R Notebook – ITS2 diss.ch3

This is an R Markdown Notebook. When you execute code within the notebook, the results appear beneath the code.

Ch.3 Fungal Analyses

Load Libs

```
n.threads=20
options("Ncpus" = n.threads)

if (!require("devtools", quietly = TRUE))
    install.packages("devtools")
source_url("https://raw.githubusercontent.com/kek12e/my_r_functions/refs/heads/main/my_r_functions.R")
```

```
## i SHA-1 hash of file is "1d8ba72322e9ade98165894fdf934eb8c7f0dbed"
```

```
libs = c(
    "devtools",
    "phyloseq",
        "ggplot2",
        "microViz",
        "ggtext",
        "stringr",
        "colorBlindness",
        "phytools",
        "gplots",
        "viridis",
        "hrbrthemes",
        "vegan",
        "dplyr",
        "decontam"
    )

my_load_libs(libs)
```

```
## >>> Libraries loaded:
##   devtools version 2.4.5
##   phyloseq version 1.50.0
##   ggplot2 version 3.5.1
##   microViz version 0.12.6
##   ggtext version 0.1.2
##   stringr version 1.5.1
##   colorBlindness version 0.1.9
##   phytools version 2.4.4
##   gplots version 3.2.0
##   viridis version 0.6.5
##   hrbrthemes version 0.8.7
##   vegan version 2.6.10
##   dplyr version 1.1.4
##   decontam version 1.26.0
```

Set Variables

```
# directories for dada2 files
work.dir = "./dada2"
pdf.dir = file.path(work.dir,"pdf")
rds.dir = file.path(work.dir,"rds")
csv.dir = file.path(work.dir,"csv")
fna.dir = file.path(work.dir,"fasta")
rdat.dir = file.path(work.dir,"rdata")
tree.dir = file.path(work.dir,"trees")
itsx.dir = file.path(work.dir, "itsx")

# phyloseq object
ps.p = readRDS(file.path(rds.dir,"ps.p.RDS"))
```

Decontam

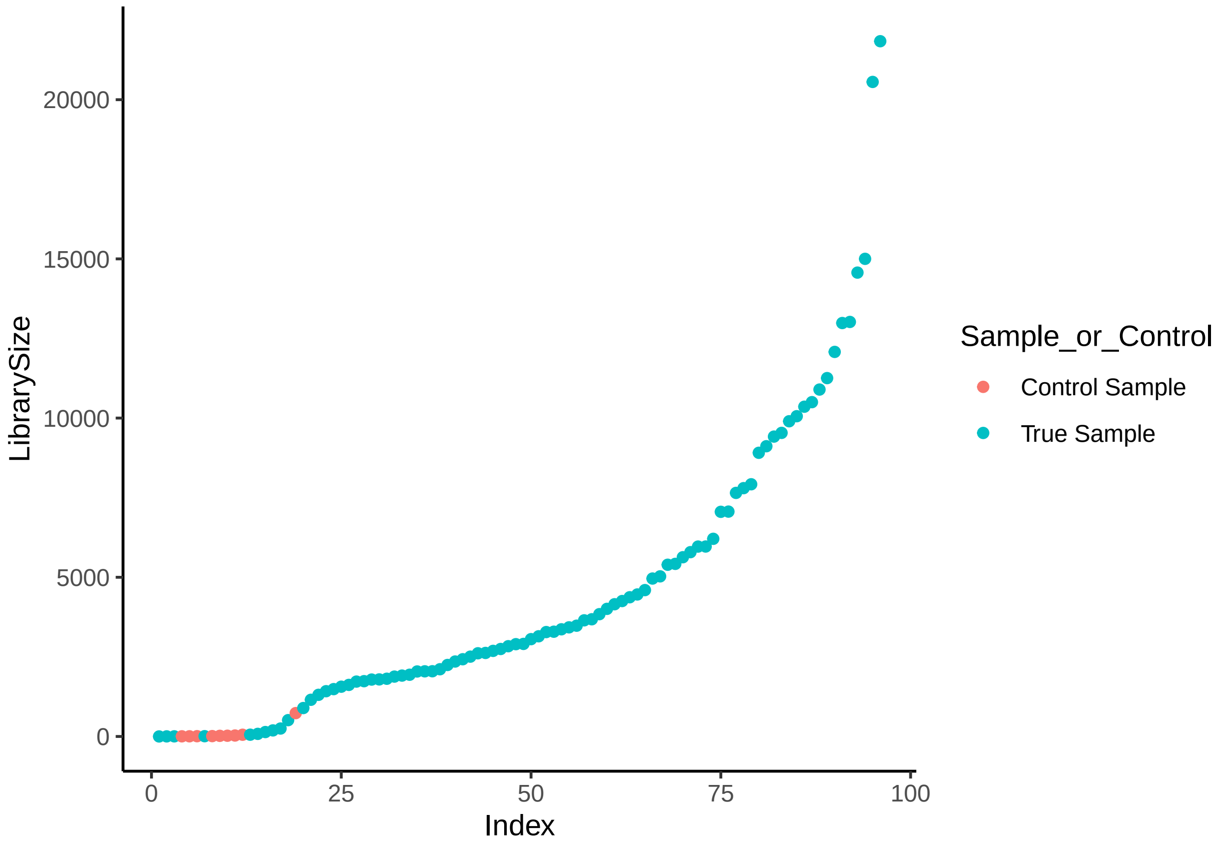
```
samdat =
  data.frame(
    "seq.id" = sample_names(ps.p),
    "Sample_or_Control"="True Sample",
    stringsAsFactors=FALSE
  )
sample_names(ps.p)
```

```
##  [1] "d3e01" "d3e02" "d3e03" "d3e04" "d3e05" "d3e06" "d3e07" "d3e08" "d3e09"
## [10] "d3e10" "d3e13" "d3e14" "d3e15" "d3e16" "d3e17" "d3e18" "d3e19" "d3e20"
## [19] "d3e21" "d3e22" "d3e25" "d3e26" "d3e27" "d3e28" "d3e29" "d3e30" "d3e31"
## [28] "d3e32" "d3e33" "d3e34" "d3e37" "d3e38" "d3e39" "d3e40" "d3e41" "d3e42"
## [37] "d3e43" "d3e44" "d3e45" "d3e46" "NCe12" "NCe24" "NCe36" "NCe48" "NFH20"
## [46] "TI02"  "TI03"  "TI06"  "TI07"  "TINC"  "TIPC"  "TSe01" "TSe02" "TSe03"
## [55] "TSe04" "TSe05" "TSe06" "TSe07" "TSe08" "TSe09" "TSe10" "TSe11" "TSe12"
## [64] "TSe13" "TSe14" "TSe15" "TSe16" "TSe17" "TSe18" "TSe19" "TSe20" "TSe21"
## [73] "TSe22" "TSe23" "TSe24" "TSe25" "TSe26" "TSe27" "TSe28" "TSe29" "TSe30"
## [82] "TSe31" "TSe32" "TSe33" "TSe34" "TSe35" "TSe36" "TSe37" "TSe38" "TSe39"
## [91] "TSe40" "TSe41" "TSe42" "TSNC1" "TSNC2" "TSNC3"
```

```
controls.i = grep("NFW|NC|NFH2",samdat$seq.id)
controls.sn = sample_names(ps.p)[controls.i]
samdat$Sample_or_Control[controls.i] = "Control Sample"
rownames(samdat) = samdat$seq.id

sample_data(ps.p) = samdat

sample_data(ps.p)$is.neg =
  sample_data(ps.p)$Sample_or_Control == "Control Sample"
```

Plot Lib Size vs. True or Contam

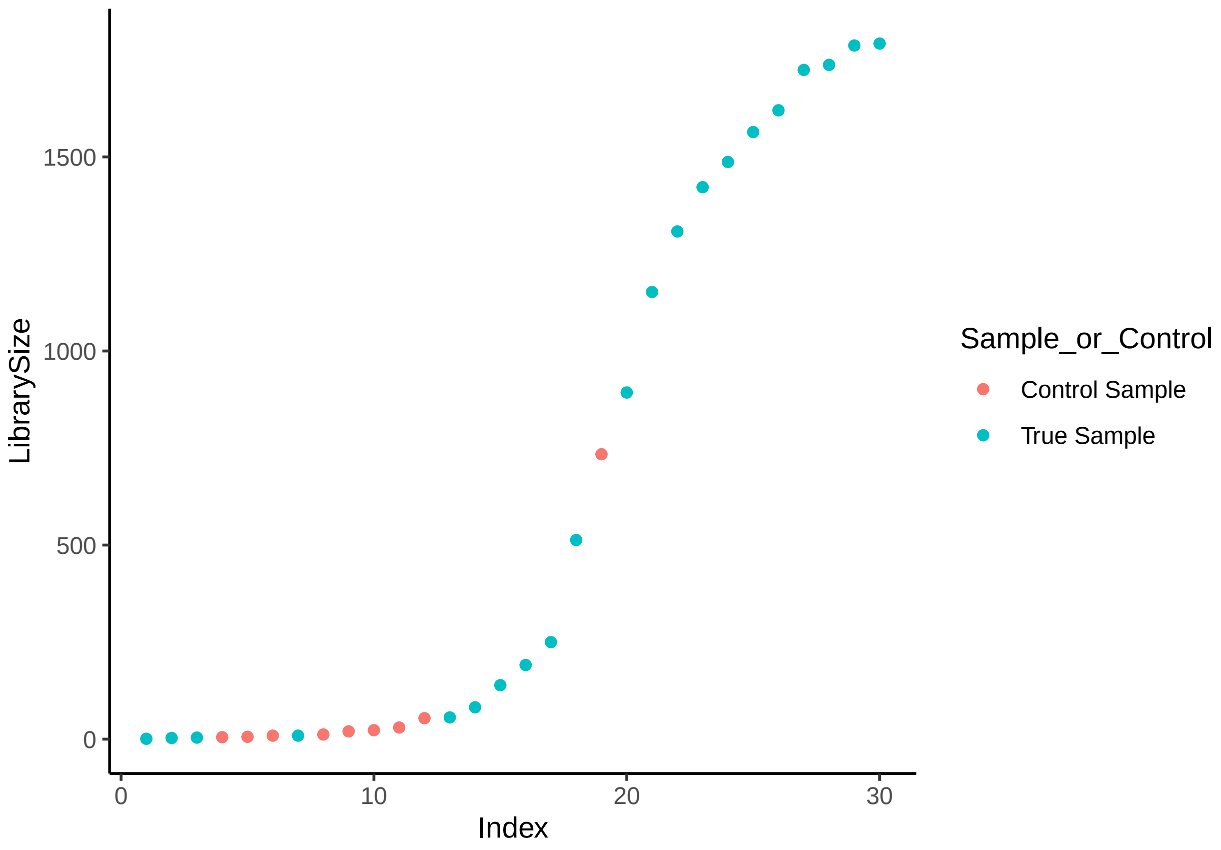
```r
df <- as.data.frame(sample_data(ps.p))
df$LibrarySize <- sample_sums(ps.p)
df <- df[order(df$LibrarySize),]
df$Index <- seq(nrow(df))

# pdf("libsize_trueVScontrol.pdf")
ggplot( data=df,
        aes(x=Index, y=LibrarySize, color=Sample_or_Control)
      ) +
  geom_point() +
  theme_classic()
```



```r
# dev.off()
```

Zoom – Plot Lib Size vs. True or Contam

```r
df[1:30,] %>%
  ggplot(
    aes(x=Index, y=LibrarySize, color=Sample_or_Control)
  ) +
  geom_point() +
  theme_classic()
```



Call Contaminants

```
thresh = c(0.1, 0.25, 0.5)
contamdf.ls = list()
for( i in seq_along(thresh) ){
  ln = paste0( "prev", as.character(thresh[i]) )
  contamdf.ls[[ln]] <-
    isContaminant(  ps.p,
                    method="prevalence",
                    neg="is.neg",
                    threshold=thresh[i]
    )
}

# how many contaminants per threshold
lapply(contamdf.ls,
       function(x) table(x["contaminant"]) )
```

```
## $prev0.1
## contaminant
## FALSE  TRUE
##   323     2
##
## $prev0.25
## contaminant
## FALSE  TRUE
##   316     9
```

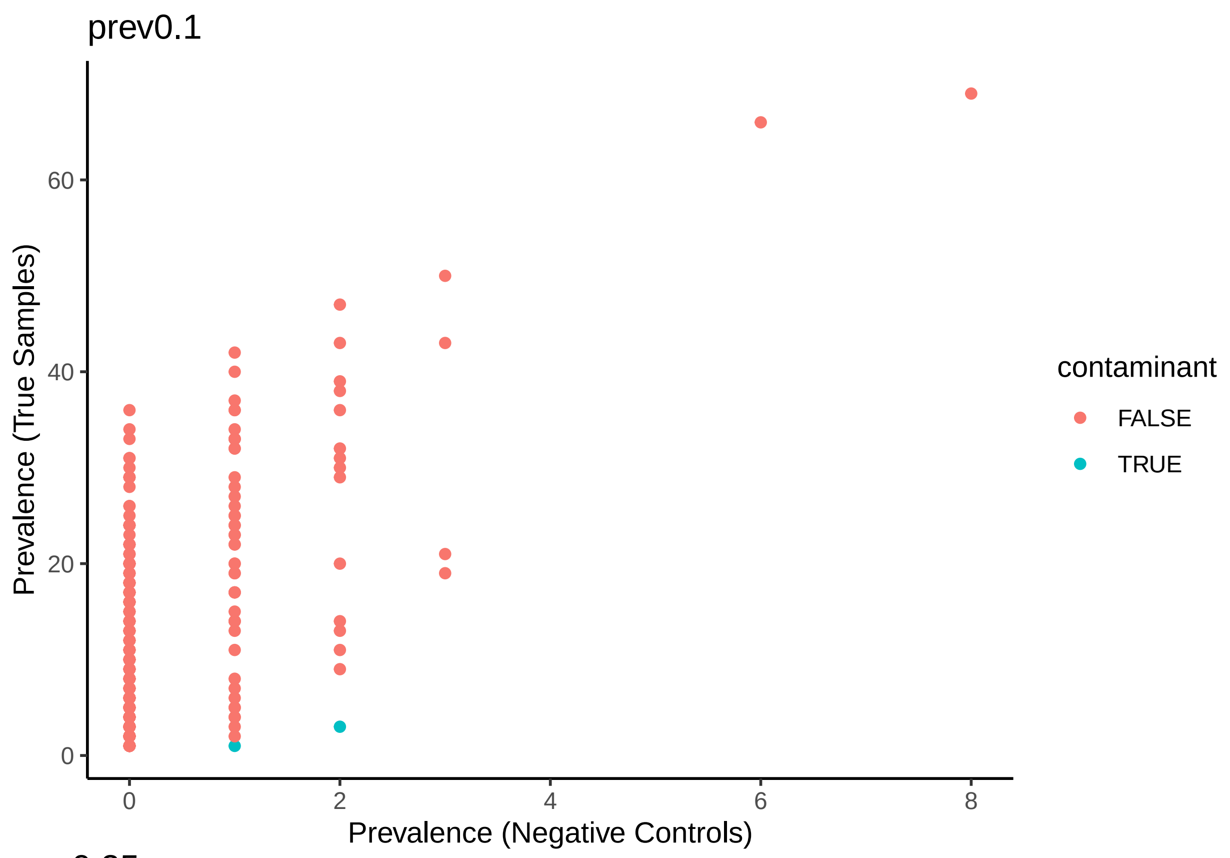
```
##
## $prev0.5
## contaminant
## FALSE  TRUE
##   306    19
```

Pres-Abs

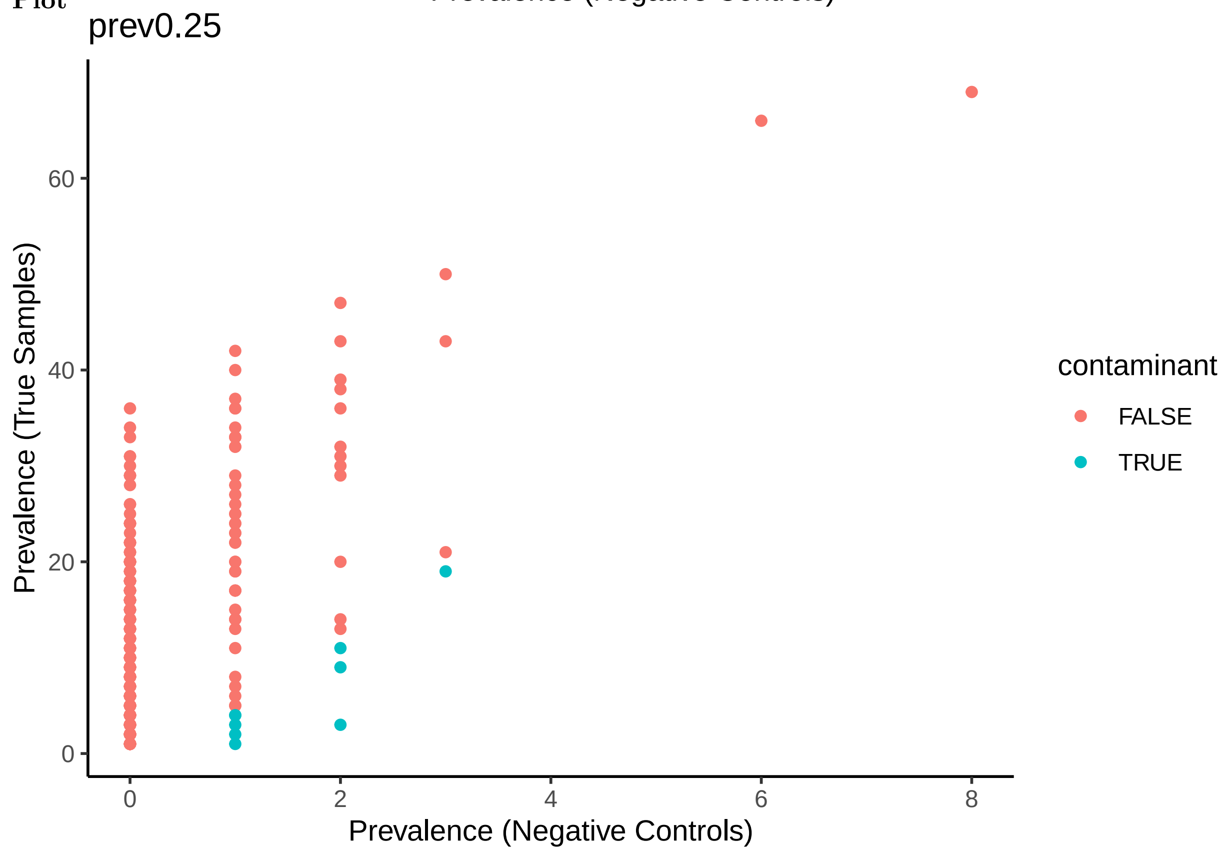
Make phyloseq object of presence-absence in negative controls and true samples

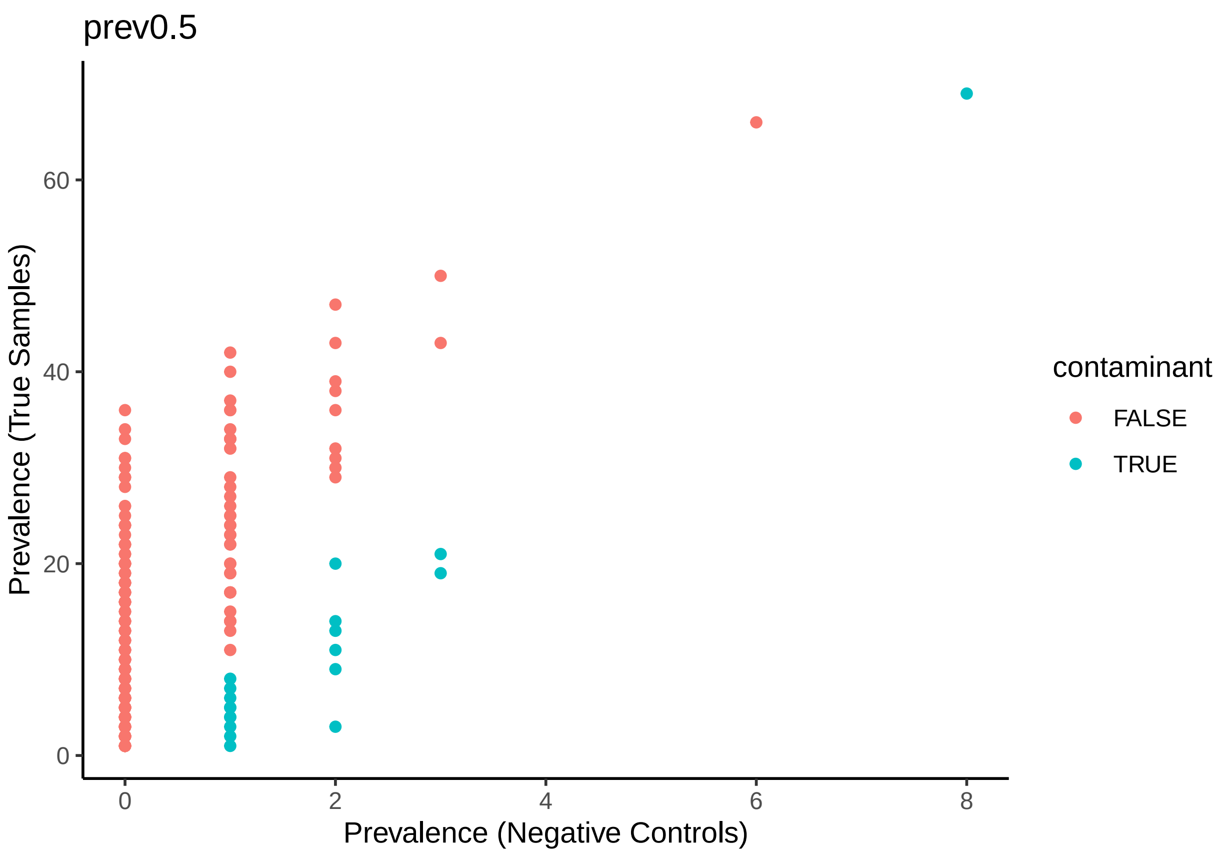
```
ps.p.pa <- transform_sample_counts(ps.p, function(abund) 1*(abund>0))

# control samples pres-abs
ps.p.pa.neg <-
  prune_samples(  sample_data(ps.p.pa)$Sample_or_Control == "Control Sample",
                  ps.p.pa
  )
# true samples pres-abs
ps.p.pa.pos <-
  prune_samples(  sample_data(ps.p.pa)$Sample_or_Control == "True Sample",
                  ps.p.pa
  )
```

```
# ---- plot contam prevalence
for( ln in names(contamdf.ls) ) {
  # ---- data frame of pres-abs by contaminant.yn
  df.pa <-
    data.frame( pa.pos = taxa_sums(ps.p.pa.pos),
                pa.neg = taxa_sums(ps.p.pa.neg),
                contaminant = contamdf.ls[[ln]]$contaminant
  )

  # ---- plot
  print(
    ggplot( data=df.pa,
          aes(x=pa.neg, y=pa.pos, color=contaminant)
    ) +
      geom_point() +
      xlab("Prevalence (Negative Controls)") +
      ylab("Prevalence (True Samples)") +
      ggtitle(ln) +
      theme_classic()
  )
}
```


prev0.1
Prevalence (True Samples)
Prevalence (Negative Controls)
0
20
40
60
0
2
4
6
8
contaminant
FALSE
TRUE

