-0139", "TUNKI-0140", "TUNKI-
233", "TUNKI-0012",
                   "TUNKI-0079", "TUNKI-0080", "TUNKI-0081", "TUNKI-
0082", "TUNKI-0083",
                   "TUNKI-0094", "TUNKI-0095", "TUNKI-0264", "TUNKI-
0016", "TUNKI-0233")
alldata2 <- alldata2 %>% filter(!Sample.ID %in% ids_to_remove) #
remove target samples
nrow(alldata2) #count number of rows remaining in data set

## remove all values of "0[n]", replace with zeros
# Function to replace "0[n]" with blank
replace_zero_n_brackets <- function(x) {
  gsub("0\\[n\\]", "0", x)
}


# Apply the function to all columns of the data frame
alldata2 <- alldata2 %>% mutate_all(~ replace_zero_n_brackets(.))


#%%%%%%%% Data Summaries
nrow(alldata2) # counter records in data frame
names(alldata2) # show all variable names in data frame

## correctly identified (field identification versus genetic
reference identity)
alldata2$IDmatching<-ifelse(alldata2$Identification ==
alldata2$Species,1,0)
length(alldata2$IDmatching)
table(alldata2$IDmatching)
```

```
write.csv(alldata2[alldata2$IDmatching==0,],"mismatches.csv",row.names
 = F) # export mismatches to CVS file
```

Protocol references

Sahlin K, Lim MCW, Prost S. NGSpeciesID: DNA barcode and amplicon consensus generation from long-read sequencing data. Ecol Evol. 2021 Jan 11;11(3):1392-1398. doi: 10.1002/ece3.7146. PMID: 33598139; PMCID: PMC7863402.

```
## sequences by order
orders<-ddply(alldata2,.(Order),nrow) # count number of sequences per
Order
names(orders)[2]<-"count" # correct variable name
write.csv(ddply(alldata2,.(Order),nrow),"order.csv") # export
sequence count by Order

## pie charts of sequences by Order for each taxonomic Class
class.orders<-ddply(alldata2,.(Class,Order),nrow) # count sequences
by Order by Class
names(class.orders)[3]<-"count" # correct variable name

# create list of data frames for each Class
class.orders_l<-split(class.orders,class.orders$Class)

# function to generate pie chart
generate_pie_chart <- function(df) {
  df <- df %>%
    arrange(desc(count)) %>%
    mutate(
      prop = count / sum(count) * 100,
      Order = paste(Order, "(", round(prop, 1), "%)", sep = "")
    )
  # Determine number of legend columns
  legend_cols <- ifelse(nrow(df) > 6, 2, 1)

  ggplot(df, aes(x = "", y = prop, fill = Order)) +
    geom_bar(stat = "identity", width = 1, color = "white") +
    coord_polar("y", start = 0) +
    theme_void() +
    scale_fill_viridis_d() +
    labs(title = paste(unique(df$Class)), fill = "Order") +
    guides(fill = guide_legend(ncol = legend_cols))
}
# run function across all objects in the list of data frames
plots <- lapply(class.orders_l, generate_pie_chart)

# export each plot in turn
ggsave(filename = "plot1.png",plot = plots[[1]], device = 'png',width
= 800, height = 700, dpi = 150, units = "px", bg="white")
= 800, height = 700, dpi = 150, units = "px", bg="white")
ggsave(filename = "plot2.png",plot = plots[[2]], device = 'png',width
= 800, height = 700, dpi = 150, units = "px", bg="white")
ggsave(filename = "plot3.png",plot = plots[[3]], device = 'png',width
= 800, height = 700, dpi = 150, units = "px", bg="white")
ggsave(filename = "plot4.png",plot = plots[[4]], device = 'png',width
= 800, height = 700, dpi = 150, units = "px", bg="white")
ggsave(filename = "plot5.png",plot = plots[[5]], device = 'png',width
= 800, height = 700, dpi = 150, units = "px", bg="white")

