Developing a non-destructive metabarcoding protocol for detection of target insects in bulk trap catches

J. Batovska, A.M. Piper, I. Valenzuela, J.P. Cunningham & M.J. Blacket

2019/10/15

Introduction

This is the R based reproducible workflow that performed the metabarcoding analyses presented for the manuscript "Developing a non-destructive metabarcoding protocol for detection of pest insects in bulk trap catches" by J. Batovska, A.M. Piper, I. Valenzuela, J.P. Cunningham & M.J. Blacket

The data that was analysed here includes 20 mock communities made up of Hemiptera, and 10 trap samples from a potato field. These were seperated over 3 runs on an illumina MiSeq

Run_1 - 15 pools of 100, 500, and 1000 insects Run_2 - 5 pools of 250 insects, run in duplicate to compare combinatorial indexing and unique-dual indexing Run_3 - 10 trap samples with varying numbers of insects

3 seperate loci were multiplexed in the one PCR amplification, and libraries were prepared from these

In this analysis pipeline, each MiSeq run was processed processed within a for-loop, so parameters are kept consistant

Remove primers

DADA2 requires Non-biological nucleotides i.e. primers, adapters, linkers, etc to be removed. Prior to begining this workflow, samples were demultiplexed and illumina adapters were removed by the MiSeq software, however primer sequences still remain in the reads and must be removed prior to use with the DADA2 algorithm.

In this study there were 3 amplicons of different size locirated in a multiplexed PCR. While COI should only contain the 5' primer, the 12S and 18S loci are length variable, and therefore may contain 3' primer sequences in addition to the 5'primer.

Therefore, for this workflow we will be using the Kmer based adapter trimming software BBDuk (Part of BBTools package https://jgi.doe.gov/data-and-tools/bbtools/) to trim the primers from our raw data files.

Name	Illumina overhang adapter	Primer sequences
Sterno18S_F2_tail	ACACTCTTTCCCTACACGACGCTCTTCCGATCT	ATGCATGTCTCAGTGCAAG
Sterno18S_R1_tail	GACTGGAGTTCAGACGTGTGCTCTTCCGATC	TCGACAGTTGATAAGGCAGAC
Sterno12S_F2_tail	ACACTCTTTCCCTACACGACGCTCTTCCGATCT	CAYCTTGACYTAACAT
Sterno12S_R2_tail	GACTGGAGTTCAGACGTGTGCTCTTCCGATC	TAAAYYAGGATTAGATACCC
SternoCOI_F1_tail	ACACTCTTTCCCTACACGACGCTCTTCCGATCT	ATTGGWGGWTTYGGAAAYTG
SternoCOI_R1_tail	GACTGGAGTTCAGACGTGTGCTCTTCCGATC	TATRAARTTRATWGCTCCTA

```
mkdir cleaned
ls | grep "R1_001.fastq.gz" | sort > test_ls_F
ls | grep "R2_001.fastq.gz" | sort > test_ls_R

let files=$(grep -c "fastq.gz" test_ls_F)

declare -i x
```

```
x=1

while [ $x -le $files ]
    do

queryF=$(sed -n "${x}p" test_ls_F)
queryR=$(sed -n "${x}p" test_ls_R)

sample_nameF=$(echo $queryF | awk -F . '{ print $1}')
sample_nameR=$(echo $queryR | awk -F . '{ print $1}')

#Need to set location of bbduk - on basc it is contained in my root ~/bbmap/bbduk.sh

#Trim 3' primers from forward and reverse reads and any bases to the left
~/bbmap/bbduk.sh in=$sample_nameF.fastq.gz \
in2=$sample_nameR.fastq.gz \
out=$sample_nameF.temp.fastq.gz \
out2=$sample_nameR.temp.fastq.gz \
literal=ATTGGWGGWTTYGGAAAYTG,TATRAARTTRATWGCTCCTA,ATGCATGTCTCAGTGCAAG, \
TCGACAGTTGATAAGGCAGAC,CAYCTTGACYTAACAT,TAAAYYAGGATTAGATACCC \
===copyundefined k=14 ordered=t rcomp=f ktrim=l tbo tpe;

#Trim 5' primers from forward and reverse reads and any bases to the right
~/bbmap/bbduk.sh in=$sample_nameF.temp.fastq.gz \
in2=$sample_nameR.temp.fastq.gz \
out=./cleaned/$sample_nameF.trimmed.fastq.gz \
out2=./cleaned/$sample_nameR.trimmed.fastq.gz \
literal=GGGTATCTAATCCTRRTTTA,ATGTTARGTCAAGRTG \
copyundefined k=14 ordered=t rcomp=f ktrim=r tbo tpe;

let x=x+1

done 2> bbduk_primer_trimming_stats.txt
rm *.temp.*
```

Metabarcoding analysis with DADA2

Set up analysis

Load all requried packages, and import helper functions from scripts folder

```
sapply(c("dada2", "phyloseq","ggplot","ips", "DECIPHER",
         "data.table", "tidyverse","Biostrings",
         "ShortRead","scales","stringdist","patchwork",
         "psadd","ggpubr","seqinr", "limma", "ggforce",
         "viridis", "data.tree", "ggtree"),
       require, character.only = TRUE)

source('scripts/helper_functions.R')
```

Error visualisation

We start by visualizing the quality profiles of the forward read and reverse reads for each run:

```r
runs <- dir("data/", pattern="run")

for (i in seq(along=runs)){
path <- paste0("data/",runs[i])
fastqFs <- sort(list.files(path, pattern="R1_001.trimmed.fastq.gz", full.names = TRUE))
fastqRs <- sort(list.files(path, pattern="R2_001.trimmed.fastq.gz", full.names = TRUE))
  p1 <- plotQualityProfile(fastqFs[1:4]) + ggtitle(paste0(runs[i]," Forward Reads"))
  p2 <- plotQualityProfile(fastqRs[1:4]) + ggtitle(paste0(runs[i]," Reverse Reads"))
  print(p1+p2)
}
```

Filter and trim

The forward reads for the hemiptera metabarcoding data are of good quality, while The reverse reads are of slightly worse worse quality at the end, which is common in Illumina sequencing. Informed by these profiles, we will use the Truncate quality function (TruncQ=2) to cut the reads at any point the Q score crashes below 2.

```r
runs <- dir("data/", pattern="run")
filtered_out <- list()

for (i in seq(along=runs)){
  path <- paste0("data/",runs[i])
  filtpath <- file.path(path, "filtered")

  fastqFs <- sort(list.files(path, pattern="R1_001.trimmed.fastq.gz"))
  fastqRs <- sort(list.files(path, pattern="R2_001.trimmed.fastq.gz"))

  if(length(fastqFs) != length(fastqRs)){
    stop(paste0("Forward and reverse files for ",runs[i]," do not match."))
  }

  filtered_out[[i]] <- (filterAndTrim(fwd=file.path(path, fastqFs),
                                      filt=file.path(filtpath, fastqFs),
                                      rev=file.path(path, fastqRs),
                                      filt.rev=file.path(filtpath, fastqRs),
                                      maxEE=c(2,5), truncQ=2, maxN = 2, minLen = 100,
                                      rm.phix=TRUE, compress=TRUE, verbose=TRUE))
}

print(filtered_out)
```