Plot


prev0.25
Prevalence (True Samples)
Prevalence (Negative Controls)
0
20
40
60
0
2
4
6
8
contaminant
FALSE
TRUE



Prune contams

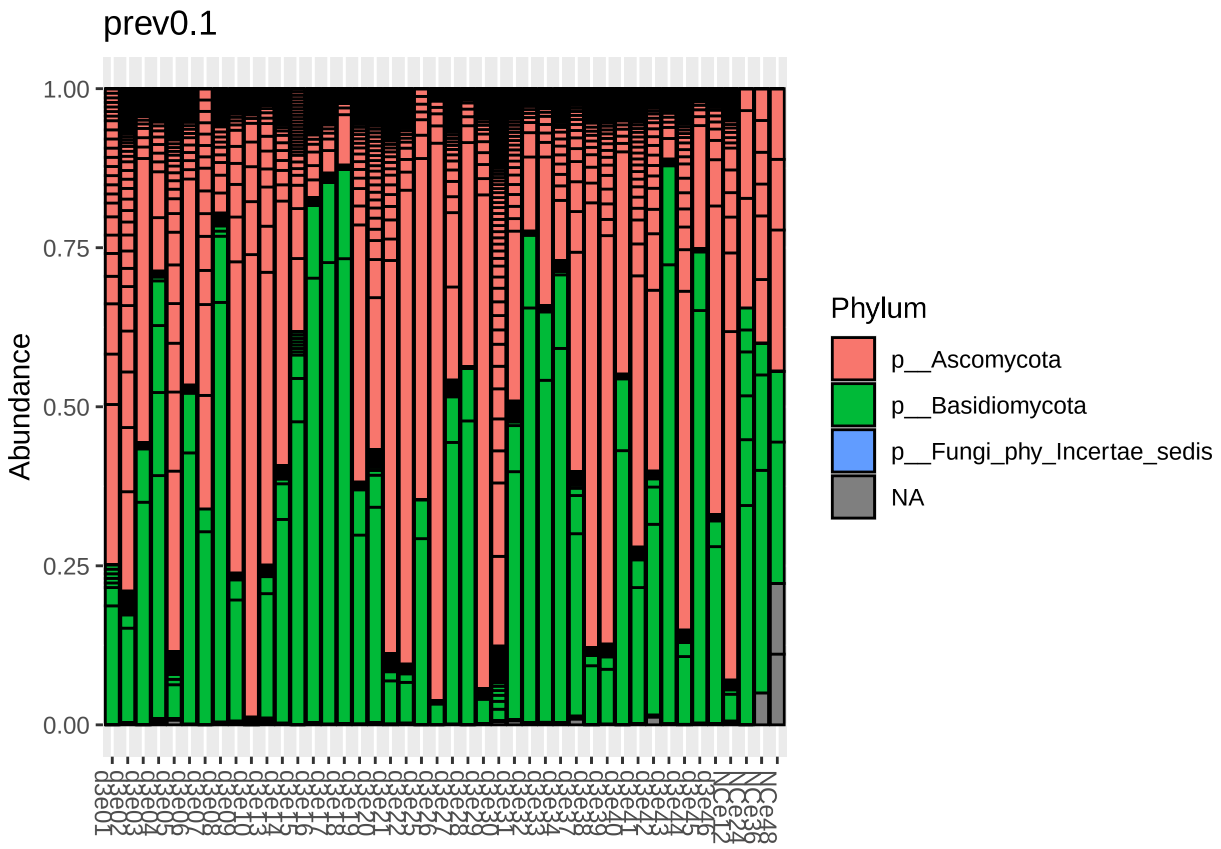
```
ps.nocontam.ls = list()
for( ln in names(contamdf.ls) ) {
  # ---- prune contaminants
  asv.keep =
    contamdf.ls[[ln]] %>% filter(contaminant == "FALSE") %>% rownames()
  ps.nocontam.ls[[ln]] = prune_taxa(asv.keep, ps.p)

  # ---- save noncontam
  f = file.path(rds.dir, paste0("ps.p.decontam.", ln, ".RDS"))
  saveRDS(ps.nocontam.ls[[ln]], f)
}
```

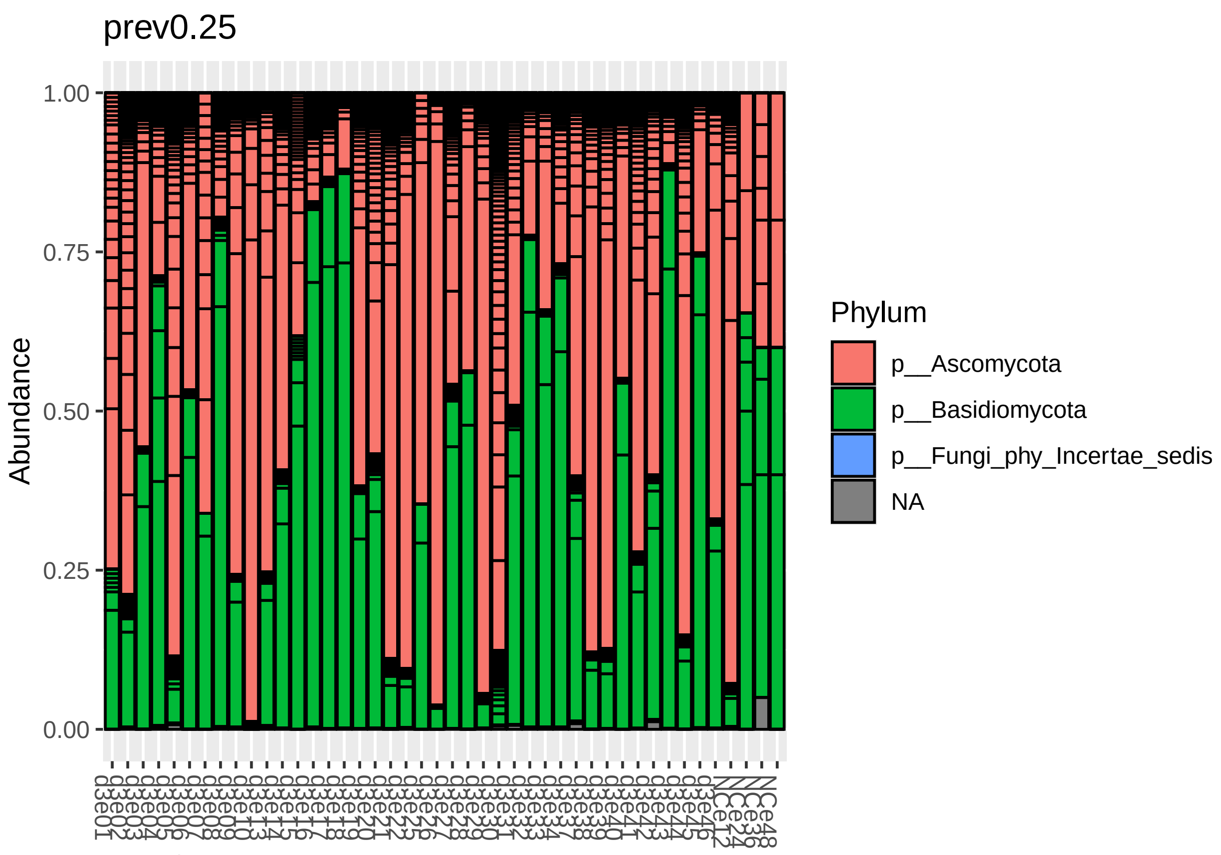
StackedBar

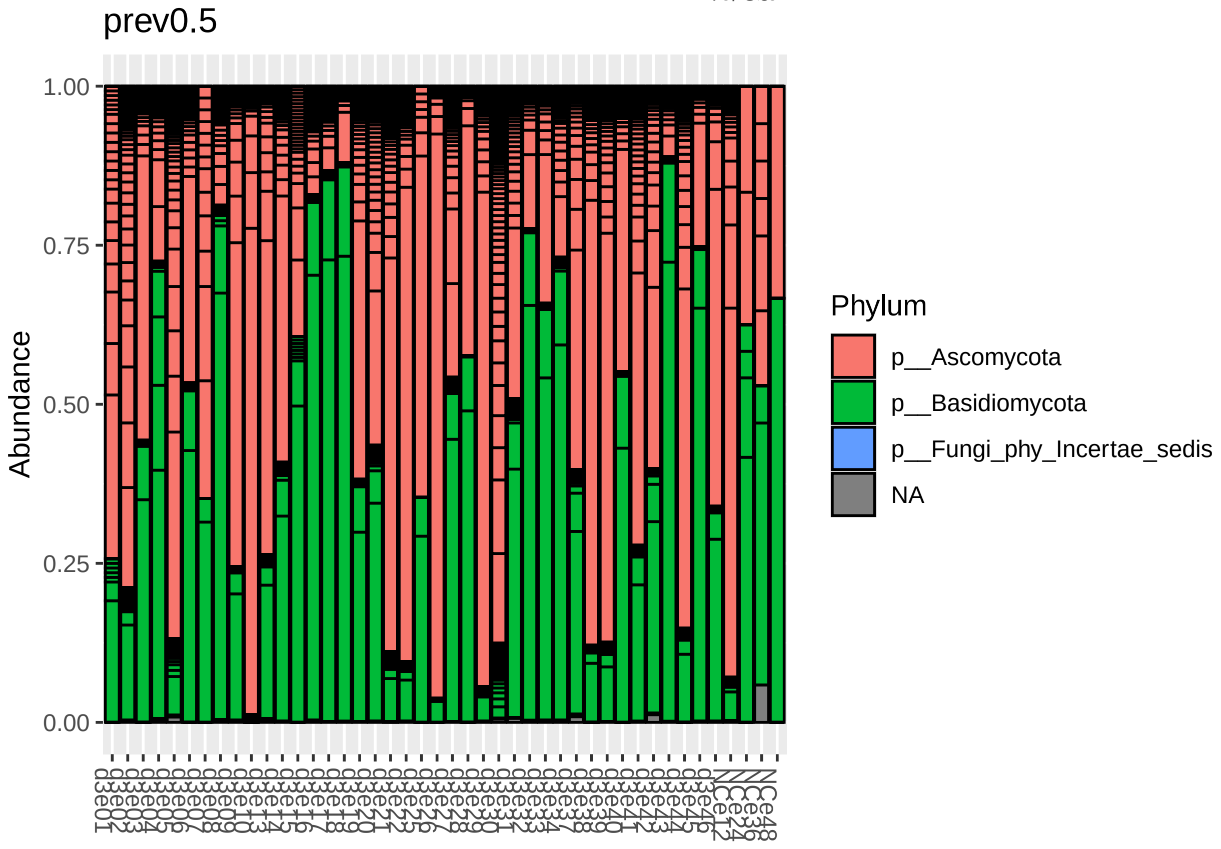
```
# update sample_data
samdat = samdat %>%
  mutate(
    exp.name = case_when(
      str_detect(seq.id, "^d3") ~ "decon3",
      str_detect(seq.id, "^NC") ~ "decon3",
      str_detect(seq.id, "^TI") ~ "testInf",
      str_detect(seq.id, "^TS") ~ "TSinf",
      .default = "other"
    )
  )
for( i in seq_along(ps.nocontam.ls) ) {
  sample_data(ps.nocontam.ls[[i]]) = samdat
}

## ---- make relabund
rab.ps.nocontam.ls = make_rel_abund(ps.nocontam.ls)
```

Plot noncontam DECON3 by phylum

```
for( ln in names(rab.ps.nocontam.ls) ) {
  print( rab.ps.nocontam.ls[[ln]] %>%
    subset_samples(exp.name == "decon3") %>%
    plot_bar(fill="Phylum") +
    xlab(NULL) +
    ggtitle(ln) )
}
```

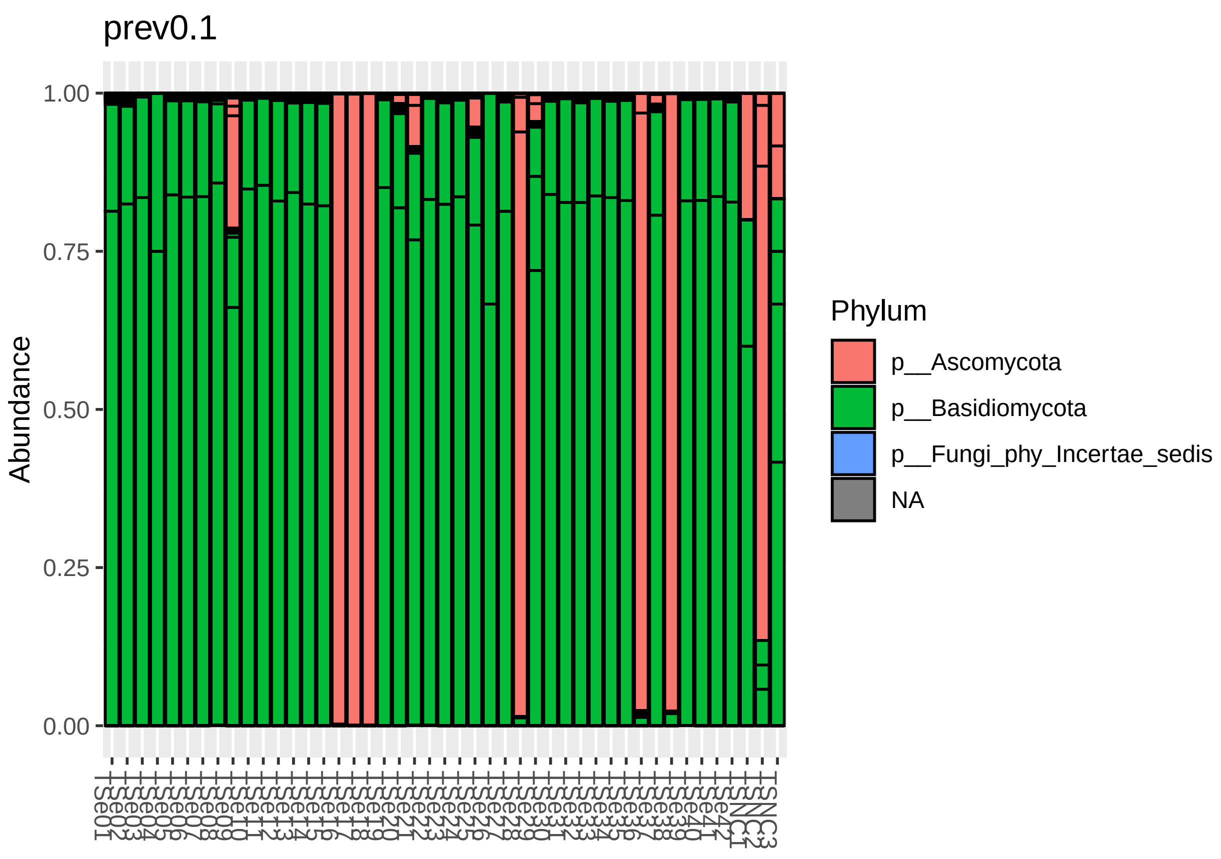


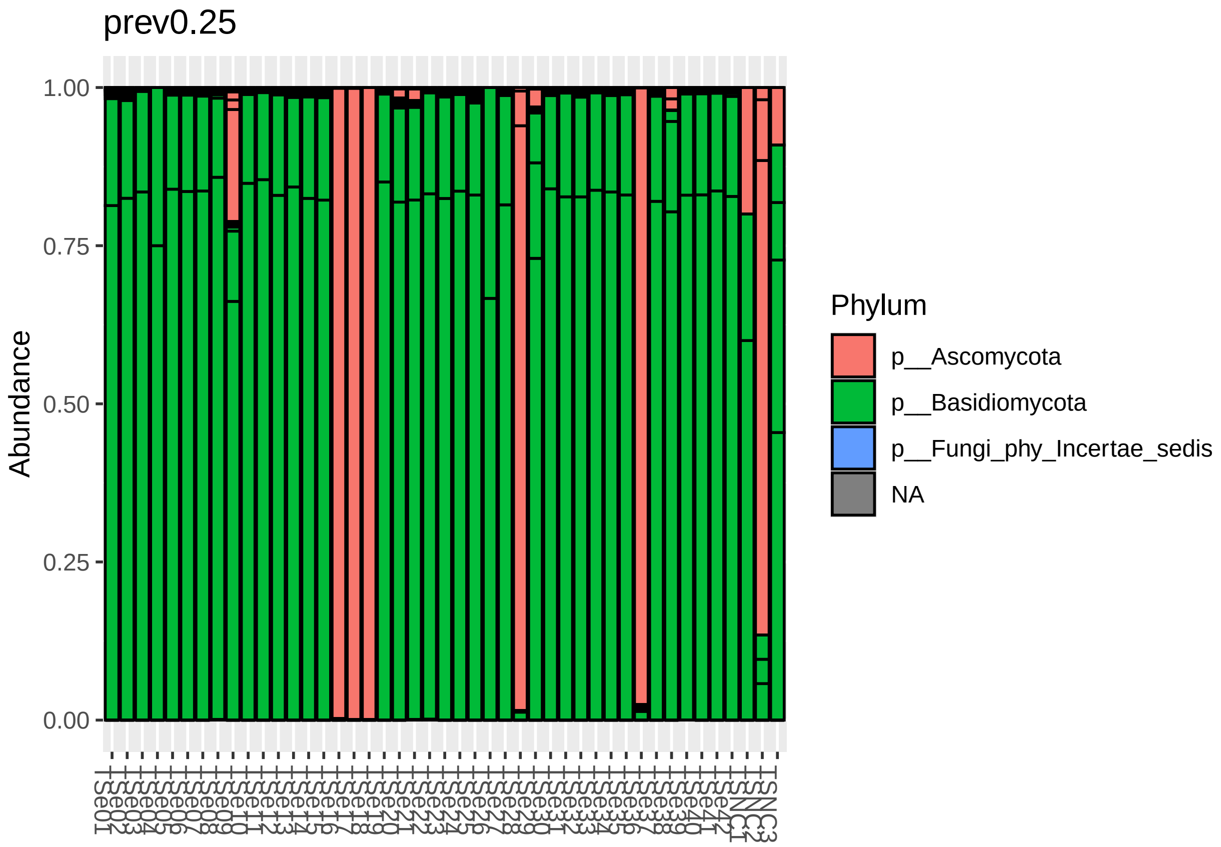


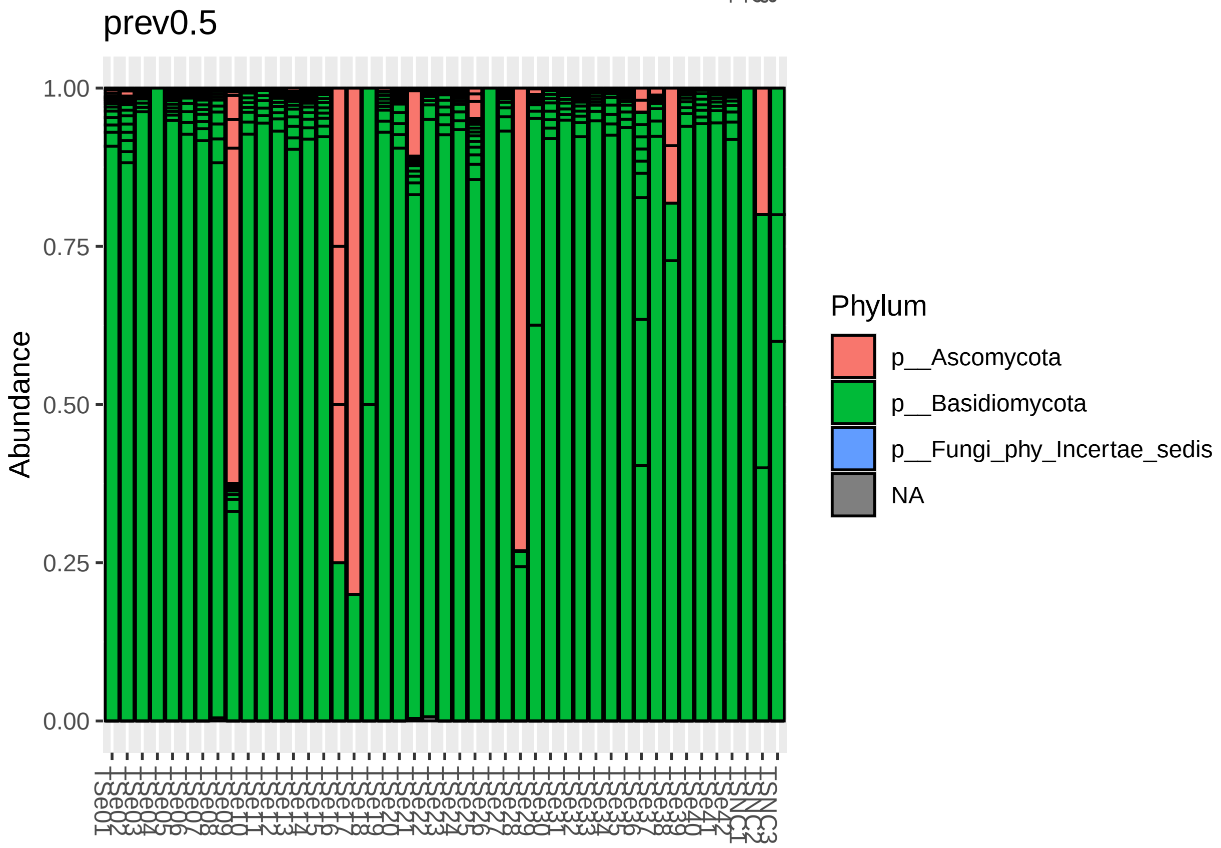


Plot noncontam TSinf by phylum

```r
for( ln in names(rab.ps.nocontam.ls) ) {
  print( rab.ps.nocontam.ls[[ln]] %>%
    subset_samples(exp.name == "TSinf") %>%
    plot_bar(fill="Phylum") +
    xlab(NULL) +
    ggtitle(ln) )
}
```