# display all plots in single figure
grid.arrange(grobs = plots, ncol = 2)


## sequences by family
write.csv(ddply(alldata2,.(Family),nrow),"family.csv")

## sequences by subfamily
nrow(ddply(alldata2,.(Subfamily),nrow))

## sequences by genus
nrow(ddply(alldata2,.(Genus),nrow))

## sequences by species
nrow(ddply(alldata2,.(Species),nrow))

## sequences by subspecies
nrow(ddply(alldata2,.(Subspecies),nrow))

## Table 4: summary table per gene marker
names(alldata2) # show variable names of full data set
ddply(alldata2,.(Class),nrow) # number of individuals per Class
names(alldata2)[7:12]<-c("RBCL","18S","CO1","MATK","CYTB","TRNH-
PSBA") # correct gene marker names
unique(alldata2$Class) # display all unique Class values
alldata3<-alldata2[c(1,39,44,7:12)] # make filtered, working copy of
data set
names(alldata3) # show varible names of filtered dataset
# convert all gene markeres to binary 1 (present) 0 (absent)
alldata3[4:9]<-alldata3[4:9] %>% mutate_all(~ifelse(is.na(.), NA,
ifelse(. != 0, 1, 0)))
alldata3.melt<-melt(alldata3,id.vars =
c("Sample.ID","Class","Species")) # transform filtered data to long
format
alldata3.melt<-alldata3.melt[alldata3.melt$value>0,] # remove all
records with a sequence
alldata3 class<-ddply(alldata3.melt,.(Class),nrow);alldata3 seq #
```

```
count number of sequences per Class
alldata3_gene_seq<-ddply(alldata3.melt,.
(Class,variable),nrow);alldata3_seq # count number of sequences per
gene per Class
alldata3_class<-ddply(alldata3.melt,.
(Class,Species),nrow);alldata3_class # count species per Class
alldata3_sum1<-ddply(alldata3.melt,.
(Class,Species,variable),summarize,seq_num=sum(value));alldata3_sum1
# count sequences per species per gene per Class
alldata3_sum2<-ddply(alldata3_sum1,.
(Class,variable),nrow);alldata3_sum2 # species per gene per class



#%%%%%%%% complimenting the Mammal gap analysis (Pacheco, V. et al.
Disproportion between the Peruvian Amazonian megadiverse mammalian
fauna and the available molecular information. Zoologia 41, e23110
(2024).)

gap_mammals<-read.csv('Gaps_mammals.csv') # import data table from
publication
names(gap_mammals) # show variable names
gap_mammals<-gap_mammals[-9] # filter unwanted variable
names(gap_mammals)[5:8]<-
c("peru_12S","peru_CO1","peru_CYTB","peru_MIT");names(gap_mammals) #
rename gene marker variables with 'peru' prefix
gap_mammals<-gap_mammals %>% mutate_at(vars(names(gap_mammals)[5:8]),
 ~replace(., . == "N", "0")) # convert gene markers to binary
variables
gap_mammals<-gap_mammals %>% mutate_at(vars(names(gap_mammals)[5:8]),
as.integer) # ensure gene markers variable is an integer
gap_mammals$peru_prior_seq <- rowSums(gap_mammals[5:8]) # sum across
all available gene markers per species and store in new column
gap_mammals <- gap_mammals %>% mutate(Species = gsub('"', '',
Species)) # remove quotation marks
gap_mammals[c(5:9)] <- gap_mammals[c(5:9)] %>% mutate_all(~ifelse(.
!= "0", "1", "0")) # convert data to binary again
gap_mammals[c(5:9)] <- gap_mammals[c(5:9)] %>% mutate_all(as.integer)
 # ensure binary gene marker values are integers
colSums(gap_mammals[c(5:9)]) # count sequence records for each column
(last column indicates any sequence record for a species)


names(alldata2) # show variable names of new data set
alldata2m<-
alldata2[alldata2$Class=="Mammalia",c(1,39,43:44,7:12)];names(alldata2
m) # filter to only mammals and variables of interest, and make new
working data frame
names(alldata2m)[5:10]<-
c("RBCL","g18S","CO1","MATK","CYTB","TRNH");names(alldata2m) #
correct gene marker names
alldata2m[5:10] <- alldata2m[5:10] %>% mutate_all(~ifelse(. != "0",
"1", "0")) # make gene marker values binary
alldata2m[5:10] <- alldata2m[5:10] %>% mutate_all(as.integer) #
ensure binary gene marker values are integers

