Supplementary to: Developing a non-destructive metabarcoding protocol for detection of pest insects in bulk trap catches

Reproducible workflow

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

2019/11/28

Introduction

This is the R based reproducible workflow that performed the metabarcoding analyses presented in 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 colony reared Aphid and Psyllid species, and 10 trap samples from a potato field. The amplicon libraries were prepared in three batches: (1) replicated sets of the 5x 250 pool mock communities to compare combinatorial and unique dual indexing strategies; Run_1 - 15 pools of 100, 500, and 1000 insects; Run_3 - Ten field trap samples with varying numbers of insects. Amplicons were generated using multiplex PCR in which all three target genes were amplified in a single reaction per sample using the three sets of metabarcoding primers

In this analysis pipeline, each MiSeq run is processed processed within a for-loop, so parameters are kept consistent but error estimation is conducted separately by run

Remove primers

DADA2 requires Non-biological nucleotides i.e. primers, adapters, linkers, etc to be removed. In this study there were 3 amplicons of different size amplified 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 and adapter sequences in addition to the 5'primer.To remove these we use the Kmer based adapter trimming software BBDuk (Part of BBTools package https://jgi.doe.gov/data-and-tools/bbtools/)

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

```
# BASH
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}')

#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 required packages, and import helper functions from scripts folder

```
# R
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)
```

```
## Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
## logical.return = TRUE, : there is no package called 'ggplot'
```

```
##      dada2   phyloseq     ggplot        ips   DECIPHER data.table  tidyverse
##       TRUE       TRUE      FALSE       TRUE       TRUE       TRUE       TRUE
## Biostrings  ShortRead     scales stringdist  patchwork      psadd     ggpubr
##       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE       TRUE
##     seqinr      limma    ggforce    viridis  data.tree     ggtree
##       TRUE       TRUE       TRUE       TRUE       TRUE       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 have slightly worse quality at the end, which is common in Illumina sequencing. Informed by these profiles, we will filter reads with more than 2 expected errors or ambiguous bases, truncate the reads cut the reads at any point the Q score crashes below 2, and maintain only filtered reads longer than 100bp.

```
# 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(2,2), truncQ=2, maxN = 2, minLen = 100,
    rm.phix=TRUE, compress=TRUE, verbose=TRUE)
    )
}

print(filtered_out)
```

Post filtering error plotting

We will plot the read quality as a 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 data set 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 "Amplicon sequence variants" (ASVs) 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 paired reverse reads, and then constructing the merged "contig" sequences from those with at least 20bp overlap. Following this step, a sequence variant table is constructed for each run and saved as an RDS file.

```
# R
runs <- dir("data/", pattern="run")
set.seed(100) # Set a random seed

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 & save sequence table for each run
seqtab<- makeSequenceTable(mergers)
saveRDS(seqtab, paste0(path,"/seqtab.rds"))
}
```

Merge Runs, Remove Chimeras

Following independent inference of sequences 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.

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

# Read in seqtabs
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))
}

# Merge and filter
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"))

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

Assign Taxonomy to sequence variants

Taxonomy was assigned to the lowest rank possible with a minimum bootstrap support of 80% using the RDP Naive Bayes classifier as implemented in the DADA2 R package, followed by species level assignment using exact matching between the query and reference sequences.

```
# R
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.gz",
                      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.gz",
                       allowMultiple=TRUE)

# Add spp. to species rank for those with only a genus rank assignment
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=" ")

# 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. All later filtering and plotting is conducted on this object.