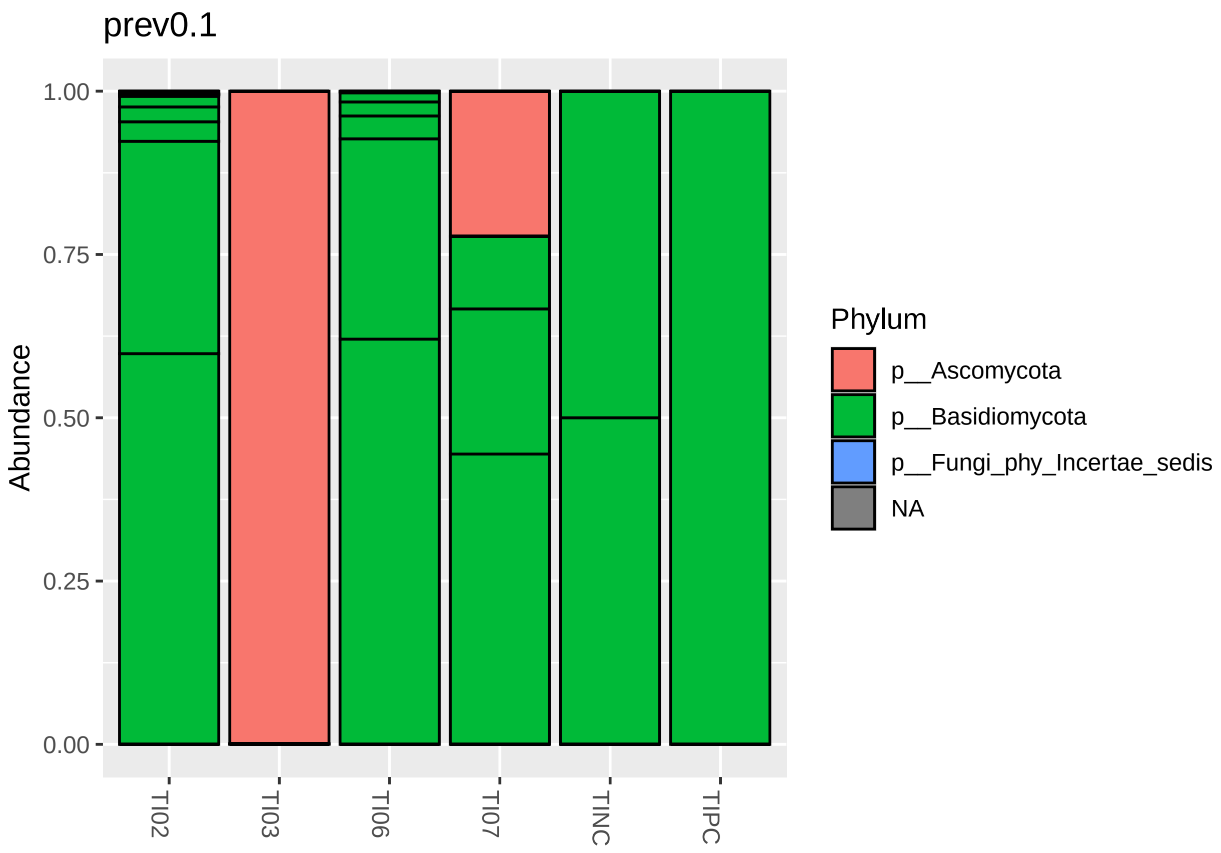


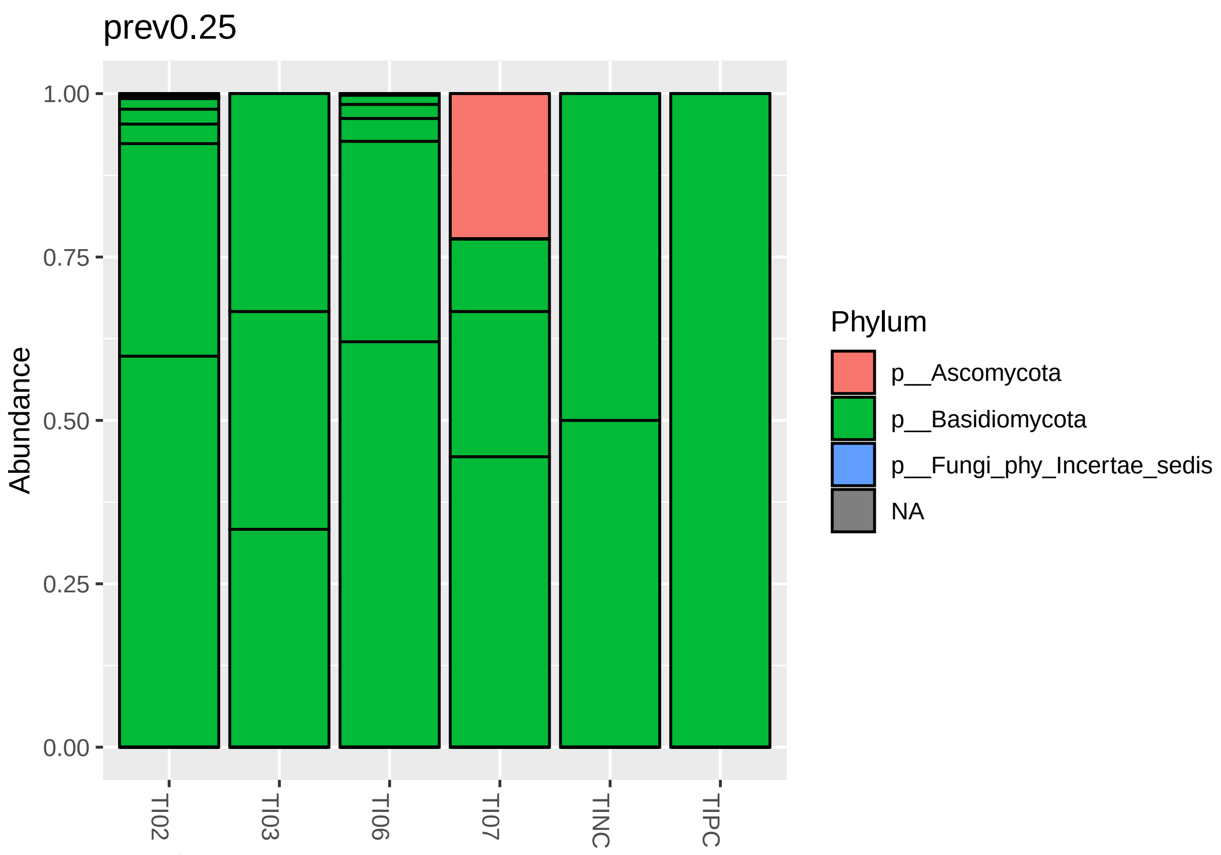


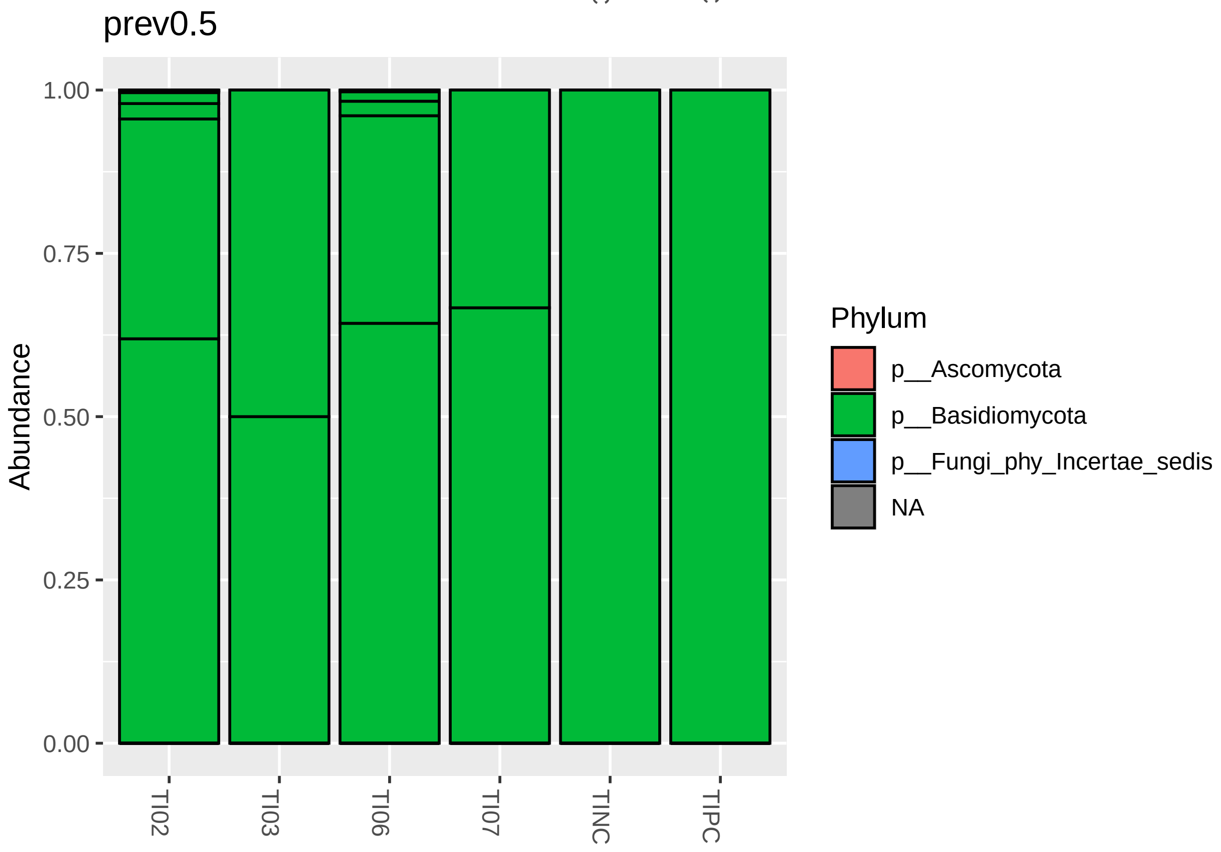


Plot testInf (even though we don't really care about this dataset)

```r
for( ln in names(rab.ps.nocontam.ls) ) {
  print( rab.ps.nocontam.ls[[ln]] %>%
    subset_samples(exp.name == "testInf") %>%
    plot_bar(fill="Phylum") +
    xlab(NULL) +
    ggtitle(ln) )
}
```

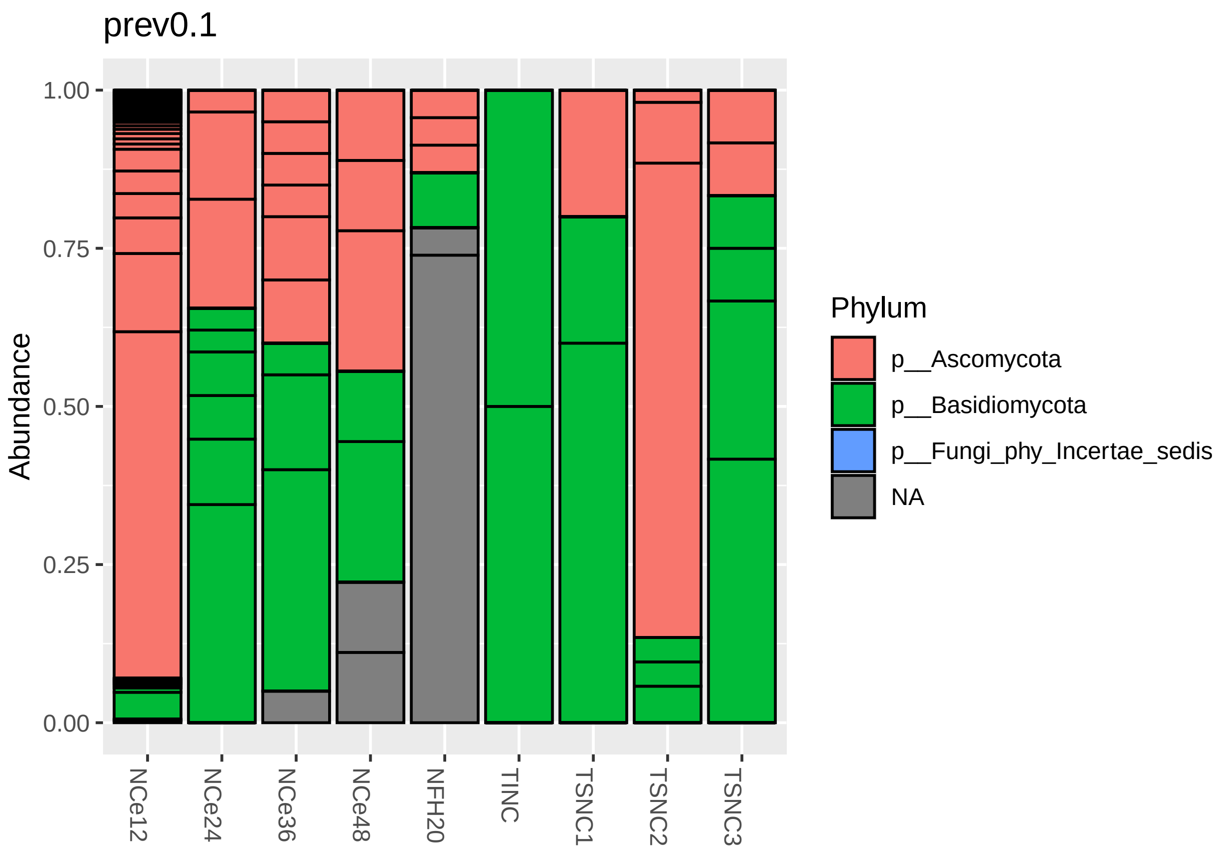




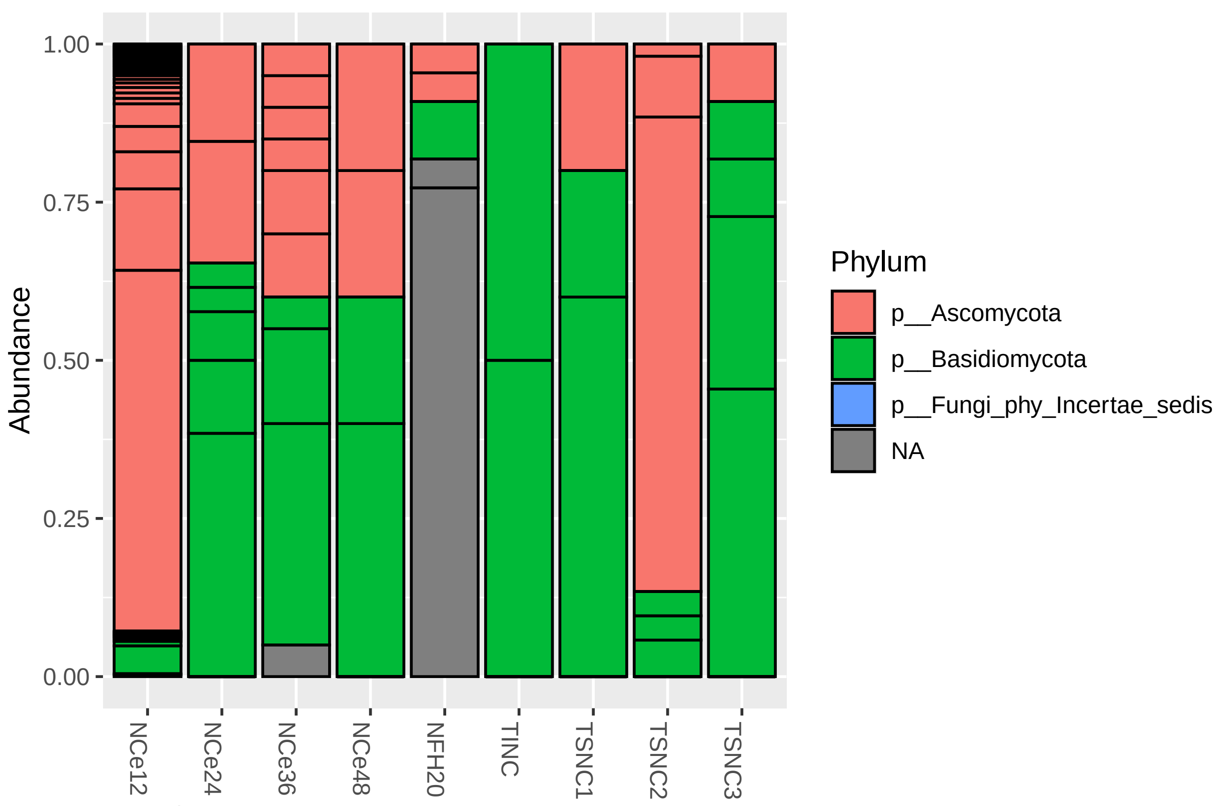


Check how controls look after decontam

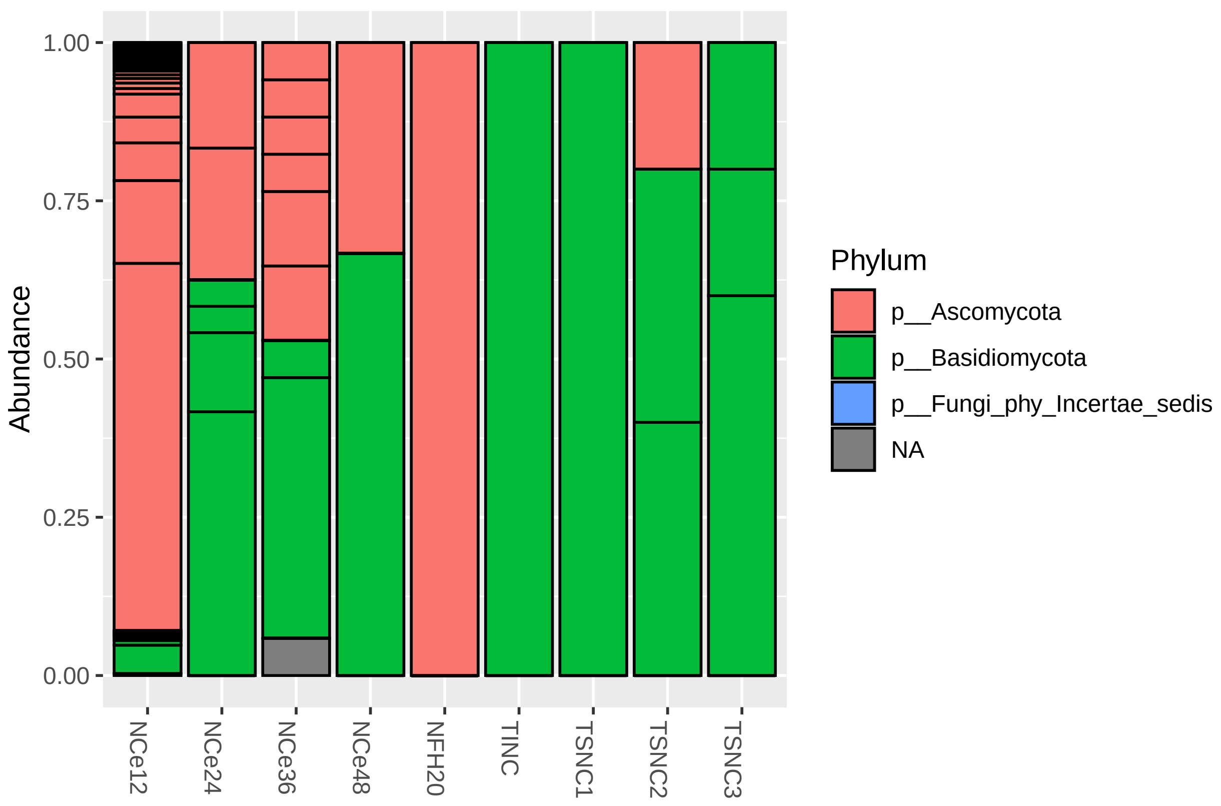
```r
for( ln in names(rab.ps.nocontam.ls) ) {
  print(
    prune_samples(controls.sn, rab.ps.nocontam.ls[[ln]]) %>%
    plot_bar(fill="Phylum") +
    xlab(NULL) +
    ggtitle(ln) )
}
```



prev0.25



prev0.5



```
prune_samples(controls.sn, ps.nocontam.ls[[ln]]) %>% sample_sums() %>% sort()
```

```
## NFH20 TSNC1 NCe48  TINC TSNC2 TSNC3 NCe36 NCe24 NCe12
##     1     1     3     3     5     5    17    24   688
```

nums.csv

```
## ---- decontam stats
decontam.totsize.df <-
  data.frame( ps.p.nreads = sum(sample_sums(ps.p)),
              prev0.1.nreads = sum(sample_sums(ps.nocontam.ls$prev0.1)),
              prev0.25.nreads = sum(sample_sums(ps.nocontam.ls$prev0.25)),
              prev0.5.nreads = sum(sample_sums(ps.nocontam.ls$prev0.5))
  )
decontam.totsize.df <-
  decontam.totsize.df %>%
  mutate(prev0.1.percL = (ps.p.nreads-prev0.1.nreads)/ps.p.nreads*100) %>%
  mutate(prev0.25.percL = (ps.p.nreads-prev0.25.nreads)/ps.p.nreads*100) %>%
  mutate(prev0.5.percL = (ps.p.nreads-prev0.5.nreads)/ps.p.nreads*100)

samsums.decontam.df <-
  data.frame(
    ps.p.samsums = sample_sums(ps.p),
    prev0.1.samsums = sample_sums(ps.nocontam.ls$prev0.1),
    prev0.25.samsums = sample_sums(ps.nocontam.ls$prev0.25),
    prev0.5.samsums = sample_sums(ps.nocontam.ls$prev0.5)
  )

decontam.stats.df <-
  samsums.decontam.df %>%
  mutate(prev0.1.nreads.lost = ps.p.samsums-prev0.1.samsums) %>%
  mutate(prev0.25.nreads.lost = ps.p.samsums-prev0.25.samsums) %>%
  mutate(prev0.5.nreads.lost = ps.p.samsums-prev0.5.samsums) %>%
  mutate(prev0.1.percL = prev0.1.nreads.lost/ps.p.samsums*100) %>%
  mutate(prev0.25.percL = prev0.25.nreads.lost/ps.p.samsums*100) %>%
  mutate(prev0.5.percL = prev0.5.nreads.lost/ps.p.samsums*100)
write.csv(decontam.stats.df, "decontam.stats.df.csv")
```