Post filtering error plotting

sanity check to see the effects of the filter and trim step

```r
runs <- dir("data/", pattern="run")

for (i in seq(along=runs)){
  path <- paste0("data/",runs[i])
  filtpath <- file.path(path, "filtered")

  filtFs <- sort(list.files(filtpath,
                            pattern="R1_001.trimmed.fastq.gz",
                            full.names = TRUE))
  filtRs <- sort(list.files(filtpath,
                            pattern="R2_001.trimmed.fastq.gz",
                            full.names = TRUE))
  p1 <- plotQualityProfile(filtFs[1:4]) +
    ggtitle(paste0(runs[i]," Filtered Forward Reads"))
  p2 <- plotQualityProfile(filtRs[1:4]) +
    ggtitle(paste0(runs[i]," Filtered Reverse Reads"))
  print(p1+p2)
}
```

Infer sequence variants

Every amplicon dataset has a different set of error rates and the DADA2 algorithm makes use of a parametric error model (err) to model this and infer real biological sequence variation from error. Following error model learning, all identical sequencing reads are dereplicated into into "Exact sequence variants" with a corresponding abundance equal to the number of reads with that unique sequence. The forward and reverse reads are then merged together by aligning the denoised forward reads with the reverse-complement of the corresponding reverse reads, and then constructing the merged "contig" sequences. Following this step, a sequence variant table is constructed and saved as an RDS file.

For this analysis we will use all the reads to estimate error rate, and plot the error model for each run as a sanity check

```r
runs <- dir("data/", pattern="run")
set.seed(100)

for (i in seq(along=runs)){
  path <- paste0("data/",runs[i])
  filtpath <- file.path(path, "filtered")

  filtFs <- list.files(filtpath, pattern="R1_001.trimmed.fastq.gz", full.names = TRUE)
  filtRs <- list.files(filtpath, pattern="R2_001.trimmed.fastq.gz", full.names = TRUE)

  # Assumes filename = flowcell_samplename_XXX.fastq.gz
  sample.names <- sapply(strsplit(basename(filtFs), "_"), `[`, 1)
  sample.namesR <- sapply(strsplit(basename(filtRs), "_"), `[`, 1)
  if(!identical(sample.names, sample.namesR)){
    stop("Forward and reverse files from run1 do not match.")
  }
  names(filtFs) <- sample.names
  names(filtRs) <- sample.names

  # Learn error rates from samples
  errF <- learnErrors(filtFs, multithread=TRUE)
  errR <- learnErrors(filtRs, multithread=TRUE)

  ##Print error plots to check fit
  print(plotErrors(errF, nominalQ=TRUE) +
          ggtitle(paste0(runs[i]," Forward Reads")))
  print(plotErrors(errR, nominalQ=TRUE) +
          ggtitle(paste0(runs[i]," Reverse Reads")))

  #infer variants and merge of reads
  mergers <- vector("list", length(sample.names))
  names(mergers) <- sample.names
  for(sam in sample.names) {
    cat("Processing:", sam, "\n")
    derepF <- derepFastq(filtFs[[sam]])
    ddF <- dada(derepF, err=errF, multithread=TRUE)

    derepR <- derepFastq(filtRs[[sam]])
    ddR <- dada(derepR, err=errR, multithread=TRUE)
    merger <- mergePairs(ddF, derepF, ddR, derepR)
    mergers[[sam]] <- merger
  }

# Construct sequence table
seqtab<- makeSequenceTable(mergers)
saveRDS(seqtab, paste0(path,"/seqtab.rds"))
}
```

Merge Runs, Remove Chimeras

Now that the sequence tables are created for each run, they need to be merged into a larger table representing the entire study. Following this, chimeric sequences are identified and removed using removeBimeraDenovo, and any identical sequences with the only difference being length variation are collapsed using collapseNoMismatch.

```
runs <- dir("data/", pattern="run")
stlist <- vector()

for (i in seq(along=runs)){
  path <- paste0("data/",runs[i])
  seqs <- list.files(path, pattern="seqtab.rds", full.names = TRUE)

  assign(paste("st", i, sep = ""),readRDS(seqs))
  stlist <- append(stlist, paste("st", i, sep = ""), after=length(seqs))
}

st.all <- mergeSequenceTables(st1, st2, st3)

st.all <- collapseNoMismatch(st.all, minOverlap = 20, orderBy = "abundance",
                                    vec = TRUE, verbose = TRUE)
seqtab.nochim <- removeBimeraDenovo(st.all, method="consensus",
                                    multithread=TRUE, verbose=TRUE)
```

```
print(paste(sum(seqtab.nochim)/sum(st.all),"of non-chimeric abundance remaining"))

saveRDS(seqtab.nochim, "output/rds/seqtab_final.rds")
```

Assign Taxonomy to sequence variants

```
seqtab.nochim <- readRDS("output/rds/seqtab_final.rds")

# Assign Kingdom:Genus taxonomy using RDP classifier
tax <- assignTaxonomy(seqtab.nochim, "reference/merged_arthropoda_rdp_genus.fa",
                      multithread=TRUE, minBoot=80, outputBootstraps=FALSE)
colnames(tax) <- c("loci", "Phylum", "Class", "Order", "Family", "Genus")

##add species to taxtable using exact matching
tax_plus <- addSpecies(tax, "reference/merged_arthropoda_rdp_species.fa",
                       allowMultiple=TRUE)

##Add Spp. to species rank for those with only a genus rank assignmnet
for(col in seq(7,ncol(tax_plus))) {
  propagate <- is.na(tax_plus[,col]) & !is.na(tax_plus[,col-1])
  tax_plus[propagate,col:ncol(tax_plus)] <-  "spp."
}

##join genus and species name in species rank column
sptrue <- !is.na(tax_plus[,7])
tax_plus[sptrue,7] <- paste(tax_plus[sptrue,6],tax_plus[sptrue,7], sep=" ")

#Check Output
taxa.print <- tax_plus # Removing sequence rownames for display only
rownames(taxa.print) <- NULL
head(taxa.print)

# Write taxonomy table to disk
saveRDS(tax_plus, "output/rds/tax_RDP_final.rds")
```