```
# R
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, remove duplicates and subset
samdf <- read.csv("sample_data/Sample_info.csv", header=TRUE) %>%
  filter(!duplicated(SampleID)) %>%
  magrittr::set_rownames(.$SampleID) %>%
  select("collection_date", "biome", "target_gene", "feature",
"pool_comp" ,"SampleID","experimental_factor")

# Create phlyoseq object
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. 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?FFFBABAEGGGGGGGGGGGGGFHHHHHHHGHHHGHHHHHHHHHHHHHGGEEEEEAFGGHGHFAHHHHGGGH
```

```
# 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 arising 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

```
# R
# Read in original sample sheet
SampleSheet <- read_csv("demultiplexing/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("demultiplexing/determined.txt", col_names = FALSE) %>%
  magrittr::set_colnames(c("Sample_Name","count")) %>%
  mutate(Sample_Name = Sample_Name %>%
    str_replace_all("-","_") %>%
    str_split_fixed("_S",n=2) %>%
    as_tibble() %>%
    pull(V1)) %>%
  left_join(SampleSheet, by="Sample_Name") %>%
  tidyr::unite(indices, index, index2, sep="+") %>%
  select(Sample_Name, count, indices)

# Undetermined reads from mock communities
undetermined <- read_table("demultiplexing/undetermined.txt", col_names = FALSE) %>%
  magrittr::set_colnames(c("count","indices")) %>%
  mutate(Sample_Name = "Undetermined_S0_R1_001.trimmed.fastq.gz")

# Join together
```

```
indices <- rbind(determined, undetermined)

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

# Get unused combinations resulting from index switching
switched <- left_join(combos,indices,by="indices") %>%
  magrittr::set_colnames(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

From the above we determined a filtering threshold of 0.00011664. We now apply this to the data and 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
transform_sample_counts(ps, fun = proportions) %>%
  summarize_taxa("loci", "SampleID") %>%
  spread(key="SampleID", value="totalRA") %>%
  write.csv(file = "output/csv/unfiltered/loci_summarized.csv")

# Subset data to Athropoda only & Export unfiltered CSV
subset_taxa(ps, Phylum == "Arthropoda") %>%
  psmelt() %>%
  write.csv(file = "output/csv/unfiltered/raw_arthropoda.csv")

# Output Arthropoda summarised at genus level
```

```r
subset_taxa(ps, Phylum == "Arthropoda") %>%
  summarize_taxa("Genus", "SampleID") %>%
  spread(key="SampleID", value="totalRA") %>%
  write.csv(file = "output/csv/unfiltered/all_genglom_unfilt.csv")

# Convert arthropod data to proportions and apply filter threshold
ps_filtered <- subset_taxa(ps, Phylum == "Arthropoda") %>%
  transform_sample_counts(fun = proportions, thresh=filt_threshold) %>%
  filter_taxa(function(x) mean(x) > 0, TRUE)

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

#Remove combinatorially indexed samples from dataset for export
rm_c1 <-  c("Pool-C1-250","Pool-C2-250","Pool-C3-250","Pool-C4-250","Pool-C5-250")

# Export raw species level
subset_samples(ps_filtered, sample_names(ps_filtered) !=rm_c1) %>%
  tax_glom("Species", NArm = FALSE) %>%
  psmelt() %>%
  write.csv(file = "output/csv/filtered/all_sppglom_filt.csv")

# Export raw genus level
subset_samples(ps_filtered, sample_names(ps_filtered)!=rm_c1) %>%
  tax_glom("Genus", NArm = FALSE) %>%
  psmelt() %>%
  write.csv(file = "output/csv/filtered/all_genglom_filt.csv")

# Export summary Species level
summarize_taxa(ps_filtered, "Species", "SampleID") %>%
  spread(key="SampleID", value="totalRA") %>%
  write.csv(file = "output/csv/filtered/all_sppglom_filt_summarized.csv")

# Export summary genus level
summarize_taxa(ps_filtered, "Genus", "SampleID") %>%
  spread(key="SampleID", value="totalRA") %>%
  write.csv(file = "output/csv/filtered/all_genglom_filt_summarized.csv")

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

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]

  # Rescale to RA following subset
  ps_loci <- transform_sample_counts(ps_loci, fun = proportions)
```

```
  # Export summary
  summarize_taxa(ps_loci, "Species", "SampleID") %>%
    spread(key="SampleID", value="totalRA") %>%
    write.csv(file = paste0("output/csv/seperateloci/",
                            genes[i], "_sppglom_filt_summarized.csv"))

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

Figures

Figure 2 - Reference DB and ASV's assigned

This figure will summarise the taxonomic composition of the reference database, Number of amplicon sequence variants (ASVs) from mock communities and field trap samples successfully assigned to taxonomic ranks for each locus, and the taxonomic overlap between loci.

Reference database summary