# count number of sequences per gene marker by species
alldata2m<-ddply(alldata2m,.(Class,Genus,Species),summarize,
      us_RBCL=sum(RBCL,na.rm=T),
      us_18S=sum(g18S,na.rm=T),
      us_CO1=sum(CO1,na.rm=T),
      us_MATK=sum(MATK,na.rm=T),
      us_CYTB=sum(CYTB,na.rm=T),
      us_TRNH=sum(TRNH,na.rm=T))

names(alldata2m) # check variable names
alldata2m[4:9]<-alldata2m[4:9] %>% mutate_all(~ifelse(. >= 1, 1, .))
# convert gene marker variabales to binary values

names(alldata2m) # check variable names
alldata2m$peru_seq_new<-rowSums(alldata2m[4:9]) # create new variable
of gene markers generated per species

mmerge2<-merge(alldata2m,gap_mammals,by = "Species", all=T) # merge
new and prior data sets
colSums(mmerge2[18],na.rm = T)

table(mmerge2$Species) # count number of all species in merged data
mmerge2<-mmerge2[mmerge2$Species!="",] # remove empty rows

# following command will count how many mammals of the prior data
list were included in this study
table(mmerge2$Class, useNA = "ifany")

names(mmerge2) # show variable names of merged data
mmerge2[c(4:10,14:18)] <- mmerge2[c(4:10,14:18)] %>%
mutate_all(~replace_na(., 0)) # replace NA values with 0s
mmerge2[c(4:10,14:18)] <- mmerge2[c(4:10,14:18)] %>%
mutate_all(~ifelse(. > 1, 1, .)) # make all values binary

## calculate how many new sequences per species per gene
names(mmerge2) # show variable names of merged data
```

```
#how many species that had no prior genetic record
nrow(gap_mammals[gap_mammals$peru_prior_seq==0,]) # 146 total

# calculate several new values per species
mmerge2 <- mmerge2 %>%
  mutate(
    CO1_diff=ifelse(us_CO1 == 1 & peru_CO1 == 0,1,0), # value of 1 is
we generated a COI record that did not previously exist
    CYTB_diff=ifelse(us_CYTB == 1 & peru_CYTB == 0,1,0), # value of 1
if we generated a cytB record that did not previously exit
    new_species_ref=ifelse(peru_seq_new ==1 & peru_prior_seq ==
0,1,0) # value of 1 if we generated a sequence record for a species
that had none
  )
names(mmerge2)

# sum values for all variables to count total number of new gene and
species records
colSums(mmerge2[c(4:10,14:21)])

new_mammals<-mmerge2[mmerge2$new_species_ref==1,] # subset to just
mammals in which a first gene record was generated by this study

colSums(new_mammals[c(4:10,14:21)]) # sum values for all variables



#%%%%%%%% Complimenting the Bird gap analysis (Arana, A. et al. Lack
of local genetic representation in one of the regions with the
highest bird species richness, the Peruvian Amazonia. PLoS One 19,
e0296305 (2024).)
gap_bird<-read.csv('Gaps_aves.csv') # import data table from
publication
gap_bird2<-gap_bird # create working data frame
names(gap_bird2) # show variable names
names(gap_bird2)[5:10]<-
c("Amazonian_birds","CO1_peru","CO1_world","CYTB_world","18S_world","1
2S_world");names(gap_bird2) # rename gene marker variables with
'peru' prefix
gap_bird2<-gap_bird2[!is.na(gap_bird2$Amazonian_birds),] # filter to
only Amazonian birds
table(gap_bird2$CO1_world) # show table of records in 'CO1_workd"
variable
gap_bird2$CO1_world[is.na(gap_bird2$CO1_world)]<-"N" # insert "N" for
all records that are NA
gap_bird2$CO1_world[gap_bird2$CO1_world %in% c("public","private")]<-
```