Filtering

I'm choosing to use decontam prevalence threshold of 0.1. Removing negative controls samples. Filtering highly unclassified taxa

```
ps = na_to_unclassified_taxa(ps.nocontam.ls$prev0.1)
ps.nonc = prune_samples(!(sample_names(ps) %>% grepl(pattern = "NC|NFW|NFH2")),
                        ps)

# starting with 323 taxa and 87 samples
ps.filt = subset_taxa(ps.nonc, Kingdom == "k__Fungi")              # 323 taxa
ps.filt = subset_taxa(ps.filt, Phylum != "unclass. k__Fungi")    # 309 taxa

# split by experiment
psList.filt = list(
  decon3 = subset_samples(ps.filt, exp.name == "decon3"),
  tsinf = subset_samples(ps.filt, exp.name == "TSinf")
)
```

Sample Data

```
## ---- add sample data -----------------------------------------------------------------------

decon3.nonc.samdat =
  read.csv("~/Documents/ch3_analyses/decons_master_sampledata.csv")  %>%
  filter(grepl(pattern="^d3", seq.id)) %>%
  filter(!grepl(pattern="NC|NFW|NFH", seq.id))
tsinf.nonc.samdat =
  read.csv("~/Documents/ch3_analyses/TSinf_master_sampledata.csv")  %>% filter(!grepl(pattern="NC|NFW|NI
sdList = list(
        decon3 = decon3.nonc.samdat,
        tsinf = tsinf.nonc.samdat,
        decon3_noOth = decon3.nonc.samdat
    )

# ---- check if sample names match between ps and samdat
for( i in seq_along(psList.filt) ){
    if(  all( sdList[[i]]$s.id %in% sample_names(psList.filt[[i]]) ) ) {
        cat(">>> sample names for",
            names(psList.filt)[i],
            "phyloseq object s.id are the same as s.id in",
            names(sdList)[i],
            "samdat.\n"
        )

        rownames(sdList[[i]]) = sdList[[i]]$s.id
        sample_data(psList.filt[[i]]) = sdList[[i]]

        cat(">>> sample_data slot filled for phyloseq object named",
            names(psList.filt)[i],
            ".\n"
        )

        f = paste0(names(psList.filt)[i], ".ps.filt.RDS")
        saveRDS(psList.filt[[i]], f)
        cat(">>> Saving file", f,"... ... ...\n")

    } else {
        cat(">>> sample names for",
            names(psList.filt)[i],
            "phyloseq object do not match s.id's for",
            names(sdList)[i],
            "samdat! Please reorder.\n"
        )
    }
}
```

```
## >>> sample names for decon3 phyloseq object s.id are the same as s.id in decon3 samdat.
## >>> sample_data slot filled for phyloseq object named decon3 .
## >>> Saving file decon3.ps.filt.RDS ... ... ...
## >>> sample names for tsinf phyloseq object s.id are the same as s.id in tsinf samdat.
## >>> sample_data slot filled for phyloseq object named tsinf .
## >>> Saving file tsinf.ps.filt.RDS ... ... ...
```

make expect.inf var

```
ps = psList.filt$tsinf

# create variable for samples we expect to be infected or not
ps =
  ps %>%
  ps_mutate(expect.inf = "no", .after="day.post.inf") %>%
  ps_mutate(
    expect.inf = replace(expect.inf,
                         s.type == "fungus garden" &
                           treatment == "Trichoderma" &
                           ants.yn == "no" &
                           day.post.inf > 0,
                         "yes")
    ) %>% ps_mutate(
      expect.inf = replace(expect.inf,
                           s.type == "trash" &
                             treatment == "Trichoderma",
                           "yes")
    ) %>% ps_mutate(
        expect.inf = replace(expect.inf,
                             s.type == "trash" &
                               treatment != "Trichoderma",
                             "maybe")
  )

# set the one -ants NT that we know had Trichoderma NT-r1-ants (TSe38 - day4, TSe35 = day2)
ps = ps %>%
  ps_mutate(
    expect.inf = replace(expect.inf,
                         s.id == "TSe38",
                         "yes")
    ) %>%
  ps_mutate(
    expect.inf = replace(expect.inf,
                         s.id == "TSe35",
                         "maybe")
  )

# save
psList.filt$tsinf = ps
```

Make cultivar fungus

```
# define cultivar fungus non-0.01 other filt
psList.filt.cf = lapply(psList.filt, make_cultivar_fungus)
```

```
## >>> Potential cultivar fungus ASVs detected. Creating fasta ... ... ...
## >>> Writing file ./dada2/fasta/cultivar_fungus_ASVs.fasta  ... ... ...
## >>> Potential cultivar fungus ASVs detected. Creating fasta ... ... ...
## >>> Writing file ./dada2/fasta/cultivar_fungus_ASVs.fasta  ... ... ...
```

Decon3 ASV1 matches to Leucoagaricus gonglyophorus with BLASTn.

```
tax_table(psList.filt.cf$decon3)["ASV1","Genus"] = "Cultivar Fungus"
psList.filt.cf$decon3 = tax_glom(psList.filt.cf$decon3, "Genus")
```

Decon3 ASV2 matches Acer ITS2 ?!?!

```
psList.filt.cf$decon3 =
  prune_taxa(taxa_names(psList.filt.cf$decon3) != "ASV2",
             psList.filt.cf$decon3 )
```

Save Final ps objects

```
# saveRDS(psList.filt.cf, "psList.filt.cf.RDS")
```

other 0.01 filter

```
# set taxa <1% in all samples to other
psList.filt.cf.0.01 = lapply(psList.filt.cf, per_sample_taxafilt, glom = "Genus")
```

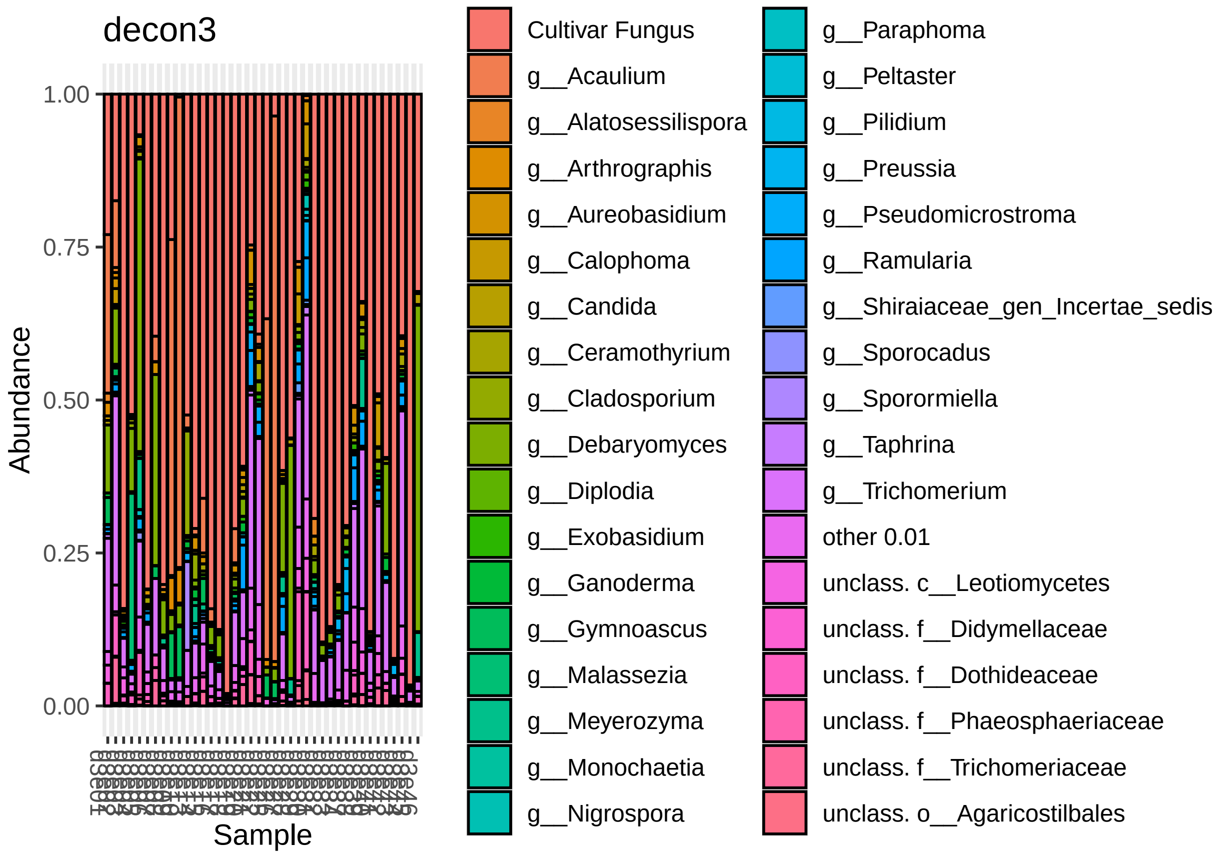
```
##
## >>> Glomming by Genus ... ... ...
##
## >>> Glomming by Genus ... ... ...
```

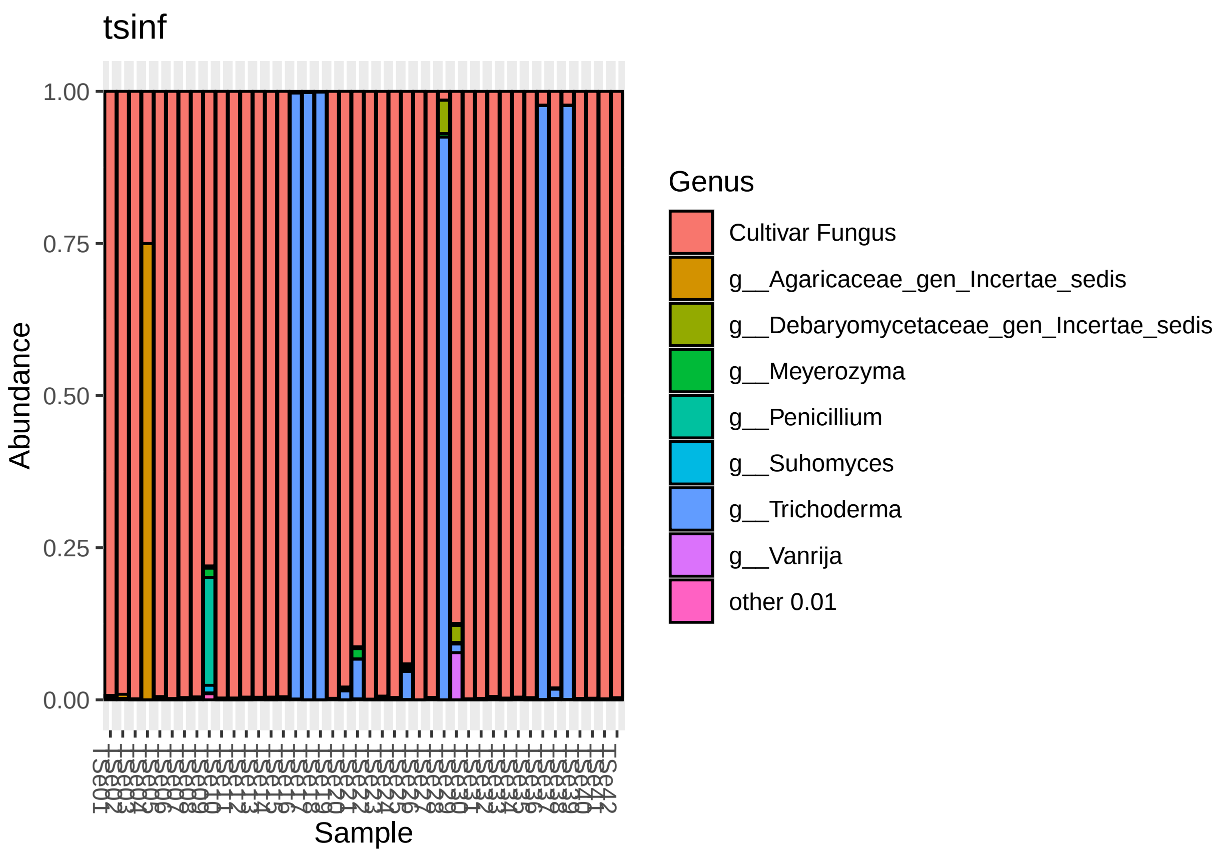
```
psList.filt.cf.0.01 # d3: 36 taxa, ts: 9 taxa
```

```
## $decon3
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 36 taxa and 40 samples ]
## sample_data() Sample Data:       [ 40 samples by 17 sample variables ]
## tax_table()   Taxonomy Table:    [ 36 taxa by 7 taxonomic ranks ]
## refseq()      DNAStringSet:      [ 36 reference sequences ]
##
## $tsinf
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 9 taxa and 42 samples ]
## sample_data() Sample Data:       [ 42 samples by 15 sample variables ]
## tax_table()   Taxonomy Table:    [ 9 taxa by 7 taxonomic ranks ]
## refseq()      DNAStringSet:      [ 9 reference sequences ]
```

stacked bar plot

```
fin.psList = psList.filt.cf.0.01
# saveRDS(psList.filt.cf.0.01, "psList.filt.cf.0.01.RDS")

for( ln in names(fin.psList) ) {
  print(
    fin.psList[[ln]] %>%
      make_rel_abund() %>%
      plot_bar(fill="Genus") +
      ggtitle(ln)
  )
}
```

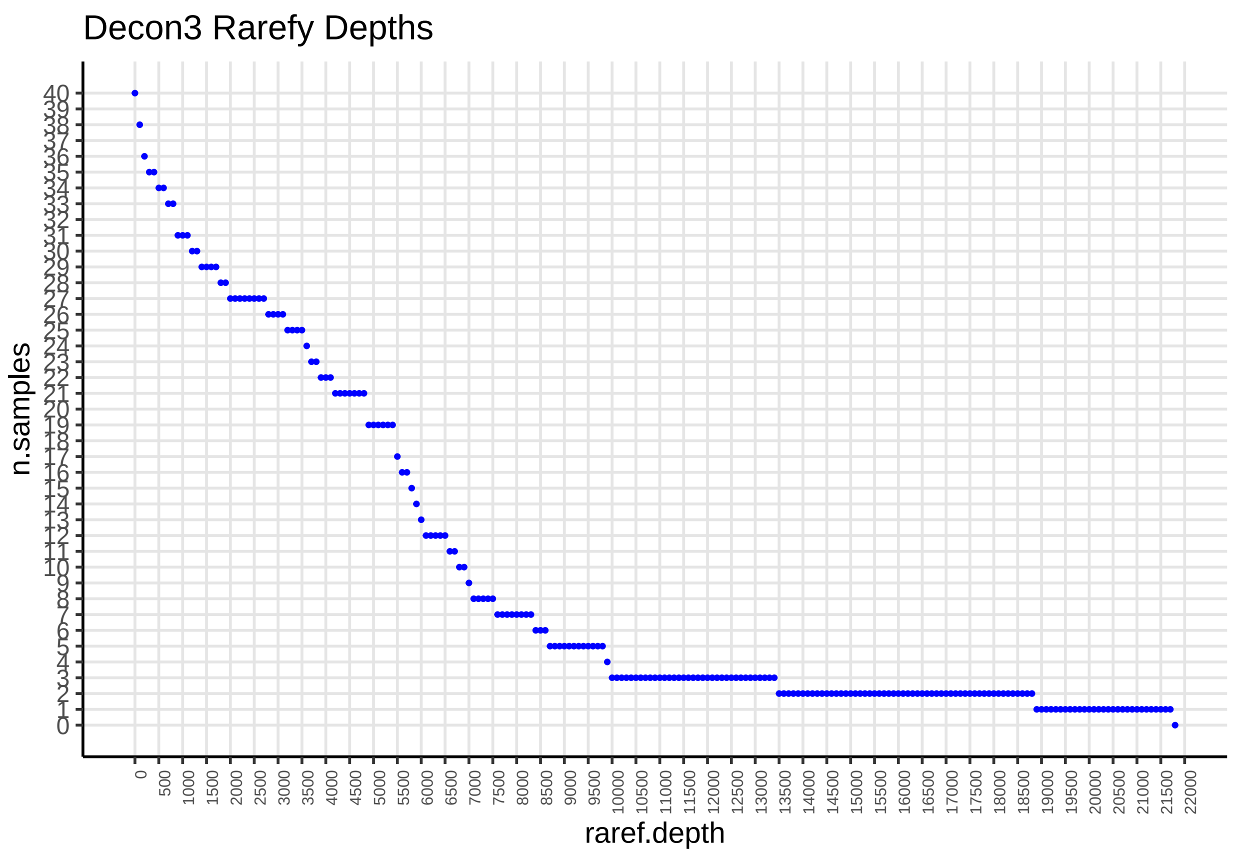




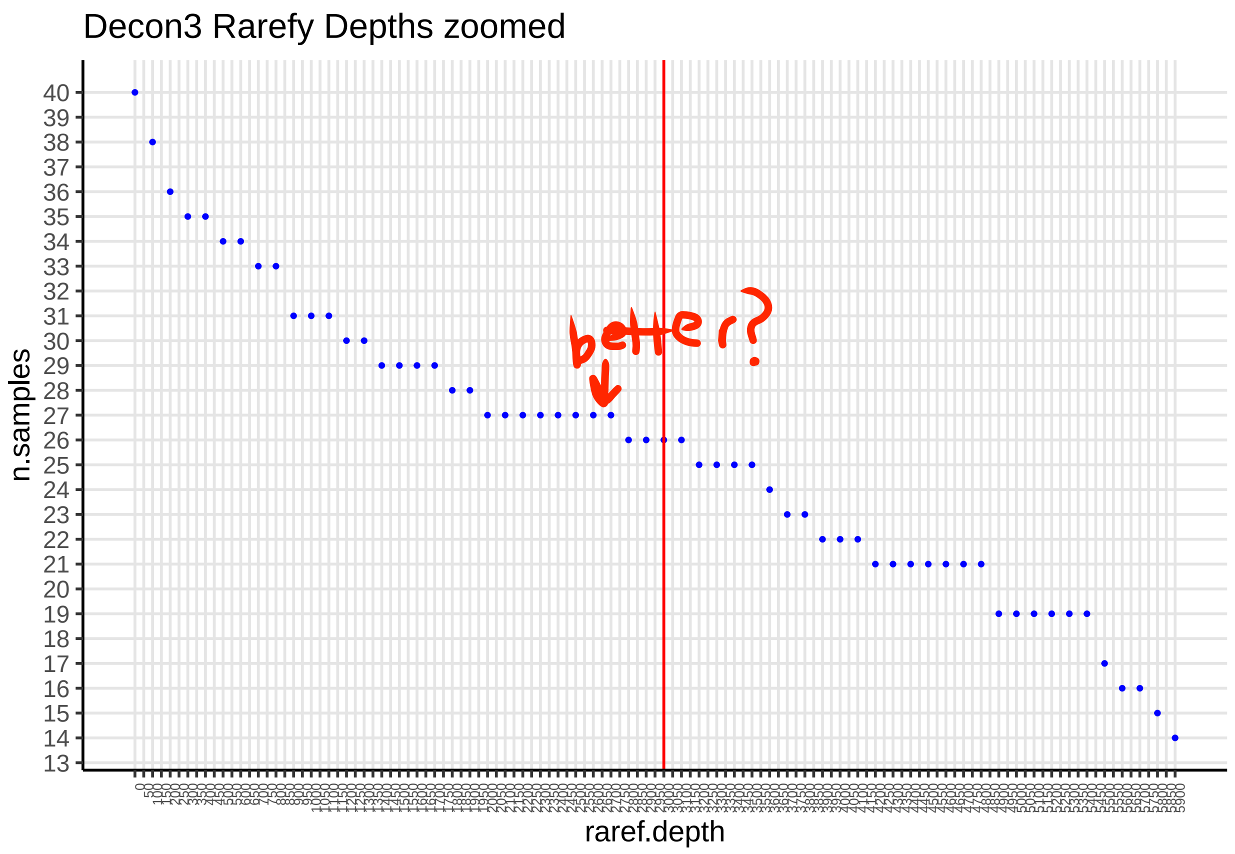
Rarefy

Decon3 explore

```
## plot raref threshold increments of 50 reads
ps = fin.psList$decon3

m = max(sample_sums(ps))
b = 100
xvals = seq(0, m+b, by=b)
yvals = sapply(xvals, function(x) length(which(sample_sums(ps) >= x)))
df = data.frame(raref.depth = xvals, n.samples = yvals)

ggplot( df,
        aes(x=raref.depth, y=n.samples)
    ) +
    geom_point(
        size=0.5,
        color="blue"
    ) +
    ggtitle("Decon3 Rarefy Depths") +
    scale_x_continuous(
        breaks = seq(0, m+500, by=500)
    ) +
    scale_y_continuous(
        breaks = seq(0,nsamples(ps))
    ) +
    theme_classic() +
    theme(
        axis.text.x = element_text(angle = 90, hjust=1, size=6),
        #panel.background = element_blank(),
        panel.grid.major.y  = element_line(color = "grey90"),
        panel.grid.major.x = element_line(color="grey90")
    )
```



```
#ggsave("png/decon3_rarefyexplore.png")


## zoom in on lower left side of graph
df2 = df %>% dplyr::slice(1:60)
x.max = max(df2$raref.depth)
y.max = max(df2$n.samples)

  ggplot(
      df2,
        aes(x=raref.depth, y=n.samples)
    ) +
    geom_point(
        size=0.5,
        color="blue"
    ) +
    ggtitle("Decon3 Rarefy Depths zoomed") +
    scale_x_continuous(
        breaks = seq(0, x.max, by=50)
    ) +
    scale_y_continuous(
        breaks = seq(0, y.max, by=1)
    ) +
    theme_classic() +
    theme(
        axis.text.x = element_text(angle = 90, hjust=1, size=5),
        #panel.background = element_blank(),
        panel.grid.major.y  = element_line(color = "grey90"),
        panel.grid.major.x = element_line(color="grey90")
    ) +
    geom_vline(xintercept = 3000, color="red")
```