Make Phyloseq object

Following taxonomic assignment, the sequence table and taxonomic table are merged into a single phyloseq object alongside the sample info csv.

```
seqtab.nochim <- readRDS("output/rds/seqtab_final.rds")
tax_plus <- readRDS("output/rds/tax_RDP_final.rds")

##Correct for synonyms in tax table
tax_plus[,7] <- tax_plus[,7] %>%
  str_replace_all("rufiabdominale/rufiabdominalis","rufiabdominale") %>%
  str_replace_all("insertum/oxyacanthae","insertum")

#Load sample information
## ---- samdat ----
```

```
samdf <- read.csv("sample_data/Sample_info.csv", header=TRUE)
samdf <- samdf[!duplicated(samdf$SampleID),] #Remove duplicate entries for reverse reads

rownames(samdf) <- samdf$SampleID
keep.cols <- c("collection_date", "biome", "target_gene", "feature",
"pool_comp" ,"SampleID","experimental_factor")
samdf <- samdf[rownames(seqtab.nochim), keep.cols]

#Display samDF
head(samdf)

## ---- phyloseq ----
ps <- phyloseq(tax_table(tax_plus), sample_data(samdf),
               otu_table(seqtab.nochim, taxa_are_rows = FALSE))
##save phyloseq object
saveRDS(ps, "output/rds/ps_rdp.rds")
```

Calculate index switch rate

While using unique dual-indices will allow detection and removal of the majority of index switch reads, there will still be low level undetectable index switching present at a rate of obs/exp^2 (ref-). to determine this rate, we will first calculate the unexpected index combinations compared to the expected.

Fastq files contain the index information for each read in the read header, and therefore to get all undetermined indices, both switched and otherwise erroneous we can summarise the index sequences for each read as contained in the fasta header:

```
@M03633:307:000000000-D4262:1:1101:19524:28535 1:N:0:**GAGACGAT+GTTCTCGT**
ATACTGTGCGTACTGCAGATCGGAAGAGCACACGTCTGAACTCCAGTCACGAGACGATATCTCGTATGCCGTCTT +
BBBB?FFFBABAEGGGGGGGGGGGGGGFHHHHHHGHHHGHHHHHHHHHHHHHGGEEEEEAFGGHGHFAHHHHGGGH
```

First we need to demultiplex the data again allowing no mismatches - This needs to be done with bcl2fastq rather than the default illumina miseq lociratefastq workflow, as the workflow doesnt include indices in fastq file headers

```
###BASH###
/home/ap0y/usr/local/bin/bcl2fastq -p 12 --runfolder-dir \
/group/sequencing/180309_M03633_0209_000000000-BN3J6  \
--output-dir /group/students/Alexp/Metabarcoding/hemiptera/demulti_mocks_0mis \
--sample-sheet /group/students/Alexp/Metabarcoding/hemiptera/SampleSheet_run4.csv \
--no-lane-splitting --barcode-mismatches 0
```

Then we can summarise the switch rate

```
###BASH###

#summarise undetermined reads from R1 undetermined file
zcat Undetermined_S0_L001_R1_001.fastq.gz | grep '^@M03633' \
| cut -d : -f 10 | sort | uniq -c | sort -nr > undetermined.txt

# summarise correctly determined reads from all other R1 files
```

```
rm determined.txt
ls | grep "R1_001.fastq.gz" | sort | grep -v 'Undetermined' > test_ls_F

let files=$(grep -c "fastq.gz" test_ls_F)

declare -i x

x=1
while [ $x -le $files ]
    do

query=$(sed -n "${x}p" test_ls_F)

sample_name=$(echo $query | awk -F . '{ print $1}')

stats=$(zcat $(echo $query) | grep '^@M03633' | wc -l)
echo $query $stats >> determined.txt

let x=x+1

done
rm test_ls_F
```

To differentiate unused indices arising from switching, from unused indices arrising from other phenomena, we can compare the undetermined count file to all possible combinations of i5 and i7 indices that could be produced through switching

```
#Read in original sample sheet
SampleSheet <- read_csv("SampleSheet_run4.csv",skip=20)

##enumerate 1 mismatch to all indices to mimic demultiplexing with a single mismatch

I7_Index_ID <- c(sapply(SampleSheet$index,create_mismatch,dist=1))
I5_Index_ID  <- c(sapply(SampleSheet$index2,create_mismatch,dist=1))

#Create all possible switched combinations
combos <- expand.grid(I7_Index_ID, I5_Index_ID)
combos$indices <- paste0(combos$Var1,"+",combos$Var2)

#Determined reads from mock communities
determined <- read_table2("determined.txt",col_names = FALSE)
colnames(determined) <- c("Sample_Name","count")
determined$Sample_Name <- determined$Sample_Name %>%
  str_replace_all("-","_") %>%
  str_split_fixed("_S",n=2)
determined$Sample_Name <- determined$Sample_Name[,1]
determined <- left_join(determined, SampleSheet, by="Sample_Name")
determined$indices <- paste0(determined$index, "+",determined$index2)
determined <- determined %>%
  subset(select=c("Sample_Name","count","indices"))
head(determined)

#Undetermined reads from mock communities
```

```r
undetermined <- read_table("undetermined.txt",col_names = FALSE)
colnames(undetermined) <- c("count","indices")
undetermined$Sample_Name <- "Undetermined_S0_R1_001.trimmed.fastq.gz"
head(undetermined)

indices <- rbind(determined,undetermined)

#Calculate total read count for run
total_reads <- sum(indices$count)

#get unused combinations resulting from index switching
switched <- left_join(combos,indices,by="indices")
colnames(switched) <- c("i7","i5","indices","Sample_Name","count")

#get unused combinations resulting from other phenomena
other <- indices[!indices$indices %in% combos$indices, ]

#Count number of other undetermined
other_reads <- sum(other$count)

##Summary of index switching rate
exp_rate <- switched %>%
  filter(str_detect(Sample_Name,"Pool"))
obs_rate <- switched %>%
  filter(!str_detect(Sample_Name,"Pool"))

switch_rate <- (sum(obs_rate$count)/sum(exp_rate$count))
message(switch_rate)

#Rate of undetected switching should be switch_rate squared
filt_threshold <- switch_rate^2
message(paste0("The threshold for filtering will be: ",filt_threshold))
```