```
# R
# Load reference files
ref <- read.FASTA("reference/merged_arthropoda_rdp_genus.fa.gz")
ref_spp <- read.FASTA("reference/merged_arthropoda_rdp_species.fa.gz")

# Get lineage for tree
lineage <- 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(lineage))))
```

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

# Get data for bar plot
bar <- lineage %>%
  ungroup() %>%
  rename(id = Order) %>%
  select(id, loci, Value)

# Plot tree + metadata
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"))
```

ASV assignment to each rank

```
# R
# 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
gg.ranks <- ggplot(data=assign_ranks, aes(x=Rank, fill=loci, group=loci)) +
  geom_bar(position="dodge", stat="count", alpha=0.8) +
  geom_text(stat='count', aes(label=..count..),
            position = position_dodge(width = 1),
            vjust=0.5, hjust=1.2, angle=90) +
  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 <- joint %>%
    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)
}
```

```r
# Create multifigure
FigVenn <-  venn_list[[4]] +  venn_list[[5]] +
  venn_list[[6]] + patchwork::plot_layout(nrow=1)

plot(FigVenn)
```

Figure 3 - Bias in mock communities

As part of the mock community analysis, we wish to determine taxonomic bias by looking at expected and observed relative abundance of each species in each mock community sample. To do this, we load dummy sequence, taxonomy, and sample data tables and create a separate phyloseq object, which we then merge with the real phyloseq sample.

```r
# R
# 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) %>%
  select("collection_date", "biome", "target_gene", "feature", "pool_comp" ,"SampleID","experimental_fa
  magrittr::set_rownames(.$SampleID)

#  Make expected 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]

# Subset to mock communties
ps_exp <- subset_samples(ps_exp, biome == "Laboratory") %>%
  subset_taxa(Phylum == "Arthropoda") %>%
  subset_samples(sample_names(ps_exp)!=rm_c1_exp) %>%
  tax_glom("Species", NArm = TRUE) %>%
  filter_taxa(function(x) mean(x) > 0, TRUE) %>%
  transform_sample_counts(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
df_mock <- subset_samples(psmock, biome == "Laboratory") %>%
  subset_samples(sample_names(psmock)!=rm_c1) %>%
  tax_glom("Species", NArm = TRUE) %>%
  filter_taxa(function(x) mean(x) > 0, TRUE) %>%
  transform_sample_counts(fun= proportions)  %>%
  psmelt()

# 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") %>%
  subset_samples(biome == "Laboratory") %>%
  tax_glom("Species", NArm = TRUE) %>%
  transform_sample_counts(fun = proportions) %>%
  filter_taxa(function(x) mean(x) > 0, TRUE) %>%
  subset_samples(sample_names(ps_coi)!=rm_c1) %>%
  subset_samples(sample_names(ps_coi)!=rm_c1_exp)

tax_table(ps_coi) <- tax_table(ps_coi)[,2:7]
df_coi <- psmelt(ps_coi)

# Reorder to pool composition
```

```r
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") %>%
  subset_samples(biome == "Laboratory") %>%
  tax_glom("Species", NArm = TRUE) %>%
  transform_sample_counts(fun = proportions) %>%
  filter_taxa(function(x) mean(x) > 0, TRUE) %>%
  subset_samples(sample_names(ps_18s)!=rm_c1) %>%
  subset_samples(sample_names(ps_18s)!=rm_c1_exp)

tax_table(ps_18s) <- tax_table(ps_18s)[,2:7]
df_18s <- psmelt(ps_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") %>%
  subset_samples(biome == "Laboratory") %>%
  tax_glom("Species", NArm = TRUE) %>%
  transform_sample_counts(fun = proportions) %>%
  filter_taxa(function(x) mean(x) > 0, TRUE) %>%
```

```
  subset_samples(sample_names(ps_12s)!=rm_c1) %>%
  subset_samples(sample_names(ps_12s)!=rm_c1_exp)

tax_table(ps_12s) <- tax_table(ps_12s)[,2:7]
df_12s <- psmelt(ps_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 5 - Detection heatmap

Figure 5 of the manuscript is a Heat map displaying the relative abundance of different Arthropoda taxa in both mock community and field trapped insect sample. We will also display as a side panel The loci contributing to detection of each taxa as well as the number of amplicon sequence variant haplotypes for that taxon.