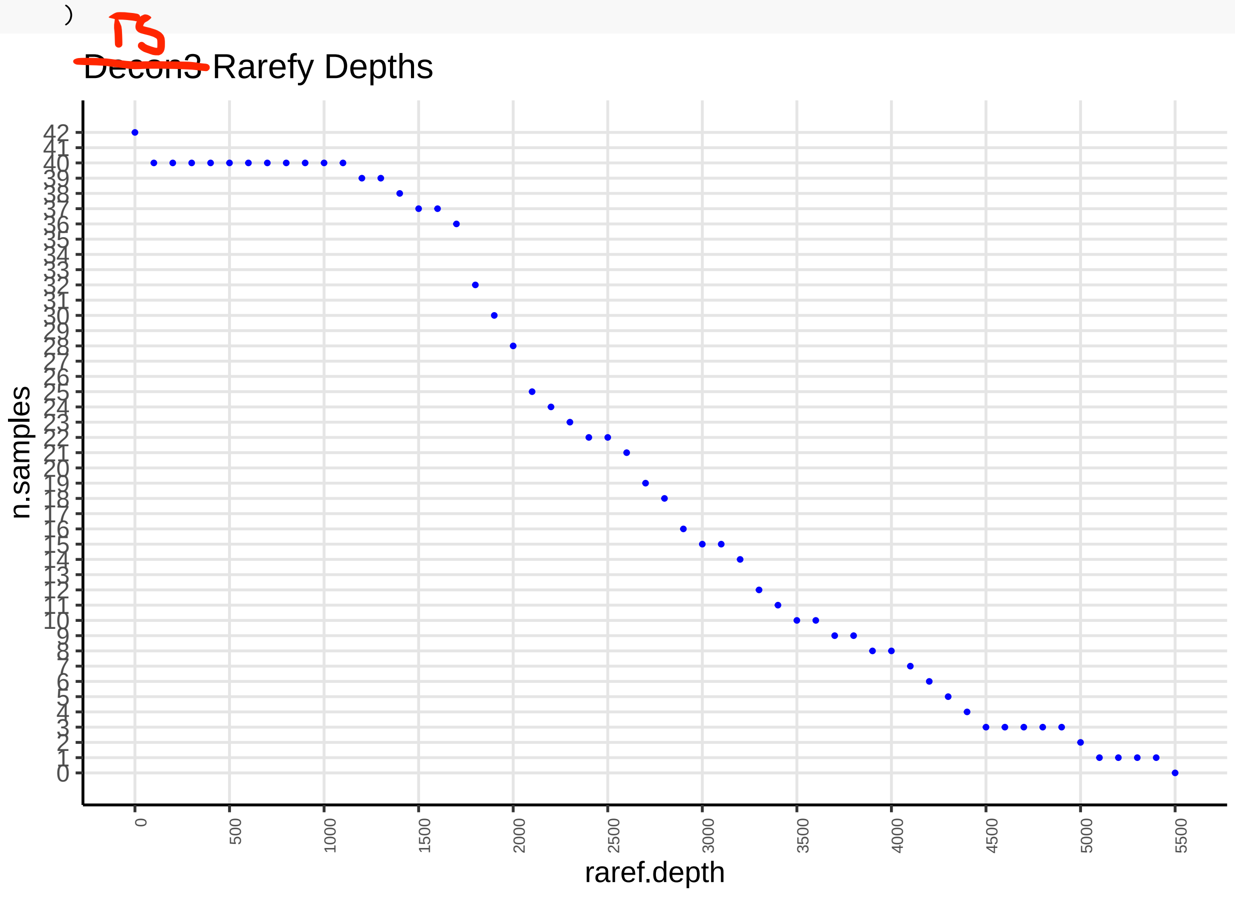


```
#ggsave("png/decon3_rarefyexplore.zoom_t700.png")
```

TSinf explore

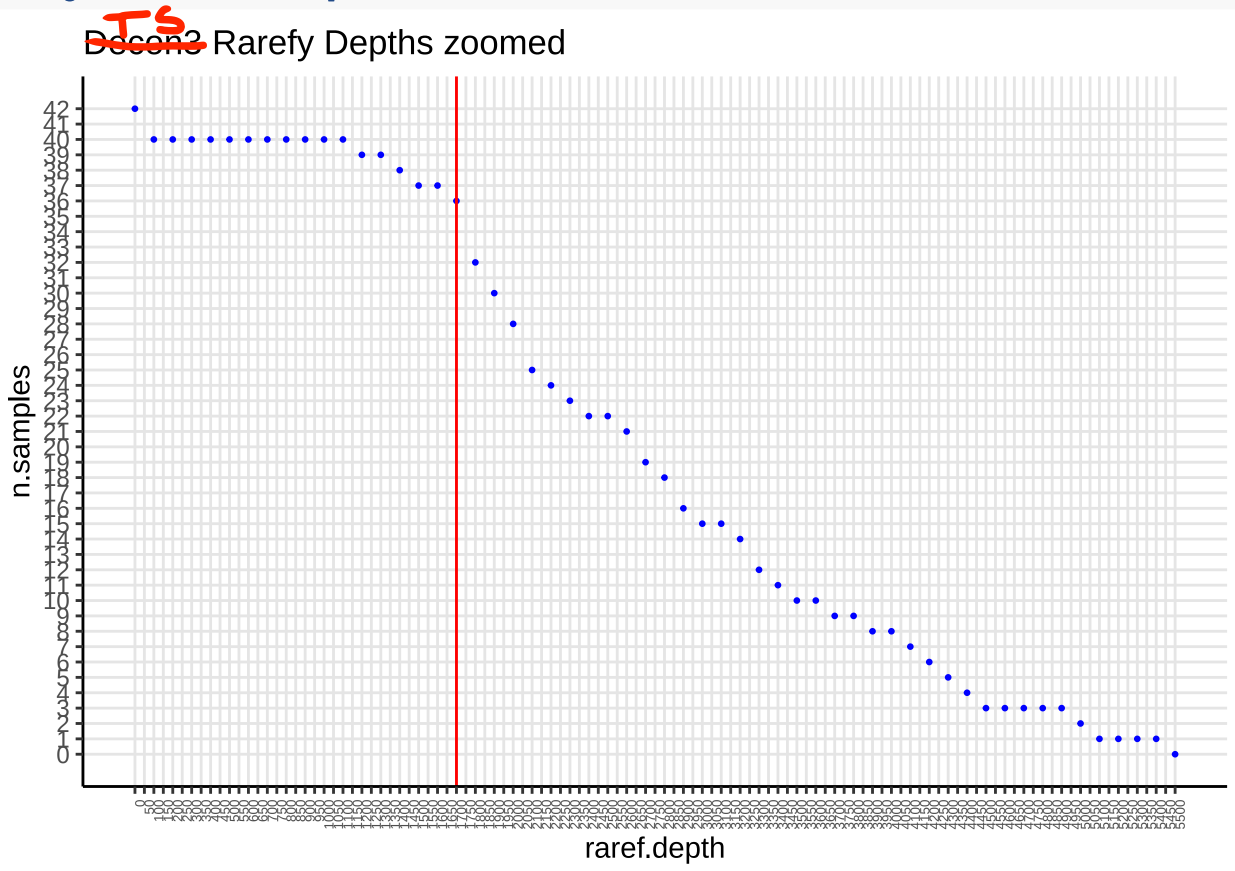
```
## plot raref threshold increments of 50 reads
ps = fin.psList$tsinf

m = max(sample_sums(ps))
b = 100
xvals = seq(0, m+b, by=b)
yvals = sapply(xvals, function(x) length(which(sample_sums(ps) >= x)))
df = data.frame(raref.depth = xvals, n.samples = yvals)
```

```
ggplot( df,
        aes(x=raref.depth, y=n.samples)
    ) +
    geom_point(
        size=0.5,
        color="blue"
    ) +
    ggtitle("Decon3 Rarefy Depths") +
    scale_x_continuous(
        breaks = seq(0, m+500, by=500)
    ) +
    scale_y_continuous(
        breaks = seq(0,nsamples(ps))
    ) +
    theme_classic() +
    theme(
        axis.text.x = element_text(angle = 90, hjust=1, size=6),
        #panel.background = element_blank(),
        panel.grid.major.y  = element_line(color = "grey90"),
        panel.grid.major.x = element_line(color="grey90")
    )
```



```
#ggsave("png/decon3_rarefyexplore.png")


## zoom in on lower left side of graph
df2 = df %>% dplyr::slice(1:60)
```

```r
x.max = max(df2$raref.depth)
y.max = max(df2$n.samples)

  ggplot(
      df2,
        aes(x=raref.depth, y=n.samples)
    ) +
    geom_point(
        size=0.5,
        color="blue"
    ) +
    ggtitle("Decon3 Rarefy Depths zoomed") +
    scale_x_continuous(
        breaks = seq(0, x.max, by=50)
    ) +
    scale_y_continuous(
        breaks = seq(0, y.max, by=1)
    ) +
    theme_classic() +
    theme(
        axis.text.x = element_text(angle = 90, hjust=1, size=5),
        #panel.background = element_blank(),
        panel.grid.major.y  = element_line(color = "grey90"),
        panel.grid.major.x = element_line(color="grey90")
    ) +
    geom_vline(xintercept = 1700, color="red")
```



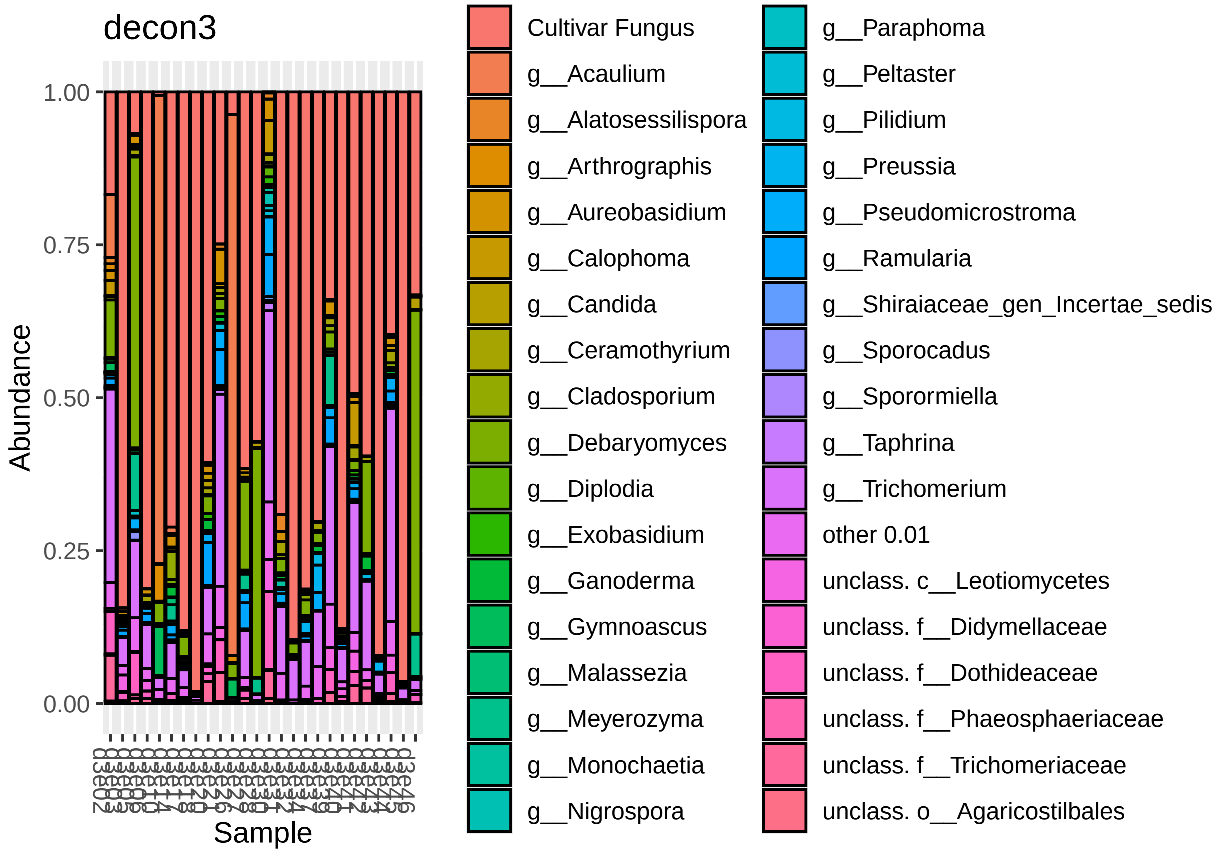
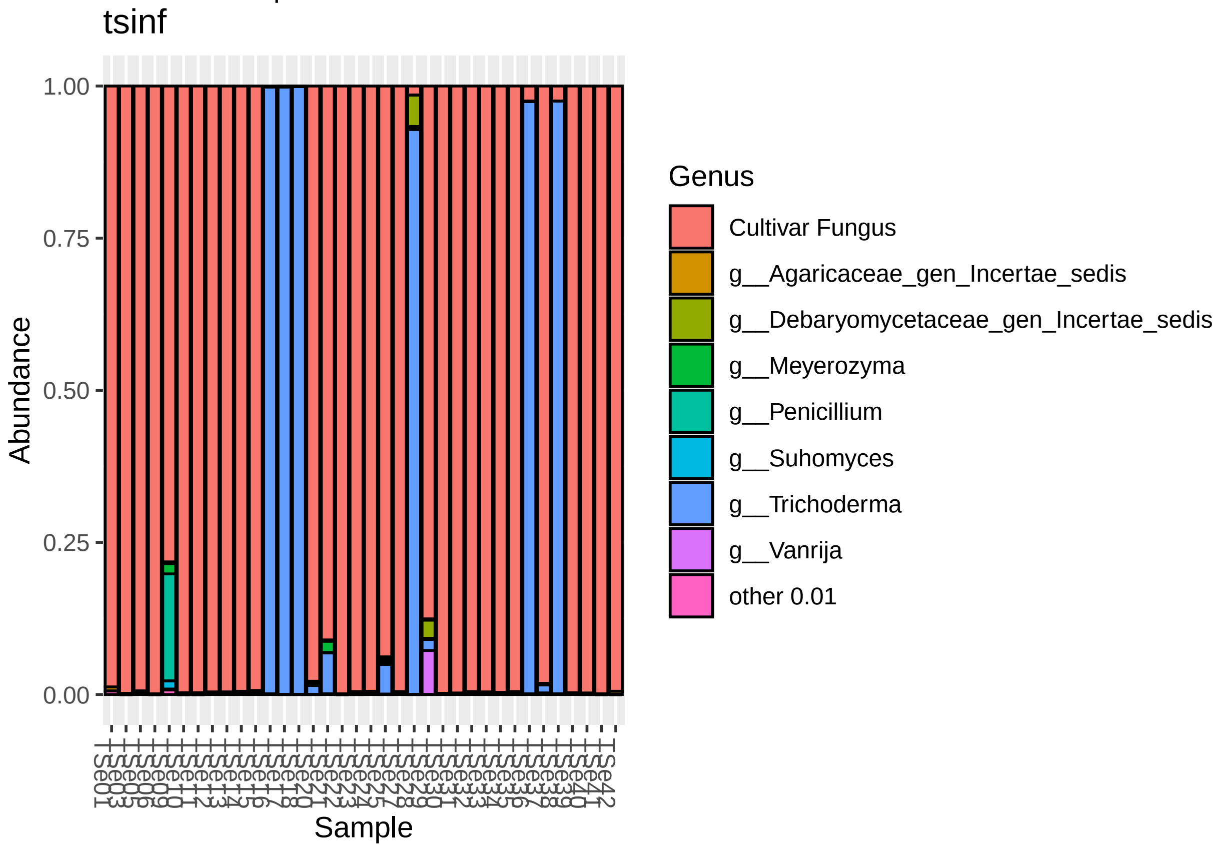
do the raref

Rarefying decon3 to 3000 reads, tsinf to 1700.

```r
# d3 -- raref to 890, 1414, 1558, 3049 ... 3000
# ts -- raref to 1486, 1620, 1724, 1737, 1787 ... 1700
fin.psList.raref =
  list(
    decon3 = rarefy_even_depth(fin.psList$decon3, sample.size=3000, rngseed=1103, replace=F),
    tsinf = rarefy_even_depth(fin.psList$tsinf, sample.size=1700, rngseed=1103, replace=F)
    )
```

```
## `set.seed(1103)` was used to initialize repeatable random subsampling.

## Please record this for your records so others can reproduce.

## Try `set.seed(1103); .Random.seed` for the full vector

## ...

## 14 samples removedbecause they contained fewer reads than `sample.size`.

## Up to first five removed samples are:

## d3e01d3e04d3e07d3e08d3e09

## ...

## `set.seed(1103)` was used to initialize repeatable random subsampling.

## Please record this for your records so others can reproduce.

## Try `set.seed(1103); .Random.seed` for the full vector

## ...

## 6 samples removedbecause they contained fewer reads than `sample.size`.

## Up to first five removed samples are:

## TSe02TSe04TSe07TSe08TSe19

## ...
```

```r
for( ln in names(fin.psList.raref) ) {
  print(
    fin.psList.raref[[ln]] %>%
      make_rel_abund() %>%
      plot_bar(fill="Genus") +
      ggtitle(ln)
  )
}
```


decon3
Abundance
1.00
0.75
0.50
0.25
0.00
Sample
Cultivar Fungus
g__Acaulium
g__Alatosessilispora
g__Arthrographis
g__Aureobasidium
g__Calophoma
g__Candida
g__Ceramothyrium
g__Cladosporium
g__Debaryomyces
g__Diplodia
g__Exobasidium
g__Ganoderma
g__Gymnoascus
g__Malassezia
g__Meyerozyma
g__Monochaetia
g__Nigrospora
g__Paraphoma
g__Peltaster
g__Pilidium
g__Preussia
g__Pseudomicrostroma
g__Ramularia
g__Shiraiaceae_gen_Incertae_sedis
g__Sporocadus
g__Sporormiella
g__Taphrina
g__Trichomerium
other 0.01
unclass. c__Leotiomycetes
unclass. f__Didymellaceae
unclass. f__Dothideaceae
unclass. f__Phaeosphaeriaceae
unclass. f__Trichomeriaceae
unclass. o__Agaricostilbales

tsinf
Abundance
1.00
0.75
0.50
0.25
0.00
Sample
Genus
Cultivar Fungus
g__Agaricaceae_gen_Incertae_sedis
g__Debaryomycetaceae_gen_Incertae_sedis
g__Meyerozyma
g__Penicillium
g__Suhomyces
g__Trichoderma
g__Vanrija
other 0.01

Heatmaps

Data Wrangling

```
## ---- more complicated HEATMAPS ---------------------------------------------------------- ##
## ---- from https://www.biostars.org/p/18211/ --------------------------------------------- ##
source_url(
  "https://raw.githubusercontent.com/obigriffith/biostar-tutorials/master/Heatmaps/heatmap.3.R")
```

```
## i SHA-1 hash of file is "015fc0457e61e3e93a903e69a24d96d2dac7b9fb"
```

```
psList.rab = lapply(fin.psList.raref, make_rel_abund)
psList.melt =
  lapply(psList.rab,
         psmelt )
```

Decon3

```

ps.m = psList.melt$decon3

## change spatial.layer to be different for trash and food samples
ps.m <-
  ps.m %>%
  mutate(
    spatial.layer =
      ifelse(s.type %in% c("trash", "food (leaves)"),
             fg.coord,
             spatial.layer
      )
  )
```

Data wrangling

Color wrangling Check available colors

```
paletteMartin
```

```
##         Black    SherpaBlue  PersianGreen       HotPink   CottonCandy
##     "#000000"     "#004949"     "#009292"     "#ff6db6"     "#ffb6db"
## PigmentIndigo   ScienceBlue    Heliotrope        Malibu    FrenchPass
##     "#490092"     "#006ddb"     "#b66dff"     "#6db6ff"     "#b6dbff"
##       RedBerry         Brown    MangoTango     Harlequin    LaserLemon
##     "#920000"     "#924900"     "#db6d00"     "#24ff24"     "#ffff6d"
```

Assign colors to each sample_variable from colorBlindness::paletteMartin…

```
## ---- decon3 metadata <--> color ---------------------------------------------------------- ##
# basically turning sample_data table into colorcode_table

## list,
##    each item named by sample_var
##    each item is two columm matrix linking each unique variable value to a color
d3.metadat.cols =
  list(s.type =
         as.data.frame(
           cbind(s.type = c("fungus garden","trash","food (leaves)"),
```

```r
               col=c("#ffff6d","#924900","#009292"))), # set colors for s.type
       spat.lay =
         as.data.frame(
           cbind( spatial.layer =
                  c("top","mid1","mid2","bottom",
                    "TT","TM","TB","Le"),
                col =
                  c("#ff6db6","#ffb6db","#b6dbff","#6db6ff",
                    "#924900","#db6d00","#920000","#004949")))#, #  cols for layers
    )

head(d3.metadat.cols)
```

```
## $s.type
##          s.type     col
## 1 fungus garden #ffff6d
## 2         trash #924900
## 3 food (leaves) #009292
##
## $spat.lay
##   spatial.layer     col
## 1           top #ff6db6
## 2          mid1 #ffb6db
## 3          mid2 #b6dbff
## 4        bottom #6db6ff
## 5            TT #924900
## 6            TM #db6d00
## 7            TB #920000
## 8            Le #004949
```