Filter and output tables of results

Proportions function is defined, and filtering function which removes all taxa under 0.00011664 which was established by looking at expected vs observed reads, and then converts them to proportions. We output a number of summary tables pre and post filtering for the supplementary methods

```r
ps <- readRDS("output/rds/ps_rdp.rds")

##Export raw csv
export <- psmelt(ps)
write.csv(export, file = "output/csv/unfiltered/rawdata.csv")

#Agglomerate all OTU's to loci level and export proportions - For supplementary table
sum_loci <- transform_sample_counts(ps, fun = proportions)
sum_loci <- summarize_taxa(sum_loci, "loci", "SampleID")
sum_loci <- spread(sum_loci, key="SampleID", value="totalRA")

write.csv(sum_loci, file = "output/csv/unfiltered/loci_summarized.csv")
```

```r
#Subset data to Athropoda only & Export CSV
ps_arthropod = subset_taxa(ps, Phylum == "Arthropoda")
export <- psmelt(ps_arthropod)
write.csv(export, file = "output/csv/unfiltered/raw_arthropoda.csv")

#
arth_genus <- summarize_taxa(ps_arthropod, "Genus", "SampleID")
arth_genus <- spread(arth_genus, key="SampleID", value="totalRA")
write.csv(arth_genus, file = "output/csv/unfiltered/all_genglom_unfilt.csv")

#Convert arthropod data to proportions and apply filter threshold
ps_filtered <- transform_sample_counts(ps_arthropod, fun = proportions,
                                       thresh=filt_threshold)
#Drop missing taxa from table
ps_filtered = filter_taxa(ps_filtered, function(x) mean(x) > 0, TRUE)

##Export filtered data
export <- psmelt(ps_filtered)
write.csv(export, file = "output/csv/filtered/all_arthropoda_filt.csv")

#Agglomerate all filtered OTUS by species & Genus level taxonomy and export

#Remove combinatorially indexed samples from dataset
rm_c1 <-  c("Pool-C1-250","Pool-C2-250","Pool-C3-250","Pool-C4-250","Pool-C5-250")
spp_level <- subset_samples(ps_filtered, sample_names(ps_filtered)!=rm_c1)
spp_level = tax_glom(spp_level, "Species", NArm = FALSE)
spp_export <- psmelt(spp_level)
write.csv(spp_export, file = "output/csv/filtered/all_sppglom_filt.csv")

gen_level <- subset_samples(ps_filtered, sample_names(ps_filtered)!=rm_c1)
gen_level = tax_glom(gen_level, "Genus", NArm = FALSE)
locixport <- psmelt(gen_level)
write.csv(locixport, file = "output/csv/filtered/all_genglom_filt.csv")

#Create species level summary table and export
sum_spp_filt <- summarize_taxa(ps_filtered, "Species", "SampleID")
sum_spp_filt <- spread(sum_spp_filt, key="SampleID", value="totalRA")
write.csv(sum_spp_filt, file = "output/csv/filtered/all_sppglom_filt_summarized.csv")

sum_gen_filt <- summarize_taxa(ps_filtered, "Genus", "SampleID")
sum_gen_filt <- spread(sum_gen_filt, key="SampleID", value="totalRA")
write.csv(sum_gen_filt, file = "output/csv/filtered/all_genglom_filt_summarized.csv")

##Write out seperate data table for each loci
genes <- unique(psmelt(ps) %>% select(loci))
genes <- as.vector(genes$loci)
genes <- genes[!is.na(genes)]

for (i in 1:length(genes)){
  print(genes[i])
  ps_loci = subset_taxa(ps_filtered, loci %in% genes[i])
  tax_table(ps_loci)  <- tax_table(ps_loci)[,2:7]
  #Summary export
  sp_summary <- summarize_taxa(ps_loci, "Species", "SampleID")
  sp_summary <- spread(sp_summary, key="SampleID", value="totalRA")
  write.csv(sp_summary, file = paste0("output/csv/seperateloci/",
                                    genes[i],"_sppglom_filt_summarized.csv"))

  gen_summary <- summarize_taxa(ps_loci, "Genus", "SampleID")
  gen_summary <- spread(gen_summary, key="SampleID", value="totalRA")
  write.csv(gen_summary, file = paste0("output/csv/seperateloci/",
                                     genes[i],"_genglom_filt_summarized.csv"))
}
```

Figures

Figure 1 - Bias in mock communities

As part of the mock community analysis, we wish to determine taxonomic bias by looking at observed vs expected reads. To do this, we load dummy sequence, taxonomy, and sample data tables and create a seperate phyloseq object, which will later be merged