```
# R
# Manually set axis 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')

# Make a data frame
sumdt <- 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 taxon occurance
occurance <- sumdt %>%
  group_by(Species) %>%
  summarise(n()) %>%
  rename(occurance = `n()`)

# Summarise haplotypes by taxon and loci
haplo <- sumdt %>%
  mutate(haplo = RelativeAbundance) %>%
  mutate(haplo = case_when(
    haplo > 0 ~ 1
  )) %>%
    select(-SampleID,-RelativeAbundance, -count, -type) %>%
    unique()  %>%
    group_by(Species, loci) %>%
    summarise(n= n()) %>%
  magrittr::set_colnames(c("Species", "loci", "haplotypes"))

# Join together with raw data
joint <- sumdt %>%
  left_join(occurance, by="Species") %>%
  left_join(haplo, by=c("Species","loci")) %>%
  arrange(-occurance, desc(Species)) %>%
  mutate_at(vars(Species), funs(factor(., levels=unique(.))))


#  Make heatmap
Fig2a <- ggplot(joint, 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 Order labels for inside heatmap
ann_text <- data.frame(SampleID = 'Pool-01-100', Species = 1, lab = joint$Order,
                       Order = joint$Order) %>%
             unique()

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

# Add Order labels inside heatmap
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))


# Make haplotype summary
Fig2b <- ggplot(joint, aes(loci, Species)) +
  geom_point(aes(fill = loci,size=haplotypes), shape = 21, color = "black") +
  geom_text(data=joint[which(joint$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)) +
  facet_grid(Order~., space="free", scales="free" )

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

plot(Fig2)
```

Supplementary figures

Supplementary Figure 1 - Index Switching

To summarise how the index switching measurement used as a filtering threshold was achieved, we will make a heat map displaying the number of reads demultiplexed in all possible combinations, taking into account switching of either i5 and i7 indexes.

```
# R

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 2 - Edit Distances

To explore if index switching was caused by errors in the index reads, we will plot a heatmap of the pairwise edit distances (including substitutions, insertions and deletions), and a scatter plot of the relationship between switched reads and edit distance.

```
# R

# Set colour table for plot
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 axes to be the same order as supplementary fig 1
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))

# Make distance plot of i7
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 axes to be the same order as supplementary fig 1
colnames(lvi5) <- paste0(SampleSheet$Sample_Name, "-", SampleSheet$index2)
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))

# Make distance plot of i5
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()

# Plot i5 switch distance
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()

# Plot i7 switch distance
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")

# Make multiplot using patchwork
FigS2 <- plvi5 + plvi7 + gg.i5dist  + gg.i7dist

plot(FigS2)
```