Create massive color code table…

```r
## ---- color code table ------------------------------------------------------------------ ##
## making giant table of s.id (rownames) x sample_vars (columns), filled with colorscodes ?
# joining one column at a time

tmp.j =
  inner_join(
      inner_join(
        ps.m %>% select(s.id, s.type) %>% group_by(s.id) %>% distinct(),
        d3.metadat.cols$s.type
        ) %>% select(s.id, s.type = col), # current table: [s.id, s.type]
      inner_join(
        ps.m %>% select(s.id, spatial.layer) %>% group_by(s.id) %>% distinct(),
        d3.metadat.cols$spat.lay
        ) %>% select(s.id, spatial.lay = col),
      join_by(s.id)
    ) # [s.id, s.type, spatial.lay]
```

```
## Joining with `by = join_by(s.type)`
## Joining with `by = join_by(spatial.layer)`
```

```r
# turn column s.id into rownames
d3.samdat.hm3colorbars =
  tmp.j %>%
  # filter(s.id != "d3e21") %>%
  tibble::column_to_rownames(., var="s.id") %>%
  as.matrix()

## check
d3.samdat.hm3colorbars
```

```
##       s.type    spatial.lay
## d3e18 "#ffff6d" "#ffb6db"
## d3e45 "#ffff6d" "#ffb6db"
## d3e43 "#ffff6d" "#ffb6db"
## d3e32 "#ffff6d" "#b6dbff"
## d3e26 "#924900" "#924900"
## d3e17 "#ffff6d" "#6db6ff"
## d3e40 "#ffff6d" "#ff6db6"
## d3e03 "#ffff6d" "#ff6db6"
## d3e34 "#ffff6d" "#b6dbff"
## d3e06 "#ffff6d" "#ff6db6"
## d3e10 "#924900" "#db6d00"
## d3e14 "#ffff6d" "#ffb6db"
## d3e37 "#ffff6d" "#b6dbff"
## d3e31 "#ffff6d" "#b6dbff"
## d3e27 "#ffff6d" "#6db6ff"
## d3e20 "#ffff6d" "#6db6ff"
## d3e42 "#ffff6d" "#b6dbff"
## d3e28 "#ffff6d" "#6db6ff"
## d3e46 "#ffff6d" "#6db6ff"
## d3e41 "#ffff6d" "#ffb6db"
## d3e05 "#ffff6d" "#6db6ff"
## d3e44 "#ffff6d" "#ff6db6"
## d3e39 "#ffff6d" "#ff6db6"
## d3e02 "#ffff6d" "#6db6ff"
## d3e21 "#ffff6d" "#ff6db6"
## d3e30 "#009292" "#004949"
```

```r
# d3.samdat.hm3colorbars[-which(row.names(d3.samdat.hm3colorbars)=="d3e21"),]
```

Create massive color legend…

```r
## ---- creating the massive color legend for each sample_variable ------------------------- ##

# init legend vectors
leg.names = vector(); leg.fills = vector()

# fill legend vectors
for( i in seq_along(d3.metadat.cols) ) {
    # legend names
    l = unlist(d3.metadat.cols[[i]][1], use.names=F) # vector of colors for each sample_variable
    l = c(names(d3.metadat.cols[[i]][1]), l)
    l = c(l, "")    # i think this is adding blank space at end of each var for viz separation
    leg.names = append(leg.names, l)

    # legend fills
    f = unlist(d3.metadat.cols[[i]][2], use.names=F)
    f = c("white", f, "white") # one white fill is for blanks, the other is for NAs i think?
    leg.fills = append(leg.fills, f)
}
# check
cbind(leg.names, leg.fills)
```

```
##       leg.names       leg.fills
##  [1,] "s.type"        "white"
##  [2,] "fungus garden" "#ffff6d"
##  [3,] "trash"         "#924900"
##  [4,] "food (leaves)" "#009292"
##  [5,] ""              "white"
##  [6,] "spatial.layer" "white"
##  [7,] "top"           "#ff6db6"
##  [8,] "mid1"          "#ffb6db"
##  [9,] "mid2"          "#b6dbff"
## [10,] "bottom"        "#6db6ff"
## [11,] "TT"            "#924900"
## [12,] "TM"            "#db6d00"
## [13,] "TB"            "#920000"
## [14,] "Le"            "#004949"
## [15,] ""              "white"
```

Plotting Can't plot in RMarkdown because "margins are too large". See separate PDFs.

```
## ---- plot ------------------------------------------------------------------------------- ##
save.image()

# ps.m %>%
#   select(s.id, Abundance, Genus) %>%
# # filter(s.id != "d3e21") %>%
#   tidyr::pivot_wider(names_from=Genus, values_from=Abundance) %>%
#   tibble::column_to_rownames("s.id") %>%
#   as.matrix() %>%
#   t() %>% # need samples as rows and Genus as columns for heatmap
#   heatmap.3(
#     trace = "none",
#       col = colorRampPalette(colors=c("white","blue")),
#       breaks = seq(0, 1, length=100),
#       lwid = c(4,8),
#       lhei = c(0.6,2),
#       #cexCol = 0.3,
#       cexRow = 0.5,
#       distfun = function(x) dist(x, method = "euclidean"),
#       hclustfun = function(x) hclust(x, method="ward.D"),
#       scale = "none",
#       ColSideColors = d3.samdat.hm3colorbars,
#          # d3.samdat.hm3colorbars[-which(row.names(d3.samdat.hm3colorbars)=="d3e21"), ],
#       ColSideColorsSize = 1,
#       #margins = c(12,6),
#       key = TRUE,
#       KeyValueName="Rel. Abund."#,
#       #lmat = lmat, lwid = lwid, lhei = lhei
#       #keysize = 0.5
#   ) +
# legend(
#   "left",
#   legend=leg.names,
#   fill=leg.fills,
#   border=F,
#   bty="n",
#   y.intersp = 0.7,
#   cex=0.7,
#   inset=0,
#   xjust=1
# )
```

TSinf

```
ps.m = psList.melt$tsinf
```

Data wrangling. . .

Color wrangling. . . Get color codes from desired palette:

```
paletteMartin
```

```
##         Black    SherpaBlue  PersianGreen       HotPink   CottonCandy
##     "#000000"     "#004949"     "#009292"     "#ff6db6"     "#ffb6db"
## PigmentIndigo   ScienceBlue    Heliotrope        Malibu    FrenchPass
##     "#490092"     "#006ddb"     "#b66dff"     "#6db6ff"     "#b6dbff"
##       RedBerry         Brown    MangoTango     Harlequin    LaserLemon
##     "#920000"     "#924900"     "#db6d00"     "#24ff24"     "#ffff6d"
```

```
## ---- TSinf metadata <--> color ---------------------------------------------------------- ##
# basically turning sample_data table into colorcode_table

## list,
##    each item named by sample_var
##    each item is two columm matrix linking each unique variable value to a color
metadat.cols =
  list(s.type =
         as.data.frame(
           cbind(s.type = c("fungus garden","trash"),
               col=c("#db6d00","#924900"))), # set colors for s.type
       treatment =
         as.data.frame(
           cbind(treatment = c("Trichoderma", "PBS","NT"),
                 col=c("#004949", "#6db6ff","grey"))),
        ants.yn =
         as.data.frame(
           cbind(ants.yn = c("yes", "no"),
                 col=c("#920000","black"))),
       day.post.inf =
         data.frame(day.post.inf = c(0,2,4),
                    col = c("#ffff6d","#24ff24","#009292")),
       expect.inf =
         data.frame(expect.inf = c("yes","no","maybe"),
                    col = c("#004949","#b6dbff", "#009292"))#,
       # exp.rep =
       #   as.data.frame(
       #     cbind(exp.rep = c("r1","r2"),
       #           col = c("#ff6db6", "#490092"))),
       # extract.batch = ps.m %>%
       #   select(extract.batch) %>%
       #   unique() %>%
       #   cbind(., col=sample(paletteMartin, nrow(.), replace=F)), # random cols for extr.batch
       # pcr.batch = ps.m %>%
       #   select(pcr.batch) %>% unique() %>%
       #   cbind(., col=sample(paletteMartin, nrow(.), replace=F)) # random cols for pcr.batch
    )
# check
metadat.cols
```

```
## $s.type
##          s.type     col
## 1 fungus garden #db6d00
## 2         trash #924900
##
## $treatment
##     treatment     col
## 1 Trichoderma #004949
## 2         PBS #6db6ff
## 3          NT    grey
##
## $ants.yn
##   ants.yn     col
## 1     yes #920000
## 2      no   black
##
## $day.post.inf
##   day.post.inf     col
## 1            0 #ffff6d
## 2            2 #24ff24
## 3            4 #009292
##
## $expect.inf
##   expect.inf     col
## 1        yes #004949
## 2         no #b6dbff
## 3      maybe #009292
```

```
## ---- color code table ------------------------------------------------------------------- ##
## making giant table of s.id (rownames) x sample_vars (columns), filled with colorscodes ?
# joining one column at a time

tmp.j =
  full_join(
      left_join(
        ps.m %>% select(s.id, s.type) %>% group_by(s.id) %>% distinct(),
        metadat.cols$s.type
        ) %>% select(s.id, s.type = col), # current table: [s.id, s.type]
      left_join(
        ps.m %>% select(s.id, treatment) %>% group_by(s.id) %>% distinct(),
```

```r
        metadat.cols$treatment
        ) %>% select(s.id, treatment = col),
      join_by(s.id)
    ) # [s.id, s.type, treatment]
```

```
## Joining with `by = join_by(s.type)`
## Joining with `by = join_by(treatment)`
```

```r
tmp.j =
  full_join(
    tmp.j,
    left_join(
      ps.m %>% select(s.id, day.post.inf) %>% group_by(s.id) %>% distinct(),
      metadat.cols$day.post.inf
      ) %>% select(s.id, day.post.inf = col),
    join_by(s.id)
    ) # [s.id, s.type, treatment, day.post.inf]
```

```
## Joining with `by = join_by(day.post.inf)`
```

```r
tmp.j =
  full_join(
    tmp.j,
    left_join(
      ps.m %>% select(s.id, ants.yn) %>% group_by(s.id) %>% distinct(),
        metadat.cols$ants.yn
        ) %>% select(s.id, ants.yn = col),
    join_by(s.id)
    ) # [s.id, s.type, treatment, day.post.inf, ants.yn]
```

```
## Joining with `by = join_by(ants.yn)`
```

```r
tmp.j =
  full_join(
    tmp.j,
    left_join(
      ps.m %>% select(s.id, expect.inf) %>% group_by(s.id) %>% distinct(),
      metadat.cols$expect.inf
      ) %>% select(s.id, expect.inf = col),
    join_by(s.id)
    ) # [s.id, s.type, treatment, day.post.inf, ants.yn, expect.inf]
```

```
## Joining with `by = join_by(expect.inf)`
```

```r
# tmp.j =
#   full_join(
#     tmp.j,
#     left_join(
#       ps.m %>% select(s.id, exp.rep) %>% group_by(s.id) %>% distinct(),
#       metadat.cols$exp.rep
#       ) %>% select(s.id, exp.rep = col),
#     join_by(s.id)
#     ) # [s.id, s.type, treatment, day.post.inf, ants.yn, expect.inf, exp.rep]
# tmp.j =
#   full_join(
#     tmp.j,
#     left_join(
#       ps.m %>% select(s.id, extract.batch) %>% group_by(s.id) %>% distinct(),
#       metadat.cols$extract.batch
#       ) %>% select(s.id, extract.batch = col),
#     join_by(s.id)
#     ) # [s.id,s.type,treatment,day.post.inf,ants.yn,expect.inf,exp.rep,extract.batch]
# tmp.j =
#   full_join(
#     tmp.j,
#     left_join(
#       ps.m %>% select(s.id, pcr.batch) %>% group_by(s.id) %>% distinct(),
#       metadat.cols$pcr.batch
#       ) %>% select(s.id, pcr.batch = col),
#     join_by(s.id)
#     ) # [s.id,s.type,treatment,day.post.inf,ants.yn,expect.inf,exp.rep,extract.batch,pcr.batch]

# turn column s.id into rownames
samdat.hm3colorbars =
  tmp.j %>%
  tibble::column_to_rownames(., var="s.id") %>%
  as.matrix()

## check
samdat.hm3colorbars
```

```
##       s.type    treatment day.post.inf ants.yn   expect.inf
## TSe18 "#db6d00" "#004949" "#009292"    "black"   "#004949"
## TSe41 "#db6d00" "#6db6ff" "#ffff6d"    "#920000" "#b6dbff"
## TSe22 "#db6d00" "#6db6ff" "#ffff6d"    "black"   "#b6dbff"
## TSe06 "#db6d00" "grey"    "#009292"    "black"   "#b6dbff"
## TSe30 "#db6d00" "#6db6ff" "#009292"    "#920000" "#b6dbff"
## TSe03 "#db6d00" "#6db6ff" "#24ff24"    "black"   "#b6dbff"
## TSe17 "#db6d00" "#004949" "#009292"    "black"   "#004949"
## TSe31 "#db6d00" "#004949" "#24ff24"    "#920000" "#b6dbff"
## TSe40 "#db6d00" "#6db6ff" "#ffff6d"    "black"   "#b6dbff"
## TSe10 "#db6d00" "#6db6ff" "#24ff24"    "black"   "#b6dbff"
## TSe39 "#db6d00" "#6db6ff" "#009292"    "black"   "#b6dbff"
## TSe11 "#db6d00" "#6db6ff" "#24ff24"    "#920000" "#b6dbff"
## TSe16 "#924900" "#004949" "#009292"    "#920000" "#004949"
## TSe34 "#db6d00" "grey"    "#ffff6d"    "#920000" "#b6dbff"
## TSe33 "#db6d00" "#004949" "#24ff24"    "#920000" "#b6dbff"
## TSe12 "#db6d00" "grey"    "#24ff24"    "#920000" "#b6dbff"
## TSe13 "#db6d00" "grey"    "#ffff6d"    "black"   "#b6dbff"
## TSe35 "#db6d00" "grey"    "#24ff24"    "black"   "#009292"
## TSe32 "#db6d00" "#004949" "#ffff6d"    "#920000" "#b6dbff"
## TSe27 "#db6d00" "grey"    "#24ff24"    "black"   "#b6dbff"
## TSe23 "#db6d00" "#004949" "#ffff6d"    "black"   "#b6dbff"
## TSe14 "#db6d00" "grey"    "#24ff24"    "#920000" "#b6dbff"
## TSe24 "#db6d00" "#6db6ff" "#009292"    "black"   "#b6dbff"
## TSe42 "#db6d00" "#004949" "#009292"    "#920000" "#b6dbff"
## TSe05 "#db6d00" "#004949" "#009292"    "#920000" "#b6dbff"
## TSe15 "#db6d00" "#004949" "#ffff6d"    "black"   "#b6dbff"
## TSe01 "#db6d00" "#6db6ff" "#ffff6d"    "#920000" "#b6dbff"
## TSe37 "#db6d00" "grey"    "#ffff6d"    "black"   "#b6dbff"
## TSe20 "#db6d00" "grey"    "#009292"    "#920000" "#b6dbff"
## TSe38 "#db6d00" "grey"    "#009292"    "black"   "#004949"
## TSe36 "#db6d00" "#004949" "#24ff24"    "black"   "#004949"
## TSe25 "#924900" "#6db6ff" "#009292"    "#920000" "#009292"
## TSe28 "#db6d00" "#004949" "#24ff24"    "black"   "#004949"
## TSe21 "#924900" "#6db6ff" "#009292"    "#920000" "#009292"
## TSe29 "#924900" "grey"    "#009292"    "#920000" "#009292"
## TSe09 "#924900" "grey"    "#009292"    "#920000" "#009292"
```

```
## ---- creating the massive color legend for each sample_variable ------------------------- ##

# init legend vectors
leg.names = vector(); leg.fills = vector()

# fill legend vectors
for( i in seq_along(metadat.cols) ) {
    # legend names
    l = unlist(metadat.cols[[i]][1], use.names=F) # vector of colors for each sample_variable
    l = c(names(metadat.cols[[i]][1]), l)
    l = c(l, "")    # i think this is adding blank space at end of each var for viz separation
    leg.names = append(leg.names, l)

    # legend fills
    f = unlist(metadat.cols[[i]][2], use.names=F)
    f = c("white", f, "white") # one white fill is for blanks, the other is for NAs i think?
    leg.fills = append(leg.fills, f)
}
# check
cbind(leg.names, leg.fills)
```

```
##       leg.names       leg.fills
##  [1,] "s.type"        "white"
##  [2,] "fungus garden" "#db6d00"
##  [3,] "trash"         "#924900"
##  [4,] ""              "white"
##  [5,] "treatment"     "white"
##  [6,] "Trichoderma"   "#004949"
##  [7,] "PBS"           "#6db6ff"
##  [8,] "NT"            "grey"
##  [9,] ""              "white"
## [10,] "ants.yn"       "white"
## [11,] "yes"           "#920000"
## [12,] "no"            "black"
## [13,] ""              "white"
## [14,] "day.post.inf"  "white"
## [15,] "0"             "#ffff6d"
## [16,] "2"             "#24ff24"
## [17,] "4"             "#009292"
## [18,] ""              "white"
## [19,] "expect.inf"    "white"
## [20,] "yes"           "#004949"
## [21,] "no"            "#b6dbff"
## [22,] "maybe"         "#009292"
## [23,] ""              "white"
```

Plotting. . . Can't plot in RMarkdown because "margins are too large". See separate PDFs.

```
save.image()
## ---- plot ------------------------------------------------------------------------------ ##
# ps.m %>%
#   select(s.id, Abundance, Genus) %>%
#   tidyr::pivot_wider(names_from=Genus, values_from=Abundance) %>%
#   tibble::column_to_rownames("s.id") %>%
#   as.matrix() %>%
#   t() %>% # need samples as rows and Genus as columns for heatmap
#   heatmap.3(
#     trace = "none",
#       col = colorRampPalette(colors=c("white","blue")),
#       breaks = seq(0, 1, length=100),
#       lwid = c(4,8),
#       lhei = c(0.6,2),
#       #cexCol = 0.3,
#       cexRow = 0.5,
#       distfun = function(x) dist(x, method = "euclidean"),
#       hclustfun = function(x) hclust(x, method="ward.D"),
#       scale = "none",
#       ColSideColors = samdat.hm3colorbars,
#       ColSideColorsSize = 2,
#       #margins = c(12,6),
#       key = TRUE,
#       KeyValueName="Rel. Abund."#,
#       #lmat = lmat, lwid = lwid, lhei = lhei
#       #keysize = 0.5
#   ) +
#   legend(
#     "left",
#     legend=leg.names,
#     fill=leg.fills,
#     border=F,
#     bty="n",
#     y.intersp = 0.7,
#     cex=0.7,
#     inset=0,
#     xjust=1
#   )
```

Alpha Diversity

decon3

```
ps = psList.filt$decon3
spatial.layer.col.ls =
  list("top"="#ff6db6", "mid1"="#ffb6db","mid2"="#b6dbff","bottom"="#6db6ff")
```

```
p <-
  ps %>%
  plot_richness(., x = "s.type",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_violin(
  ) +
  ggtitle("Decon3 (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

  ) +
  stat_summary(
    fun.y= median,
    geom="point",
    shape=23,
    size=2
  )
```

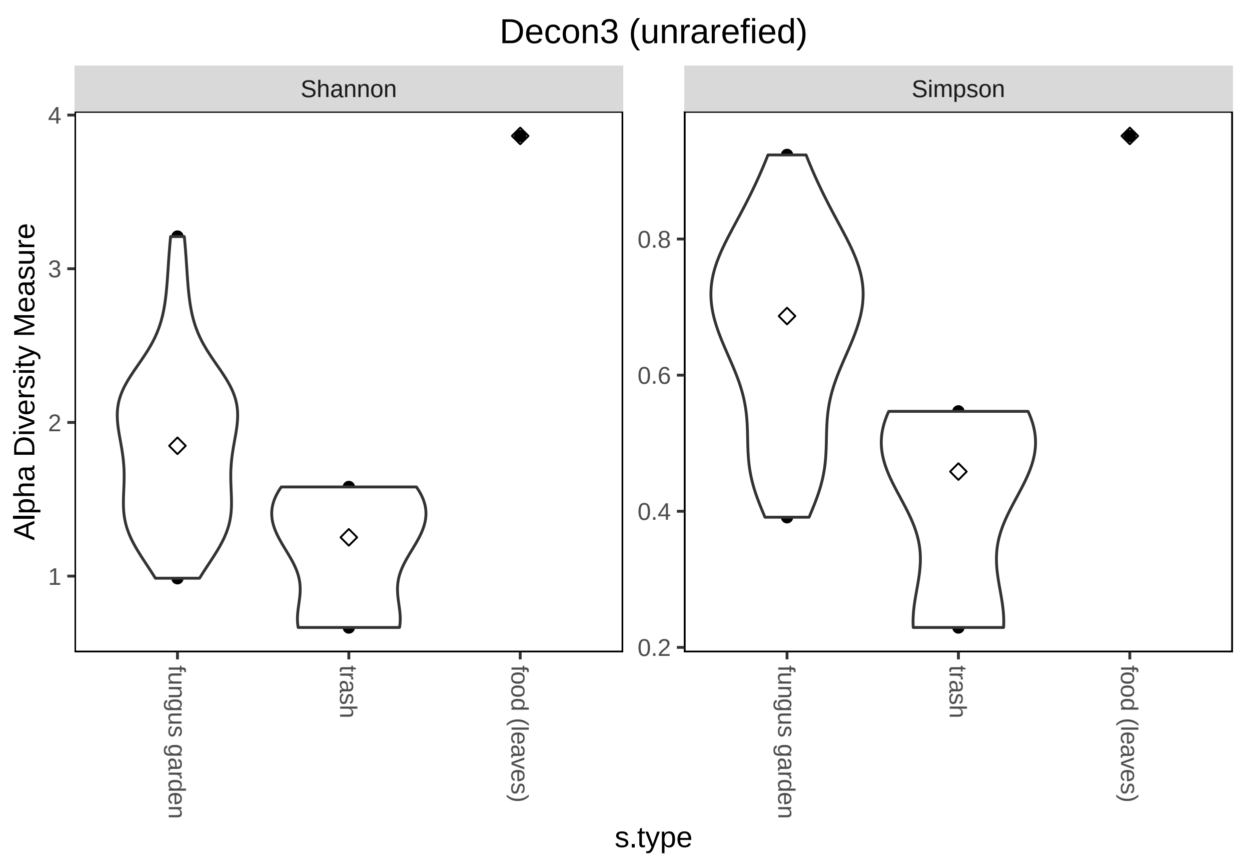
violin-s.type

```
## Warning: The `fun.y` argument of `stat_summary()` is deprecated as of ggplot2 3.3.0.
## i Please use the `fun` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
p$data$s.type = factor(p$data$s.type, levels=c("fungus garden","trash","food (leaves)"))
p
```