This loads a dummy sequence table, taxonomy table, and sample data table and merges it into the existing phyloseq object

```
#get expected abundances
exp_seqtab <- as.matrix(read.csv("sample_data/expected/exp_seqtab.csv",
                                 row.names=1, header=TRUE))
exp_taxtab <- as.matrix(read.csv("sample_data/expected/exp_taxtab.csv",
                                 row.names=1, header=TRUE))
exp_samdf <- read.csv("sample_data/expected/exp_samdf.csv", header=TRUE)

keep.cols <- c("collection_date", "biome", "target_gene", "feature",
"pool_comp" ,"SampleID","experimental_factor")
rownames(exp_samdf) <- exp_samdf$SampleID
exp_samdf <- exp_samdf[rownames(exp_seqtab), keep.cols]

## Make phyloseq and merge
ps_exp <- phyloseq(tax_table(exp_taxtab), sample_data(exp_samdf),
               otu_table(exp_seqtab, taxa_are_rows = FALSE))

rm_c1 <-  c("Pool-C1-250","Pool-C2-250","Pool-C3-250",
            "Pool-C4-250","Pool-C5-250")
rm_c1_exp<- c("Pool-C1-250-exp","Pool-C2-250-exp","Pool-C3-250-exp",
              "Pool-C4-250-exp","Pool-C5-250-exp")

## Plot Fig1a - expected abundances

#Drop Kingdom column to merge results for 3 loci
tax_table(ps_exp) <- tax_table(ps_exp)[,2:7]
ps_exp <- subset_samples(ps_exp, biome == "Laboratory")
ps_exp <- subset_taxa(ps_exp, Phylum == "Arthropoda")
ps_exp <- subset_samples(ps_exp, sample_names(ps_exp)!=rm_c1_exp)
ps_exp = tax_glom(ps_exp, "Species", NArm = TRUE)
```

```
##Subset to mock communities
ps_exp = filter_taxa(ps_exp, function(x) mean(x) > 0, TRUE)
ps_exp <- transform_sample_counts(ps_exp, fun= proportions) # Reset scale to 1
df_exp <- psmelt(ps_exp)

#Reorder to pool composition
df_exp$SampleID <- factor(df_exp$SampleID,
                          levels = unique(df_exp$SampleID[order(-df_exp$pool_comp)]))

Fig1a <- ggplot(df_exp, aes(x= SampleID, y=Abundance,fill= Genus)) +
  geom_bar(stat = "identity", position = "stack", color = "NA")   +
  theme_pubclean() +
    theme(axis.text.x = element_text(angle = -90, hjust = 0),
        plot.title=element_text(hjust = 0.5)) +
  ggtitle(paste0("Expected")) +
  scale_fill_manual(values=c("#0c4687","#ae0707","#fa6e24","#3a9e82","#95cf77")) +
  coord_flip()

## Plot Fig1b - all 3 loci merged

psmock <- ps_filtered

#Drop Kingdom column to merge results for 3 loci
tax_table(psmock) <- tax_table(psmock)[,2:7]

#Subset to mock communities
psmock <- subset_samples(psmock, biome == "Laboratory")
psmock <- subset_taxa(psmock, Phylum == "Arthropoda")
rm_c1 <-  c("Pool-C1-250","Pool-C2-250","Pool-C3-250","Pool-C4-250","Pool-C5-250")
psmock <- subset_samples(psmock, sample_names(psmock)!=rm_c1)
psmock = tax_glom(psmock, "Species", NArm = TRUE)

psmock = filter_taxa(psmock, function(x) mean(x) > 0, TRUE)

# Reset scale to 1 following NArm
psmock <- transform_sample_counts(psmock, fun= proportions)
df_mock <- psmelt(psmock)

#Reorder to pool composition
df_mock$SampleID <- factor(df_mock$SampleID,
                           levels = unique(df_mock$SampleID[order(-df_mock$pool_comp)]))

Fig1b <- ggplot(df_mock, aes(x= SampleID, y=Abundance,fill= Genus)) +
  geom_bar(stat = "identity", position = "stack", color = "NA")   +
  theme_pubclean() +
    theme(axis.text.x = element_text(angle = -90, hjust = 0),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank(),
        plot.title=element_text(hjust = 0.5),
        legend.position = "none") +
  ggtitle(paste0("3 loci")) +
  scale_fill_manual(values=c("#0c4687","#ae0707","#fa6e24","#3a9e82","#95cf77")) +
  coord_flip()

## Plot Fig1c - COI data only
ps_coi <- subset_taxa(ps_filtered, loci == "COI-Eukaryota")
ps_coi <- subset_taxa(ps_coi, Phylum == "Arthropoda")
ps_coi = subset_samples(ps_coi, biome == "Laboratory")
ps_coi = tax_glom(ps_coi, "Species", NArm = TRUE)
tax_table(ps_coi) <- tax_table(ps_coi)[,2:7]

##Transform data to proportions
psra_coi <- transform_sample_counts(ps_coi, fun = proportions)
psra_coi = filter_taxa(psra_coi, function(x) mean(x) > 0, TRUE)

#Remove combinatorial indexed samples
psra_coi <- subset_samples(psra_coi, sample_names(psra_coi)!=rm_c1)
psra_coi <- subset_samples(psra_coi, sample_names(psmock)!=rm_c1_exp)
df_coi <- psmelt(psra_coi)

#Reorder to pool composition
df_coi$SampleID <- factor(df_coi$SampleID,
                          levels = unique(df_coi$SampleID[order(-df_coi$pool_comp)]))

Fig1c <- ggplot(df_coi, aes(x= SampleID, y=Abundance,fill= Genus)) +
  geom_bar(stat = "identity", position = "stack", color = "NA")  +
  theme_pubclean() +
    theme(axis.text.x = element_text(angle = -90, hjust = 0),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank(),
        plot.title=element_text(hjust = 0.5),
        legend.position = "none") +
  ggtitle(paste0("COI")) +
  scale_fill_manual(values=c("#0c4687","#ae0707","#fa6e24","#3a9e82","#95cf77")) +
  coord_flip()

## Plot Fig1d - 18S data only
ps_18s <- subset_taxa(ps_filtered, loci == "18s-Eukaryota")
ps_18s <- subset_taxa(ps_18s, Phylum == "Arthropoda")
ps_18s = subset_samples(ps_18s, biome == "Laboratory")
ps_18s = tax_glom(ps_18s, "Species", NArm = TRUE)
tax_table(ps_18s) <- tax_table(ps_18s)[,2:7]

##Transform data to proportions
psra_18s <- transform_sample_counts(ps_18s, fun = proportions)
psra_18s = filter_taxa(psra_18s, function(x) mean(x) > 0, TRUE)

#Remove combinatorial indexed samples
psra_18s <- subset_samples(psra_18s, sample_names(psra_18s)!=rm_c1)
psra_18s <- subset_samples(psra_18s, sample_names(psmock)!=rm_c1_exp)
df_18s <- psmelt(psra_18s)
```

```
#Reorder to pool composition
df_18s$SampleID <- factor(df_18s$SampleID,
                          levels = unique(df_18s$SampleID[order(-df_18s$pool_comp)]))

Fig1d <- ggplot(df_18s, aes(x= SampleID, y=Abundance,fill= Genus)) +
  geom_bar(stat = "identity", position = "stack", color = "NA")  +
  theme_pubclean() +
    theme(axis.text.x = element_text(angle = -90, hjust = 0),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank(),
        plot.title=element_text(hjust = 0.5),
        legend.position = "none") +
  ggtitle(paste0("18S")) +
  scale_fill_manual(values=c("#0c4687","#ae0707","#fa6e24","#3a9e82","#95cf77")) +
  coord_flip()

## Plot Fig1e - 12S data only

ps_12s <- subset_taxa(ps_filtered, loci == "12S-Eukaryota")
ps_12s <- subset_taxa(ps_12s, Phylum == "Arthropoda")
ps_12s = subset_samples(ps_12s, biome == "Laboratory")
ps_12s = tax_glom(ps_12s, "Species", NArm = TRUE)
tax_table(ps_12s) <- tax_table(ps_12s)[,2:7]

##Transform data to proportions
psra_12s <- transform_sample_counts(ps_12s, fun = proportions)
psra_12s = filter_taxa(psra_12s, function(x) mean(x) > 0, TRUE)

#Remove combinatorial indexed samples
psra_12s <- subset_samples(psra_12s, sample_names(psra_12s)!=rm_c1)
psra_12s <- subset_samples(psra_12s, sample_names(psmock)!=rm_c1_exp)
df_12s <- psmelt(psra_12s)

#Reorder to pool composition
df_12s$SampleID <- factor(df_12s$SampleID,
                          levels = unique(df_12s$SampleID[order(-df_12s$pool_comp)]))

Fig1e <- ggplot(df_12s, aes(x= SampleID, y=Abundance,fill= Genus)) +
  geom_bar(stat = "identity", position = "stack", color = "NA")  +
  theme_pubclean() +
    theme(axis.text.x = element_text(angle = -90, hjust = 0),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank(),
        plot.title=element_text(hjust = 0.5),
        legend.position = "none") +
  ggtitle(paste0("12s")) +
  scale_fill_manual(values=c("#0c4687","#ae0707","#fa6e24","#3a9e82","#95cf77")) +
  coord_flip()
```

```
#Create final figure 1 by stitching all the subfigures together using patchwork
Fig1 <- Fig1a + Fig1b + Fig1c + Fig1d + Fig1e + plot_layout(ncol = 5)

plot(Fig1)
```