Session info

```
sessionInfo()
```

```
## R version 3.6.1 (2019-07-05)
```

```
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17763)
##
## 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] stats4    parallel  stats     graphics  grDevices utils     datasets
## [8] methods   base
##
## other attached packages:
##  [1] ggtree_2.0.0                data.tree_0.7.11
##  [3] viridis_0.5.1               viridisLite_0.3.0
##  [5] ggforce_0.3.1               limma_3.42.0
##  [7] seqinr_3.6-1                ggpubr_0.2.4
##  [9] magrittr_1.5                psadd_0.1.2
## [11] patchwork_0.0.1             stringdist_0.9.5.5
## [13] scales_1.0.0                ShortRead_1.44.0
## [15] GenomicAlignments_1.22.0    SummarizedExperiment_1.16.0
## [17] DelayedArray_0.12.0         matrixStats_0.55.0
## [19] Biobase_2.46.0              Rsamtools_2.2.0
## [21] GenomicRanges_1.38.0        GenomeInfoDb_1.22.0
## [23] BiocParallel_1.20.0         forcats_0.4.0
## [25] stringr_1.4.0               dplyr_0.8.3
## [27] purrr_0.3.3                 readr_1.3.1
## [29] tidyr_1.0.0                 tibble_2.1.3
## [31] ggplot2_3.2.1               tidyverse_1.2.1
## [33] data.table_1.12.6           DECIPHER_2.14.0
## [35] RSQLite_2.1.2               Biostrings_2.54.0
## [37] XVector_0.26.0              IRanges_2.20.0
## [39] S4Vectors_0.24.0            BiocGenerics_0.32.0
## [41] ips_0.0.11                  ape_5.3
## [43] phyloseq_1.30.0             dada2_1.14.0
## [45] Rcpp_1.0.2                  knitr_1.26
##
## loaded via a namespace (and not attached):
##  [1] colorspace_1.4-1       ggsignif_0.6.0         hwriter_1.3.2
##  [4] rstudioapi_0.10        farver_1.1.0           bit64_0.9-7
##  [7] lubridate_1.7.4        xml2_1.2.2             codetools_0.2-16
## [10] splines_3.6.1          polyclip_1.10-0        zeallot_0.1.0
## [13] ade4_1.7-13            jsonlite_1.6           broom_0.5.2
## [16] cluster_2.1.0          BiocManager_1.30.10    compiler_3.6.1
## [19] httr_1.4.1             rvcheck_0.1.6          backports_1.1.5
## [22] assertthat_0.2.1       Matrix_1.2-17          lazyeval_0.2.2
## [25] cli_1.1.0              tweenr_1.0.1           htmltools_0.4.0
## [28] tools_3.6.1            igraph_1.2.4.1         gtable_0.3.0
## [31] glue_1.3.1             GenomeInfoDbData_1.2.2 reshape2_1.4.3
## [34] fastmatch_1.1-0        cellranger_1.1.0       vctrs_0.2.0
## [37] multtest_2.42.0        nlme_3.1-141           iterators_1.0.12
## [40] xfun_0.11              rvest_0.3.5            lifecycle_0.1.0
```

```
## [43] phangorn_2.5.5         XML_3.98-1.20          zlibbioc_1.32.0
## [46] MASS_7.3-51.4          hms_0.5.2              biomformat_1.14.0
## [49] rhdf5_2.30.0           RColorBrewer_1.1-2     yaml_2.2.0
## [52] gridExtra_2.3          memoise_1.1.0          latticeExtra_0.6-28
## [55] stringi_1.4.3          plotrix_3.7-6          foreach_1.4.7
## [58] tidytree_0.2.9         permute_0.9-5          rlang_0.4.1
## [61] pkgconfig_2.0.3        bitops_1.0-6           evaluate_0.14
## [64] lattice_0.20-38        Rhdf5lib_1.8.0         treeio_1.10.0
## [67] bit_1.1-14             tidyselect_0.2.5       plyr_1.8.4
## [70] R6_2.4.1               generics_0.0.2         DBI_1.0.0
## [73] pillar_1.4.2           haven_2.1.1            withr_2.1.2
## [76] mgcv_1.8-30            survival_2.44-1.1      RCurl_1.95-4.12
## [79] modelr_0.1.5           crayon_1.3.4           rmarkdown_1.17
## [82] grid_3.6.1             readxl_1.3.1           blob_1.2.0
## [85] vegan_2.5-6            digest_0.6.22          RcppParallel_4.4.4
## [88] munsell_0.5.0          quadprog_1.5-7
```