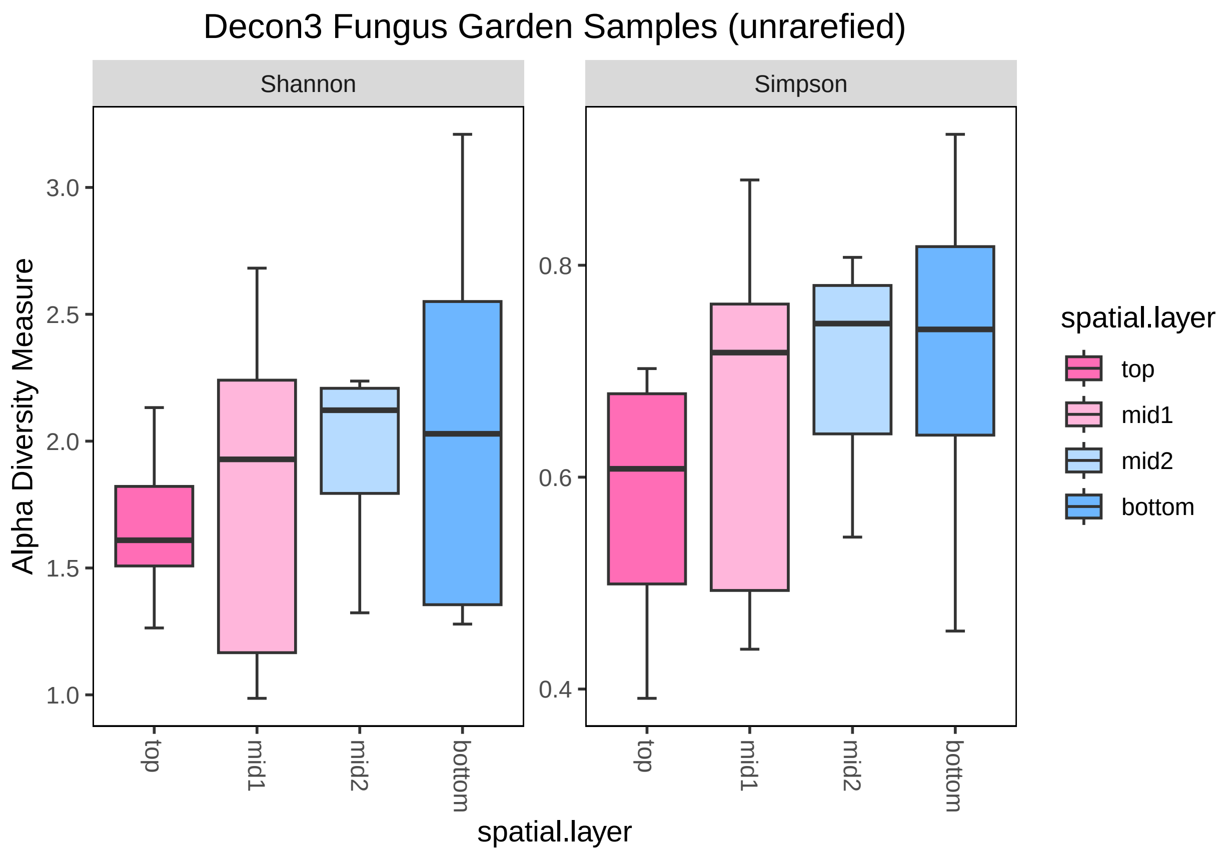
```
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
```

```
## Warning: Groups with fewer than two datapoints have been dropped.
## i Set `drop = FALSE` to consider such groups for position adjustment purposes.
```



```
p <-
  ps %>%
  subset_samples(s.type == "fungus garden") %>%
  plot_richness(., x = "spatial.layer",
               measures = c("Shannon", "Simpson"),
  ) +
  geom_boxplot(
    aes(fill = spatial.layer),
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25
  ) +
  scale_fill_manual(
    values = unlist(spatial.layer.col.ls)
  ) +
  ggtitle("Decon3 Fungus Garden Samples (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

  )
p$data$spatial.layer = factor(p$data$spatial.layer, levels=c("top","mid1","mid2","bottom"))
p$layers = p$layers[-1]
```

```
p
```



violin-layer

```
#Create dataframe of alpha diversity values for anova stats
ps.adiv.df = estimate_richness(ps, measures = c("Shannon", "Simpson"))
ps.adiv.df$s.type = sample_data(ps)$s.type
ps.adiv.df$spatial.layer = sample_data(ps)$spatial.layer

#Run anova on both Shannon and Simpson for s.type
s.type.aov = list(shan = anova(aov(Shannon ~ s.type, ps.adiv.df %>% filter(s.type != "food (leaves)"))),
                  sim = anova(aov(Simpson ~ s.type, ps.adiv.df %>% filter(s.type != "food (leaves)"))))
s.type.aov
```

anova

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##           Df Sum Sq Mean Sq F value  Pr(>F)
## s.type     1 1.3261 1.32614  4.9285 0.03263 *
## Residuals 37 9.9558 0.26908
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $sim
## Analysis of Variance Table
```

```
##
## Response: Simpson
##           Df  Sum Sq  Mean Sq F value   Pr(>F)
## s.type     1 0.18201 0.182009  9.0022 0.004805 **
## Residuals 37 0.74808 0.020218
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Run anova on both Shannon and Simpson for spatial.layer
ps.adiv.df.fg = ps.adiv.df %>% filter(s.type == "fungus garden")
layer.aov = list(shan = anova(aov(Shannon ~ spatial.layer, ps.adiv.df.fg)),
                 sim = anova(aov(Simpson ~ spatial.layer, ps.adiv.df.fg)))
layer.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##               Df Sum Sq Mean Sq F value Pr(>F)
## spatial.layer  3 0.8041 0.26802  0.9833 0.4129
## Residuals     32 8.7226 0.27258
##
## $sim
## Analysis of Variance Table
##
## Response: Simpson
##               Df  Sum Sq  Mean Sq F value Pr(>F)
## spatial.layer  3 0.11319 0.037729  2.0774 0.1228
## Residuals     32 0.58118 0.018162
```

```
TukeyHSD(aov(Shannon ~ s.type, ps.adiv.df %>% filter(s.type != "food (leaves)")))
```

TukeyHSD

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Shannon ~ s.type, data = ps.adiv.df %>% filter(s.type != "food (leaves)"))
##
## $s.type
##                             diff      lwr         upr     p adj
## trash-fungus garden -0.6920146 -1.32361 -0.06041886 0.0326257
```

```
TukeyHSD(aov(Simpson ~ s.type, ps.adiv.df %>% filter(s.type != "food (leaves)")))
```

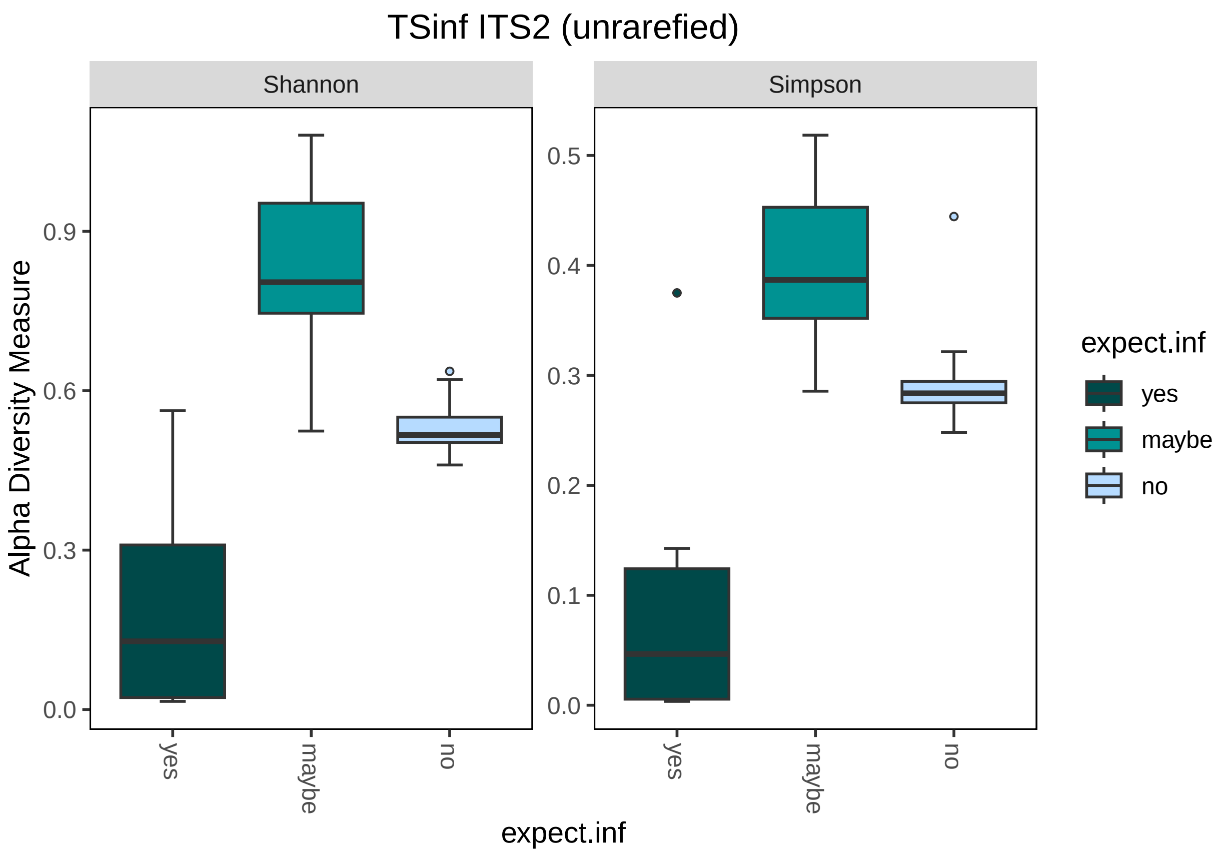
```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Simpson ~ s.type, data = ps.adiv.df %>% filter(s.type != "food (leaves)"))
##
## $s.type
##                             diff        lwr        upr     p adj
## trash-fungus garden -0.2563698 -0.4295001 -0.0832396 0.0048051
```

TSinf

```r
ps = psList.filt$tsinf

# colors
exp.inf.col.ls = list("yes" = "#004949",
                      "maybe" = "#009292",
                      "no"= "#b6dbff")
treatment.col.ls = list("Trichoderma" = "#004949",
                        "PBS" = "#6db6ff",
                        "NT" ="grey")
ants.yn.col.ls = list("yes" = "#920000", "no" = "black")
s.type.col.ls = list("fungus garden" = "#db6d00",
                      "trash" = "#924900" )
```

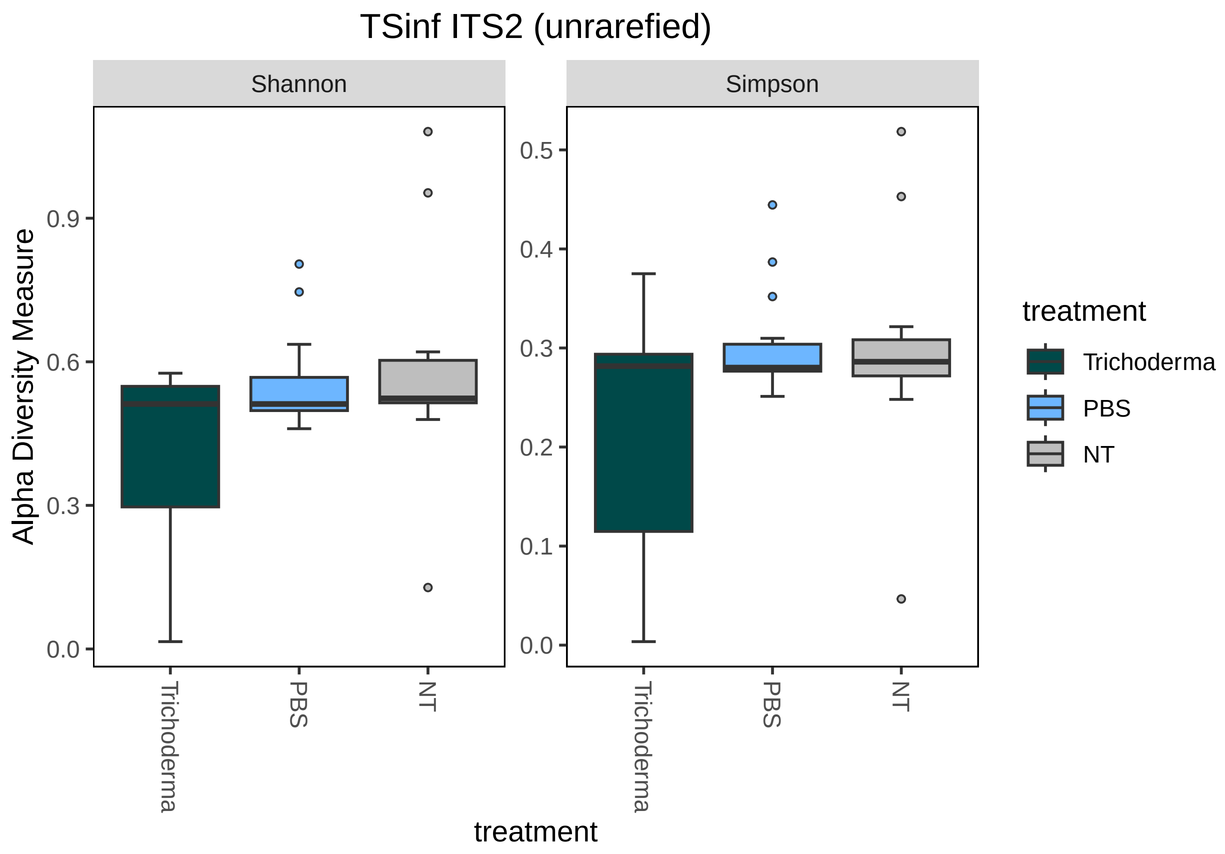
boxplot #####expect.inf

```r
p <-
  ps %>%
  plot_richness(., x = "expect.inf",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_boxplot(
    aes(fill = expect.inf),
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25

  ) +
  scale_fill_manual(
    values = unlist(exp.inf.col.ls)
  ) +
  ggtitle("TSinf ITS2 (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

  )
p$data$expect.inf = factor(p$data$expect.inf, levels=c("yes","maybe", "no"))
p$layers = p$layers[-1]
p
```



```r
p <-
  ps %>%
  plot_richness(., x = "treatment",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_boxplot(
    aes(fill = treatment),
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25

  ) +
  scale_fill_manual(
    values = unlist(treatment.col.ls)
  ) +
  ggtitle("TSinf ITS2 (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

  )
p$data$treatment = factor(p$data$treatment, levels=c("Trichoderma","PBS", "NT"))
p$layers = p$layers[-1]
```

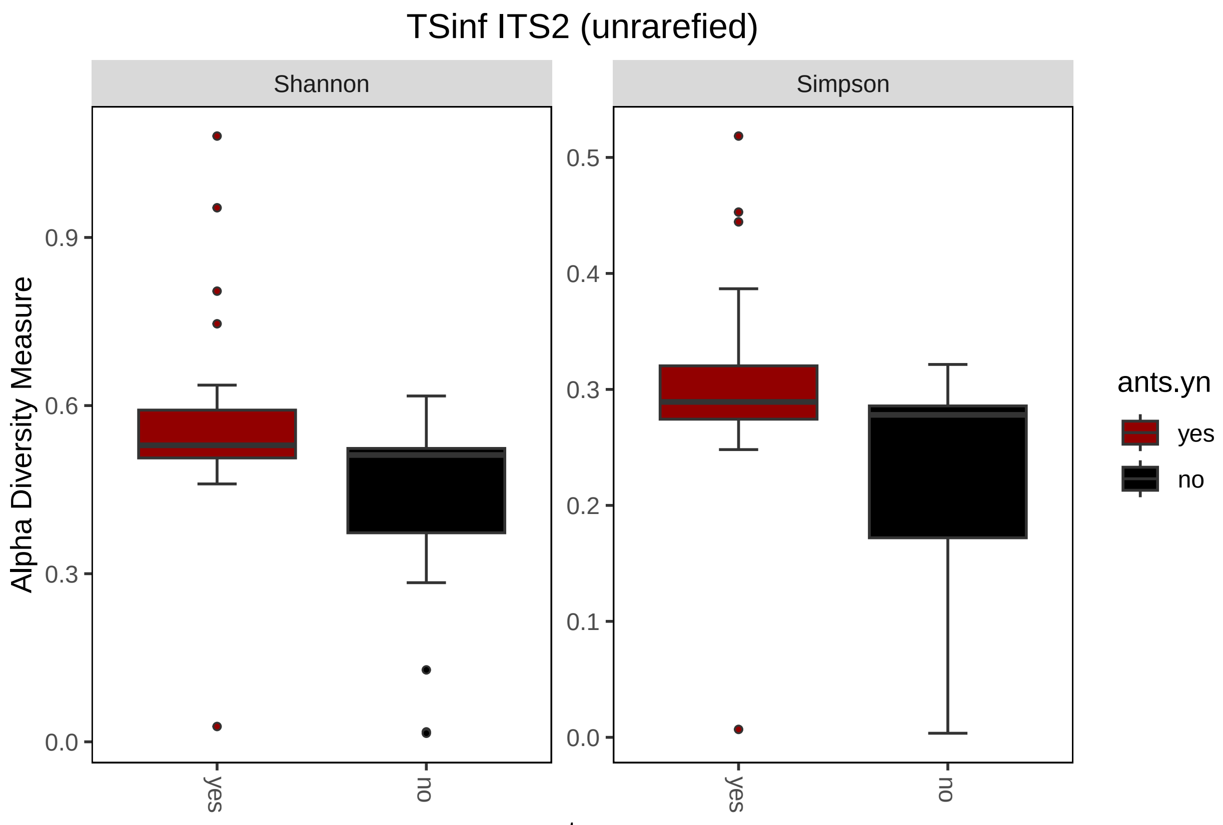
```
p
```



treat

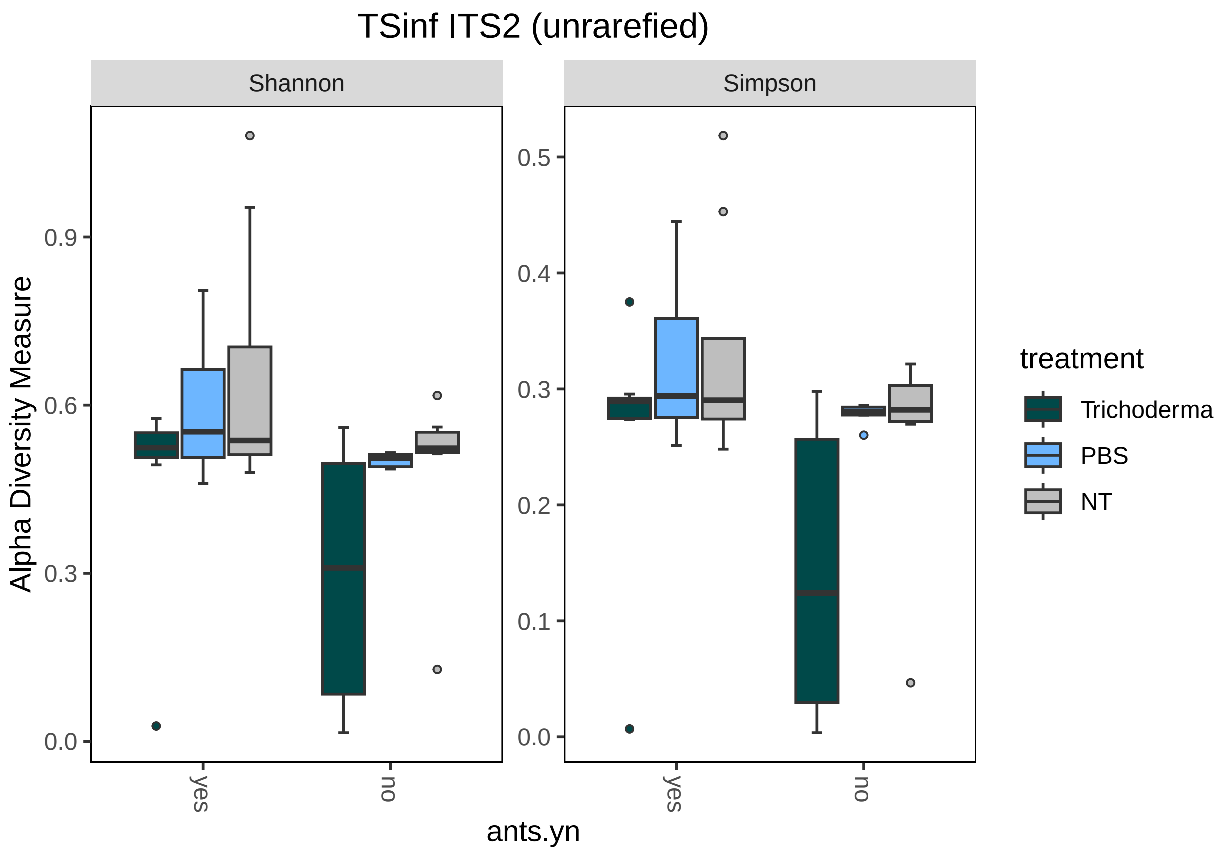
```
p <-
  ps %>%
  plot_richness(., x = "ants.yn",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_boxplot(
    aes(fill = ants.yn),
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25

  ) +
  scale_fill_manual(
    values = unlist(ants.yn.col.ls)
  ) +
  ggtitle("TSinf ITS2 (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

  )
p$data$ants.yn = factor(p$data$ants.yn, levels=c("yes","no"))
```

```
p$layers = p$layers[-1]
p
```