Figure 2 - Detection heatmap

```
#Manually set plotting positions
positions = c('Pool-01-100', 'Pool-02-100', 'Pool-03-100',
              'Pool-04-100', 'Pool-05-100', 'Pool-U1-250',
              'Pool-U2-250', 'Pool-U3-250', 'Pool-U4-250',
              'Pool-U5-250', 'Pool-06-500', 'Pool-07-500',
              'Pool-08-500', 'Pool-09-500', 'Pool-10-500',
              'Pool-11-1000', 'Pool-12-1000', 'Pool-13-1000',
              'Pool-14-1000', 'Pool-15-1000', 'Trap-01',
              'Trap-02', 'Trap-03', 'Trap-04',
              'Trap-05', 'Trap-06', 'Trap-07',
              'Trap-08', 'Trap-09', 'Trap-10')

# Manually set labels
labels = c('100 Pool 1', '100 Pool 2', '100 Pool 3',
           '100 Pool 4', '100 Pool 5', '250 Pool 1',
           '250 Pool 2', '250 Pool 3', '250 Pool 4',
           '250 Pool 5', '500 Pool 1', '500 Pool 2',
           '500 Pool 3', '500 Pool 4', '500 Pool 5',
           '1000 Pool 1', '1000 Pool 2', '1000 Pool 3',
           '1000 Pool 4', '1000 Pool 5', 'Trap 1',
           'Trap 2', 'Trap 3', 'Trap 4',
           'Trap 5', 'Trap 6', 'Trap 7',
           'Trap 8', 'Trap 9', 'Trap 10')

mdt <- fast_melt(ps_filtered) %>%
  filter(!is.na(Species)) %>%
  mutate(type = SampleID %>%
           str_split_fixed(pattern="-",n=2) %>%
           as_tibble() %>%
           pull(V1) %>%
           str_replace(pattern="Pool","Mock Community")%>%
           str_replace(pattern="Trap","Trap Community"))  %>%
  filter(!str_detect(SampleID,pattern="Pool-C"))

#Summarise occurance
occurance <- mdt %>%
  group_by(Species) %>%
  summarise(n()) %>%
  rename(occurance = `n()`)

#summarise haplotypes and loci

haplo <- mdt %>%
  mutate(haplo = RelativeAbundance)
```

```r
haplo$haplo[haplo$haplo > 0] <- 1

haplo <- haplo %>%
    select(-SampleID,-RelativeAbundance,-count,-type) %>%
    unique()  %>%
    group_by(Species,loci)   %>%
    summarise(n= n())

colnames(haplo) <- c("Species","loci","haplotypes")

mdt <- mdt %>%
  left_join(occurance, by="Species") %>%
  left_join(haplo, by=c("Species","loci")) %>%
  arrange(-occurance,desc(Species)) %>%
  mutate_at(vars(Species), funs(factor(., levels=unique(.))))


Fig2a <- ggplot(mdt, aes(x=SampleID,y=Species)) +
                       geom_tile(aes(fill=RelativeAbundance)) +
  scale_x_discrete(limits = positions, labels=labels,expand = c(0, 0)) +
  theme_bw() +
  scale_y_discrete(expand = c(0, 0)) +
  theme(axis.text.x = element_text(angle=60, hjust=1),
        axis.title.x=element_blank(),
        axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        strip.background.y = element_blank(),
        strip.text.y = element_blank(),
        panel.grid.major = element_line(colour = "white"),
        panel.spacing = unit(0.2, "lines"),
        legend.position = "bottom",
        legend.direction = "horizontal") +
  facet_grid(Order~.,drop=TRUE, space="free",scales="free")+
    scale_fill_distiller(palette="Blues", direction= 1,
                         trans = 'log10', na.value = "white")


#Make text labels

ann_text <- data.frame(SampleID = 'Pool-01-100', Species = 1 ,lab = mdt$Order,
                       Order = mdt$Order) %>%
             unique()

for (i in 1:length(ann_text$Order)){
  ann_text$Species[i] <- nrow(mdt %>%
                                select(Species,Order) %>%
                                unique() %>%
                                  filter(Order ==  ann_text$Order[i] ))
}

Fig2a <- Fig2a +
  geom_text(data = ann_text,aes(size=Species),
            label = ann_text$lab,colour="#7F7F7F",
            hjust=0,nudge_x=-0.5, show.legend = FALSE) +
  scale_size(range=c(4,10))


Fig2b <- ggplot(mdt, aes(loci, Species)) +
  geom_point(aes(fill = loci,size=haplotypes), shape = 21, color = "black") +
  geom_text(data=mdt[which(mdt$haplotypes>1),],aes(label=haplotypes))+
  scale_fill_manual(values=c("#FC4E07", "#E7B800", "#00AFBB")) +
  theme_bw() +
  theme(axis.text.x = element_text(angle=60, hjust=1),
        axis.title.x= element_blank(),
        axis.ticks.y = element_blank(),
        legend.position = "none",
        legend.direction = "vertical",
        axis.title.y=element_blank(),
        strip.background = element_blank(),
        strip.text = element_blank(),
        panel.grid.major = element_blank(),
        panel.grid.minor = element_blank(),
        panel.border = element_blank(),
        panel.spacing =unit(0.2, "lines"),
        aspect.ratio = .5) +
  scale_size_continuous(range = c(3, 6)) +
  #scale_y_discrete(expand=c(0,0)) +
  facet_grid(Order~., space="free",scales="free" )

#Create multi Heatmap figure
Fig2 <- Fig2b + Fig2a + plot_layout(ncol = 2, widths=c(2,4))
plot(Fig2)
```

Figure 3 - ASV assignment to each rank

Barplots

```
#Get unique ASV's
assign_ranks <- psmelt(ps_filtered)  %>%
  filter(Abundance > 0) %>%
  select(c("OTU","loci","Phylum","Class","Order","Family",
           "Genus","Species","biome")) %>%
  distinct() %>%
  mutate(Species = Species %>%
          replace(str_detect(Species,pattern="spp."),values=NA)) %>%
  gather(key="Rank", value="Taxon", -OTU,-loci, -biome) %>%
  drop_na() %>%
  mutate(method = case_when(!Rank =="Species" ~"RDP",
                  Rank =="Species" ~"Exact Match")) %>%
  mutate(Rank = fct_relevel(Rank,c("Phylum","Class","Order","Family",
                                   "Genus","Species")))%>%
  mutate(method = fct_relevel(method,c("RDP","Exact Match")))

#Plot barchart of ranks
```

```r
gg.ranks <- ggplot(data=assign_ranks, aes(x=Rank, fill=loci, group=loci)) +
  geom_bar(position="dodge",stat="count",alpha=0.8) +
  facet_grid(biome~method, space="free",scales="free_x", drop=TRUE)+
  theme_bw() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust=0)) +
  ylab("Number of ASVs")+
  xlab("Taxonomic rank")+
  scale_fill_manual(values =c("#FC4E07", "#E7B800", "#00AFBB"))
```