```
"K" # replace public and private values with "K"
gap_bird2<-gap_bird2 %>%
mutate_at(vars("CO1_peru","CO1_world","CYTB_world","18S_world","12S_wo
rld"),  ~ifelse(is.na(.) | . != "K", 0, 1)) # make all gene marker
variables binary values

names(gap_bird2) # show variable names
colSums(gap_bird2[6:10]) # sum values for all gene marker variables
in prior data set


names(alldata2) # show variable names of new data set
alldata2b<-
alldata2[alldata2$Class=="Aves",c(1,39,44,7:12)];names(alldata2b) #
filter to only birds and variables of interest, and make new working
data frame
names(alldata2b)[4:9]<-
c("RBCL","g18S","CO1","MATK","CYTB","TRNH");names(alldata2b) #
correct gene marker names
alldata2b[4:9] <- alldata2b[4:9] %>% mutate_all(~ifelse(. != "0",
"1", "0")) # make gene marker values binary
alldata2b[4:9] <- alldata2b[4:9] %>% mutate_all(as.integer) # ensure
binary gene marker values are integers

# count number of sequences per gene marker by species
alldata2b<-ddply(alldata2b,.(Class,Species),summarize,
                 us_RBCL=sum(RBCL,na.rm=T),
                 us_18S=sum(g18S,na.rm=T),
                 us_CO1=sum(CO1,na.rm=T),
                 us_MATK=sum(MATK,na.rm=T),
                 us_CYTB=sum(CYTB,na.rm=T),
                 us_TRNH=sum(TRNH,na.rm=T))

alldata2b[3:8]<-alldata2b[3:8] %>% mutate_all(~ifelse(. >= 1, 1, .))
# convert gene marker variabales to binary values

names(alldata2b) # check variable names
alldata2b$peru_new_seq<-rowSums(alldata2b[3:8]) # create new variable
of gene markers generated per species

bmerge<-merge(alldata2b, gap_bird2, by.x="Species",
by.y="Scientific_name", all=T) # merge new and prior data sets

bmerge<-bmerge[bmerge$Species!="",] #remove blank records after merge

# how many birds and mammals did we cover of the existing species list
```

```
table(bmerge$Class, useNA = "ifany")

names(bmerge)  # check variable names
bmerge[c(3:9,14:18)] <- bmerge[c(3:9,14:18)] %>%
mutate_all(~replace_na(., 0)) # replace NA values with 0s
bmerge$world_prior_seq <- rowSums(bmerge[14:18]);names(bmerge) # sum
values across prior sequence records per species
bmerge[c(3:9,14:19)] <- bmerge[c(3:9,14:19)] %>% mutate_all(~ifelse(.
> 1, 1, .)) # convert all numerical values to binary

names(bmerge) # check variable names

# calculate several new values per species
bmerge <- bmerge %>%
  mutate(
    CO1peru_diff=ifelse(us_CO1 == 1 & CO1_peru == 0,1,0), # value of
1 if new COI record from Peruvian specimen generated by this study
    CO1world_diff=ifelse(us_CO1 == 1 & CO1_world == 0,1,0), # value
of 1 if new CO1 record for each species was generated by this study
    CYTBworld_diff=ifelse(us_CYTB == 1 & CYTB_world == 0,1,0), #
value of 1 if new cytB record for each species was generated by this
study
    x18Sworld_diff=ifelse(us_18S == 1 & '18S_world' == 0,1,0), #
value of 1 if new 18S record for each species was generated by this
study
    new_species_ref=ifelse(peru_new_seq == 1 & world_prior_seq ==
0,1,0) # value of 1 if first gene marker reference for each species
was generated by this study
  )
names(bmerge)

# sum values for all variables to count total number of new gene and
species records
colSums(bmerge[c(3:9,14:24)])

# subset to just mammals in which a first gene record was generated
by this study
new_birds<-bmerge[bmerge$new_species_ref==1,]

# sum values for all variables
colSums(new_birds[c(3:9,14:24)])
```