Venn diagrams of taxon overlap

```r
df.venn <- data.frame(x = c(0, 0.866, -0.866),
                      y = c(1, -0.5, -0.5),
                      labels = c('18S', 'COI', '12S'))

#Loop through taxonomic levels
vec <- c("Phylum", "Class", "Order", "Family", "Genus", "Species")

venn_list <- vector(mode="list", length=length(vec))

for (i in 1:length(venn_list)){
#Get overlapping features
  level = vec[i]

  features <- mdt %>%
    select(loci,level) %>%
    group_by(loci)%>%
    unique() %>%
    set_names(c("loci","level"))

  spp <- as.character(unique(features$level))
  spp <- spp[which(!str_detect(spp,"spp."))]
  matching <- paste0(features$loci,"_",features$level)

  overlap <- matrix(ncol=length(unique(features$loci)), nrow=length(spp))

  for (l in 1:length(spp)){
    overlap[l,1] <- paste0("12S-Eukaryota","_",spp[l]) %in% matching
    overlap[l,2] <- paste0("COI-Eukaryota","_",spp[l]) %in% matching
    overlap[l,3] <- paste0("18s-Eukaryota","_",spp[l]) %in% matching
  }

  vdc <- vennCounts(overlap)
  class(vdc) <- 'matrix'
  df.vdc <- as.data.frame(vdc)[-1,] %>%
    mutate(x = c(0, 1.2, 0.8, -1.2, -0.8, 0, 0),
           y = c(1.2, -0.6, 0.5, -0.6, 0.5, -1, 0))


  venn_list[[i]] <- ggplot(df.venn) +
                geom_circle(aes(x0 = x, y0 = y, r = 1.5, fill = labels),
                            alpha = .4, size = 1, colour = 'black') +
                  coord_fixed() +
                  theme_void() +
                  theme(legend.position = 'none',
                        plot.title = element_text(hjust = 0.5)) +
                  scale_fill_manual(values =c("#FC4E07", "#E7B800", "#00AFBB")) +
                  labs(title= vec[i]) +
                  annotate("text", x = df.vdc$x, y = df.vdc$y,
                           label = df.vdc$Counts, size = 5)

}

FigVenn <-  venn_list[[4]] +  venn_list[[5]] +
  venn_list[[6]] + patchwork::plot_layout(nrow=1)

plot(FigVenn)
```

Reference database summary

```
ref <- read.FASTA("reference/merged_arthropoda_rdp_genus.fa")
ref_spp <- read.FASTA("reference/merged_arthropoda_rdp_species.fa")

names <- names(ref) %>%
  str_split_fixed(";", n=6) %>%
  as_tibble() %>%
  set_colnames(c("loci", "Phylum", "Class", "Order", "Family", "Genus")) %>%
  mutate(Species = names(ref_spp) %>% # Add species column
           str_split_fixed(pattern=" ", n=3) %>%
           as_tibble() %>%
           unite("Species", c("V2", "V3"), sep="_") %>%
           pull(Species)) %>%
  mutate(loci = str_replace(loci, pattern="-Eukaryota", replacement=""))  %>%
  unique() %>% #get unique species
  mutate(Order = case_when(
    Class =="Insecta" ~ Order,
    !Class =="Insecta" ~ Class
  ))  %>%  group_by(loci, Phylum, Class, Order) %>%
  summarise(Value = n()) %>%
  mutate(pathString = paste("Euk", Phylum, Class, Order, sep="/"))

#Transform to tree, via newick file
tree <- read.tree(textConnection(ToNewick(as.Node(names))))

#Plot tree
p <- ggtree(tree, branch.length = "none") + geom_tiplab() + geom_nodelab(geom='label') +
    scale_x_continuous(expand=c(0, 2))

bar <- names %>%
  ungroup() %>%
  rename(id = Order) %>%
  select(id, loci, Value)
p2 <- facet_plot(p, 'Unique Species', data = bar, geom=ggstance::geom_barh,
                 mapping=aes(x=Value, fill=loci), stat="identity") +
                theme_bw() +
                scale_fill_manual(values =c("#FC4E07", "#E7B800", "#00AFBB"))
```

Supplementary Figure - Index Switching

```
switchplot <- switched
replacement <- exp_rate$Sample_Name

i=1
for (i in 1:length(replacement))
{
  switchplot$i7 <- as.character(switchplot$i7) %>%
  str_replace(pattern= as.character(exp_rate$i7[i]),
              replacement=paste0(replacement[i],"-",as.character(exp_rate$i7[i])))

  switchplot$i5 <- as.character(switchplot$i5) %>%
    str_replace(pattern= as.character(exp_rate$i5[i]),
                replacement=paste0(replacement[i],"-",as.character(exp_rate$i5[i])))
}

#Manually set ordering for plot
orderi7 <- c("Pool_1_100-GACGAGAT", "Pool_2_100-TAGTGGCA",
             "Pool_3_100-CATTAACG", "Pool_4_100-TCGTTGAA",
             "Pool_5_100-TAGTACGC", "Pool_6_500-TTCACCGT",
             "Pool_7_500-AGGACAGT", "Pool_8_500-AATCGTGG",
             "Pool_9_500-TGAATGCC",  "Pool_10_500-GTGCAATG",
             "Pool_11_1000-AGTGGCAT",  "Pool_12_1000-AGTCTACC",
             "Pool_13_1000-ATCGGTAG",  "Pool_14_1000-CGTATGAT",
             "Pool_15_1000-CTGTCGTA")

orderi5 <- c("Pool_1_100-GACTTCGT", "Pool_2_100-AATCTCGT",
             "Pool_3_100-TTGCCACT", "Pool_4_100-GCGTTAAT",
             "Pool_5_100-CTTCAACG", "Pool_6_500-AGCGTACT",
             "Pool_7_500-TACGGTGA", "Pool_8_500-AACTGTCC",
             "Pool_9_500-GACTGATA",  "Pool_10_500-ACATCTGC",
             "Pool_11_1000-ACGTTAGG",  "Pool_12_1000-CACTAGAC",
             "Pool_13_1000-TGGCATTC",  "Pool_14_1000-ACATTGCA",
             "Pool_15_1000-TATGCCAC")


switchplot$i7<- factor(switchplot$i7, levels = orderi7)
switchplot$i5 <- factor(switchplot$i5 , levels = rev(orderi5))

switchplot <- switchplot %>%
  drop_na()

FigS1 <- ggplot(data = switchplot, aes(x = i7, y = i5), stat="identity") +
  geom_tile(aes(fill = count),alpha=0.8)  +
  scale_fill_viridis(name = "reads", begin=0.1,trans = "log10")  +
  geom_text(label=switchplot$count) +
  theme_bw() +
  theme(axis.text.x = element_text(angle=90, hjust=1),
        plot.title=element_text(hjust = 0.5),
        plot.subtitle =element_text(hjust = 0.5),
        legend.position = "none") +
  labs(title= "100-500-1000 Pool Samples",
       subtitle = paste0("Total Reads: ", total_reads, " Switch rate: ",
                         sprintf("%1.2f%%", switch_rate*100)))

plot(FigS1)
```

Supplementary Figure - Edit Distances

```
#Set colour table
library(fields)
colorTable <- designer.colors(8, c( "red","yellow", "white"),
                              x = c(0, 5, 8) / 140)
col_order <- SampleSheet$index

# Levenshtein i7
lvi7 <- as.tibble(stringdistmatrix(SampleSheet$index, SampleSheet$index, "lv"))

#rearrange to be the same order as figs1a
colnames(lvi7) <- paste0(SampleSheet$Sample_Name, "-", SampleSheet$index)
lvi7$row <- paste0(SampleSheet$Sample_Name, "-", SampleSheet$index)
lvi7 <- lvi7 %>%
  gather(key="col","value",-row)


lvi7$row <- factor(lvi7$row, levels = orderi7)
lvi7$col <- factor(lvi7$col, levels = rev(orderi7))

plvi7 <- ggplot(data = lvi7, aes(x = row, y = col), stat="identity")  +
  geom_tile(data = lvi7,aes(fill = as.factor(value))) +
  scale_fill_manual(values = colorTable) +
  geom_text(label=lvi7$value)   +
  theme_bw() +
  theme(plot.title=element_text(hjust = 0.5),
        plot.subtitle =element_text(hjust = 0.5),
        legend.position = "none",
        axis.title=element_blank(),
        axis.text = element_blank(),
        axis.ticks=element_blank()) +
  labs(title= "Edit distance between i7 Indices")

# Levenshtein i5

lvi5 <- as.tibble(stringdistmatrix(SampleSheet$index2, SampleSheet$index2, "lv"))

#rearrange to be the same order as figs1a
colnames(lvi5) <- paste0(SampleSheet$Sample_Name, "-", SampleSheet$index2)
```

```r
lvi5$row <- paste0(SampleSheet$Sample_Name, "-", SampleSheet$index2)

lvi5 <- lvi5 %>%
  gather(key="col","value",-row)

lvi5$row <- factor(lvi5$row, levels = orderi5)
lvi5$col <- factor(lvi5$col, levels = rev(orderi5))

plvi5 <- ggplot(data = lvi5, aes(x = row, y = col), stat="identity")  +
  geom_tile(data = lvi5,aes(fill = as.factor(value))) +
  scale_fill_manual(values = colorTable) +
  geom_text(label=lvi5$value)  +
  theme_bw() +
  theme(plot.title=element_text(hjust = 0.5),
        plot.subtitle =element_text(hjust = 0.5),
        legend.position = "none",
        axis.title=element_blank(),
        axis.text = element_blank(),
        axis.ticks=element_blank()) +
  labs(title= "Edit distance between i5 Indices")

##Plot relationships between distance and switching

distrel <- obs_rate %>%
  filter(i5 %in% SampleSheet$index2) %>%
  filter(i7 %in% SampleSheet$index)

distrel <- distrel[which(stringdist(distrel$i5,SampleSheet$index2,method="lv")>0),]

i5dist <- list()
i7dist <- list()

for (i in seq(2,7,1)){
  value=i
  print(value)
  dfi5 <- as.tibble(distrel$count[which(
    stringdist(distrel$i5,SampleSheet$index2,method="lv")==value)])
  colnames(dfi5) <- value
  dfi7 <- as.tibble(distrel$count[which(
    stringdist(distrel$i7,SampleSheet$index,method="lv")==value)])
  colnames(dfi7) <- value
  i5dist[[i]] <- dfi5
  i7dist[[i]] <- dfi7
}
i5dist <- bind_rows(i5dist) %>%
  gather(dist,count) %>%
  drop_na()

gg.i5dist <- ggplot(data=i5dist, aes(x=dist,y=count,colour=dist)) +
  geom_jitter()+
  scale_colour_manual(values = colorTable) +
  theme_bw() +
  xlab("Edit distance") +
  ylab("Read count") +
  theme(legend.position = "none")+
    stat_summary(fun.y = mean, geom = "errorbar", aes(ymax = ..y.., ymin = ..y..),
                 width = .75, linetype = "dashed",colour="black")

i7dist <- bind_rows(i7dist) %>%
  gather(dist,count) %>%
  drop_na()

gg.i7dist <- ggplot(data=i7dist, aes(x=dist,y=count,colour=dist)) +
  geom_jitter()+
  scale_colour_manual(values = colorTable) +
  theme_bw() +
  xlab("Edit distance") +
  ylab("Read count") +
  theme(legend.position = "none")+
    stat_summary(fun.y = mean, geom = "errorbar", aes(ymax = ..y.., ymin = ..y..),
                 width = .75, linetype = "dashed",colour="black")

FigS2 <- plvi5 + plvi7 + gg.i5dist  + gg.i7dist

plot(FigS2)
```

Nucleotide distance statistics for potential exotic detections

```
genus <- read.dna("reference/merged_arthropoda_rdp_genus.fa", format="fasta")
spp <- read.dna("reference/merged_arthropoda_rdp_species.fa", format="fasta")

#Aphis varians

n_aphis <- names(genus) %>%
  str_split_fixed(pattern=";",n=6) %>%
  as_tibble() %>%
  filter(V1=="COI-Eukaryota") %>%
  filter(V6=="Aphis")
print(paste0(nrow(aphis)," ",unique(aphis$V1),
             " sequences in database for ", unique(aphis$V6)))

#Carpophilus sexpustulatus

carp <- names(genus) %>%
  str_split_fixed(pattern=";",n=6) %>%
  as_tibble() %>%
  filter(V1=="18s-Eukaryota") %>%
  filter(V6=="Carpophilus") %>%
  unite(col=all, c("V1","V2","V3","V4","V5","V6"),sep=";")
print(paste0(nrow(carp)," ",unique(carp$V1),
             " sequences in database for ", unique(carp$V6)))

carpseqs <- genus[which(names(genus) %in% carp$all)]
```

```r
carpspp <- spp[which(spp %in% carpseqs)]
```

Session info

```r
sessionInfo()
```

```
## R version 3.5.3 (2019-03-11)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_Australia.1252  LC_CTYPE=English_Australia.1252
## [3] LC_MONETARY=English_Australia.1252 LC_NUMERIC=C
## [5] LC_TIME=English_Australia.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] knitr_1.24
##
## loaded via a namespace (and not attached):
##  [1] compiler_3.5.3  magrittr_1.5    tools_3.5.3     htmltools_0.3.6
##  [5] yaml_2.2.0      Rcpp_1.0.2      stringi_1.4.3   rmarkdown_1.15
##  [9] stringr_1.4.0   xfun_0.9        digest_0.6.20   evaluate_0.14
```