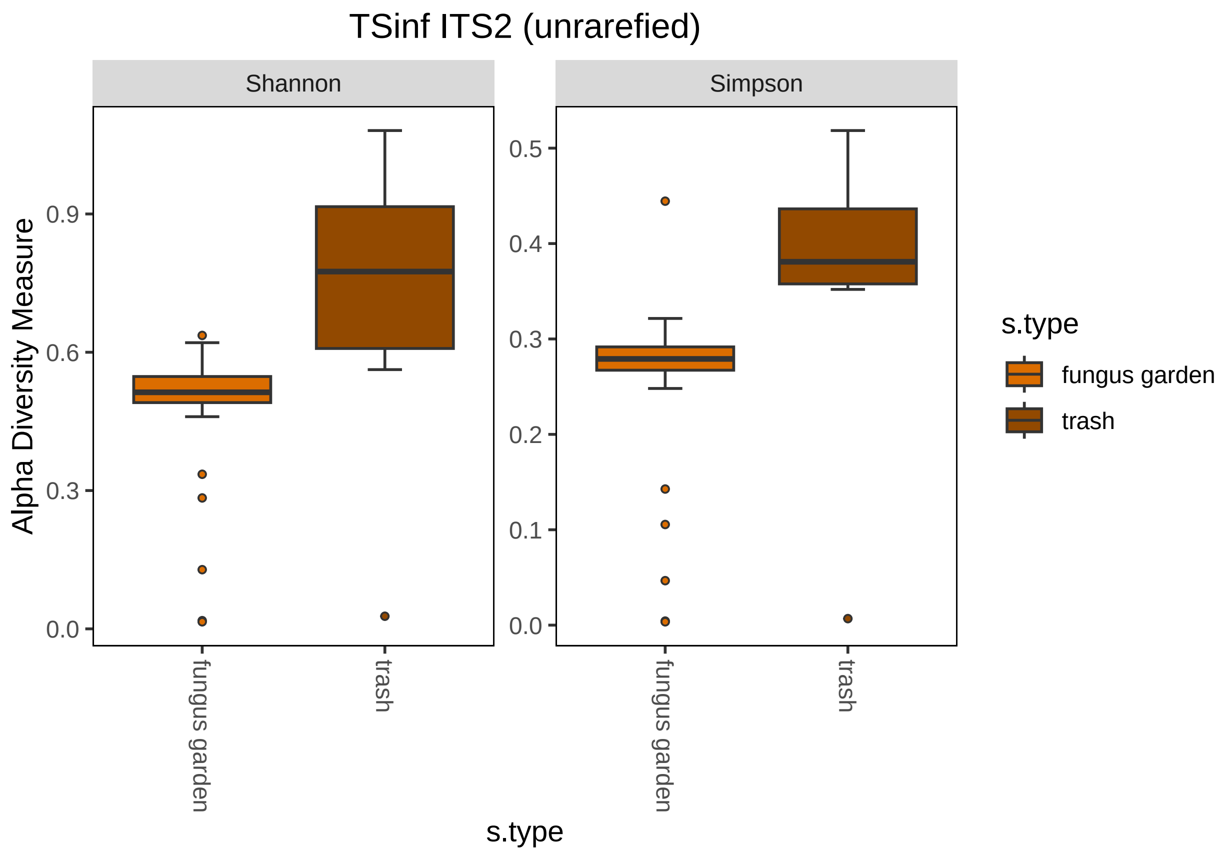
+/- ants

```
# ants.yn.col.ls
# treatment.col.ls

p <-
  ps %>%
  plot_richness(., x = "ants.yn",
                measures = c("Shannon", "Simpson")
  ) +
  geom_boxplot(
    aes(fill = treatment),
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25

  ) +
  scale_fill_manual(
    values = unlist(treatment.col.ls)
  ) +
  ggtitle("TSinf ITS2 (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

  )
p$data$ants.yn = factor(p$data$ants.yn, levels=c("yes","no"))
p$data$treatment = factor(p$data$treatment, levels=c("Trichoderma","PBS", "NT"))
p$layers = p$layers[-1]
p
```



trt:ants

```
p <-
  ps %>%
  plot_richness(., x = "s.type",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_boxplot(
    aes(fill = s.type),
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25

  ) +
  scale_fill_manual(
    values = unlist(s.type.col.ls)
  ) +
  ggtitle("TSinf ITS2 (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

  )
p$data$ants.yn = factor(p$data$ants.yn, levels=c("yes","no"))
p$data$treatment = factor(p$data$treatment, levels=c("Trichoderma","PBS", "NT"))
p$data$s.type = factor(p$data$s.type, levels=c("fungus garden","trash"))
p$layers = p$layers[-1]
p
```



s.type

```
#Create dataframe of alpha diversity values for anova stats
ps.adiv.df = estimate_richness(ps, measures = c("Shannon", "Simpson"))
ps.adiv.df$s.type = sample_data(ps)$s.type
ps.adiv.df$expect.inf = sample_data(ps)$expect.inf
ps.adiv.df$treatment = sample_data(ps)$treatment
ps.adiv.df$ants.yn = sample_data(ps)$ants.yn
ps.adiv.df$day.post.inf = sample_data(ps)$day.post.inf

#Run anova on both Shannon and Simpson for s.type
s.type.aov = list(shan = anova(aov(Shannon ~ s.type, ps.adiv.df)),
                  sim = anova(aov(Simpson ~ s.type, ps.adiv.df)))
s.type.aov
```

anova

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##           Df  Sum Sq  Mean Sq F value  Pr(>F)
## s.type     1 0.24372 0.243725  6.7863 0.01284 *
## Residuals 40 1.43657 0.035914
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $sim
## Analysis of Variance Table
##
## Response: Simpson
##           Df  Sum Sq  Mean Sq F value  Pr(>F)
## s.type     1 0.04337 0.043372  4.0231 0.05168 .
## Residuals 40 0.43123 0.010781
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(aov(Shannon ~ s.type, ps.adiv.df))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Shannon ~ s.type, data = ps.adiv.df)
##
## $s.type
##                           diff        lwr       upr     p adj
## trash-fungus garden 0.2176945 0.04880058 0.3865885 0.0128359
```

```
TukeyHSD(aov(Simpson ~ s.type, ps.adiv.df))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Simpson ~ s.type, data = ps.adiv.df)
##
## $s.type
##                            diff           lwr       upr     p adj
## trash-fungus garden 0.09183383 -0.0007010846 0.1843687 0.0516782
```

```
#Run anova on both Shannon and Simpson for spatial.layer
expect.inf.aov = list(shan = anova(aov(Shannon ~ expect.inf, ps.adiv.df)),
                      sim = anova(aov(Simpson ~ expect.inf, ps.adiv.df)))
expect.inf.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##            Df  Sum Sq Mean Sq F value    Pr(>F)
## expect.inf  2 1.18887 0.59443  47.175 3.877e-11 ***
## Residuals  39 0.49143 0.01260
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $sim
## Analysis of Variance Table
##
## Response: Simpson
##            Df  Sum Sq  Mean Sq F value    Pr(>F)
## expect.inf  2 0.30136 0.150682  33.922 2.918e-09 ***
## Residuals  39 0.17324 0.004442
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(aov(Shannon ~ expect.inf, ps.adiv.df))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Shannon ~ expect.inf, data = ps.adiv.df)
##
## $expect.inf
##                 diff        lwr        upr    p adj
## no-maybe  -0.2916084 -0.4237131 -0.1595037 1.11e-05
## yes-maybe -0.6258903 -0.7860254 -0.4657551 0.00e+00
## yes-no    -0.3342819 -0.4490764 -0.2194873 0.00e+00
```

```
TukeyHSD(aov(Simpson ~ expect.inf, ps.adiv.df))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Simpson ~ expect.inf, data = ps.adiv.df)
##
## $expect.inf
##                 diff        lwr         upr     p adj
## no-maybe  -0.1105733 -0.1890086 -0.03213805 0.0039737
## yes-maybe -0.3013680 -0.3964459 -0.20629003 0.0000000
## yes-no    -0.1907946 -0.2589522 -0.12263701 0.0000001
```

```
#Run anova on both Shannon and Simpson for spatial.layer
treatment.aov = list(shan = anova(aov(Shannon ~ treatment, ps.adiv.df)),
                     sim = anova(aov(Simpson ~ treatment, ps.adiv.df)))
treatment.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##           Df  Sum Sq  Mean Sq F value  Pr(>F)
## treatment  2 0.27791 0.138955  3.8643 0.02944 *
## Residuals 39 1.40239 0.035959
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $sim
## Analysis of Variance Table
##
```

```
## Response: Simpson
##           Df  Sum Sq  Mean Sq F value  Pr(>F)
## treatment  2 0.07509 0.037543  3.6649 0.03479 *
## Residuals 39 0.39952 0.010244
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(aov(Shannon ~ treatment, ps.adiv.df))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Shannon ~ treatment, data = ps.adiv.df)
##
## $treatment
##                        diff        lwr          upr     p adj
## PBS-NT          -0.02237413 -0.1969905  0.152242217 0.9477712
## Trichoderma-NT  -0.18265293 -0.3572693 -0.008036588 0.0386517
## Trichoderma-PBS -0.16027881 -0.3348952  0.014337540 0.0775582
```

```
TukeyHSD(aov(Simpson ~ treatment, ps.adiv.df))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Simpson ~ treatment, data = ps.adiv.df)
##
## $treatment
##                         diff         lwr          upr     p adj
## PBS-NT           0.006087548 -0.08711308 0.0992881733 0.9861415
## Trichoderma-NT  -0.086494731 -0.17969536 0.0067058944 0.0735660
## Trichoderma-PBS -0.092582279 -0.18578290 0.0006183461 0.0518563
```

```
#Run anova on both Shannon and Simpson for spatial.layer
ants.yn.aov = list(shan = anova(aov(Shannon ~ ants.yn, ps.adiv.df)),
                   sim = anova(aov(Simpson ~ ants.yn, ps.adiv.df)))
ants.yn.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##           Df  Sum Sq  Mean Sq F value  Pr(>F)
## ants.yn    1 0.22601 0.226015  6.2165 0.01689 *
## Residuals 40 1.45428 0.036357
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $sim
## Analysis of Variance Table
##
## Response: Simpson
##           Df  Sum Sq  Mean Sq F value  Pr(>F)
## ants.yn    1 0.06718 0.067182  6.5959 0.01406 *
## Residuals 40 0.40742 0.010186
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(aov(Shannon ~ ants.yn, ps.adiv.df))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Shannon ~ ants.yn, data = ps.adiv.df)
##
## $ants.yn
##             diff        lwr      upr     p adj
## yes-no 0.1482351 0.02807515 0.268395 0.0168934
```

```
TukeyHSD(aov(Simpson ~ ants.yn, ps.adiv.df))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Simpson ~ ants.yn, data = ps.adiv.df)
##
## $ants.yn
##              diff        lwr       upr     p adj
## yes-no 0.08081848 0.01721848 0.1444185 0.0140613
```

```
#Run anova on both Shannon and Simpson for spatial.layer
day.post.inf.aov = list(shan = anova(aov(Shannon ~ day.post.inf, ps.adiv.df)),
                        sim = anova(aov(Simpson ~ day.post.inf, ps.adiv.df)))
day.post.inf.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##              Df  Sum Sq  Mean Sq F value Pr(>F)
## day.post.inf  1 0.00385 0.003851  0.0919 0.7634
## Residuals    40 1.67645 0.041911
##
## $sim
## Analysis of Variance Table
##
## Response: Simpson
##              Df  Sum Sq   Mean Sq F value Pr(>F)
## day.post.inf  1 0.00226 0.0022589  0.1913 0.6642
## Residuals    40 0.47234 0.0118086
```

```
trtXants.aov = list(shan = anova(aov(Shannon ~ treatment*ants.yn, ps.adiv.df)),
                    sim = anova(aov(Simpson ~ treatment*ants.yn, ps.adiv.df)))
trtXants.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##                   Df  Sum Sq  Mean Sq F value  Pr(>F)
## treatment          2 0.27791 0.138955  4.3098 0.02099 *
## ants.yn            1 0.22601 0.226015  7.0101 0.01195 *
## treatment:ants.yn  2 0.01568 0.007839  0.2431 0.78543
## Residuals         36 1.16069 0.032242
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $sim
## Analysis of Variance Table
##
## Response: Simpson
##                    Df  Sum Sq  Mean Sq F value   Pr(>F)
## treatment           2 0.07509 0.037543  4.1958 0.023016 *
## ants.yn             1 0.06718 0.067182  7.5083 0.009496 **
## treatment:ants.yn  2 0.01021 0.005107  0.5708 0.570120
## Residuals          36 0.32212 0.008948
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(aov(Shannon ~ treatment*ants.yn, ps.adiv.df))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Shannon ~ treatment * ants.yn, data = ps.adiv.df)
##
## $treatment
##                       diff        lwr          upr     p adj
## PBS-NT          -0.02237413 -0.1882612  0.143512986 0.9419454
## Trichoderma-NT  -0.18265293 -0.3485400 -0.016765819 0.0282410
## Trichoderma-PBS -0.16027881 -0.3261659  0.005608309 0.0600367
##
## $ants.yn
##             diff        lwr       upr     p adj
## yes-no 0.1482351 0.03468734 0.2617828 0.0119524
##
## $`treatment:ants.yn`
##                                      diff          lwr        upr     p adj
## PBS:no-NT:no                  0.024270797 -0.287623801 0.33616539 0.9998948
## Trichoderma:no-NT:no         -0.183988874 -0.495883472 0.12790572 0.4939768
## NT:yes-NT:no                  0.174665330 -0.117085351 0.46641601 0.4777413
## PBS:yes-NT:no                 0.117307509 -0.174443173 0.40905819 0.8292761
## Trichoderma:yes-NT:no        -0.006985648 -0.298736329 0.28476503 0.9999997
## Trichoderma:no-PBS:no        -0.208259671 -0.520154268 0.10363493 0.3573838
## NT:yes-PBS:no                 0.150394534 -0.141356148 0.44214521 0.6347632
## PBS:yes-PBS:no                0.093036712 -0.198713969 0.38478739 0.9276248
## Trichoderma:yes-PBS:no       -0.031256445 -0.323007126 0.26049424 0.9994954
## NT:yes-Trichoderma:no         0.358654204  0.066903523 0.65040489 0.0086891
## PBS:yes-Trichoderma:no        0.301296383  0.009545701 0.59304706 0.0395563
## Trichoderma:yes-Trichoderma:no  0.177003226 -0.114747455 0.46875391 0.4630562
## PBS:yes-NT:yes               -0.057357822 -0.327466466 0.21275082 0.9871968
## Trichoderma:yes-NT:yes       -0.181650978 -0.451759623 0.08845767 0.3496551
## Trichoderma:yes-PBS:yes      -0.124293157 -0.394401801 0.14581549 0.7358017
```

```
TukeyHSD(aov(Simpson ~ treatment*ants.yn, ps.adiv.df))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
```

```
##
## Fit: aov(formula = Simpson ~ treatment * ants.yn, data = ps.adiv.df)
##
## $treatment
##                        diff         lwr           upr     p adj
## PBS-NT           0.006087548 -0.08130275  0.0934778441 0.9841514
## Trichoderma-NT  -0.086494731 -0.17388503  0.0008955652 0.0528746
## Trichoderma-PBS -0.092582279 -0.17997257 -0.0051919831 0.0358912
##
## $ants.yn
##              diff        lwr       upr     p adj
## yes-no 0.08081848 0.02100087 0.1406361 0.0094965
##
## $`treatment:ants.yn`
##                                       diff         lwr        upr     p adj
## PBS:no-NT:no                   0.026314425 -0.13799346 0.19062231 0.9965217
## Trichoderma:no-NT:no          -0.110327392 -0.27463528 0.05398049 0.3513321
## NT:yes-NT:no                   0.078715108 -0.07498085 0.23241106 0.6410560
## PBS:yes-NT:no                  0.069632500 -0.08406345 0.22332845 0.7480914
## Trichoderma:yes-NT:no          0.010094874 -0.14360108 0.16379083 0.9999545
## Trichoderma:no-PBS:no         -0.136641817 -0.30094970 0.02766607 0.1504367
## NT:yes-PBS:no                  0.052400684 -0.10129527 0.20609664 0.9061692
## PBS:yes-PBS:no                 0.043318075 -0.11037788 0.19701403 0.9561561
## Trichoderma:yes-PBS:no        -0.016219550 -0.16991550 0.13747640 0.9995312
## NT:yes-Trichoderma:no          0.189042501  0.03534655 0.34273846 0.0086427
## PBS:yes-Trichoderma:no         0.179959892  0.02626394 0.33365585 0.0138700
## Trichoderma:yes-Trichoderma:no 0.120422266 -0.03327369 0.27411822 0.1985769
## PBS:yes-NT:yes                -0.009082609 -0.15137741 0.13321219 0.9999605
## Trichoderma:yes-NT:yes        -0.068620234 -0.21091504 0.07367457 0.6964559
## Trichoderma:yes-PBS:yes       -0.059537625 -0.20183243 0.08275718 0.8047494
```

Beta Diversity

decon3

d3e21 doesnt appear to be an outlier ITS2-wise…

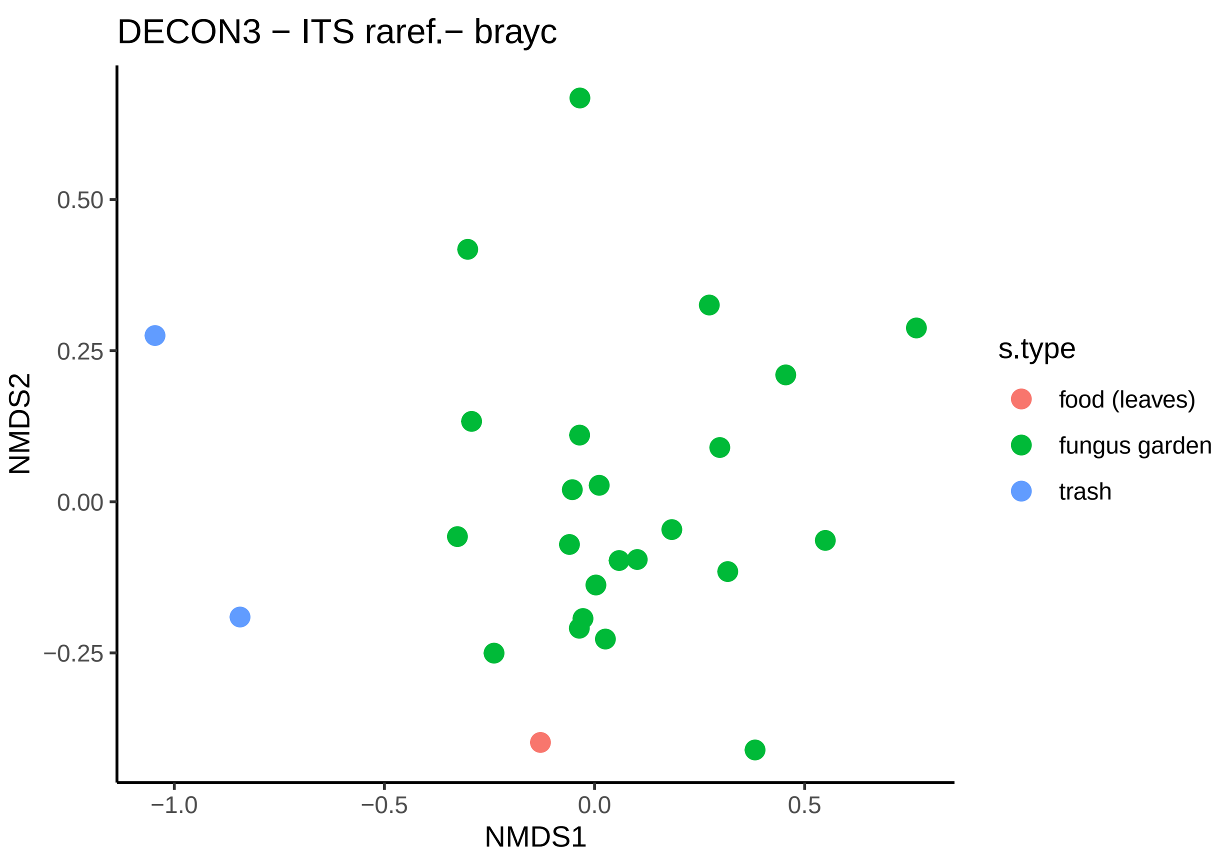
```
ps = fin.psList.raref$decon3
ps.samdat.df <- data.frame(sample_data(ps))
# fungus gardens only
ps.fg = ps %>% subset_samples(s.type == "fungus garden")
ps.fg.samdat.df = ps.samdat.df %>% filter(s.type == "fungus garden")
```

```

# Bray Curtis NMDS
ps.bcord = ordinate(ps, method= "NMDS", distance = "bray")
```

d3BC

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.1615957
## Run 1 stress 0.1655642
## Run 2 stress 0.1807695
## Run 3 stress 0.1599383
## ... New best solution
## ... Procrustes: rmse 0.09280419  max resid 0.3205441
## Run 4 stress 0.1836768
## Run 5 stress 0.1644775
## Run 6 stress 0.1629474
## Run 7 stress 0.1615957
## Run 8 stress 0.1674883
## Run 9 stress 0.2083205
## Run 10 stress 0.1629527
## Run 11 stress 0.1572082
## ... New best solution
## ... Procrustes: rmse 0.03455708  max resid 0.1343299
## Run 12 stress 0.1748189
## Run 13 stress 0.1568646
## ... New best solution
## ... Procrustes: rmse 0.008616926  max resid 0.03297576
## Run 14 stress 0.1828154
## Run 15 stress 0.1579025
## Run 16 stress 0.1629474
## Run 17 stress 0.1803467
## Run 18 stress 0.1615957
## Run 19 stress 0.1676544
## Run 20 stress 0.1750435
## *** Best solution was not repeated -- monoMDS stopping criteria:
##      1: no. of iterations >= maxit
##     18: stress ratio > sratmax
##      1: scale factor of the gradient < sfgrmin
```

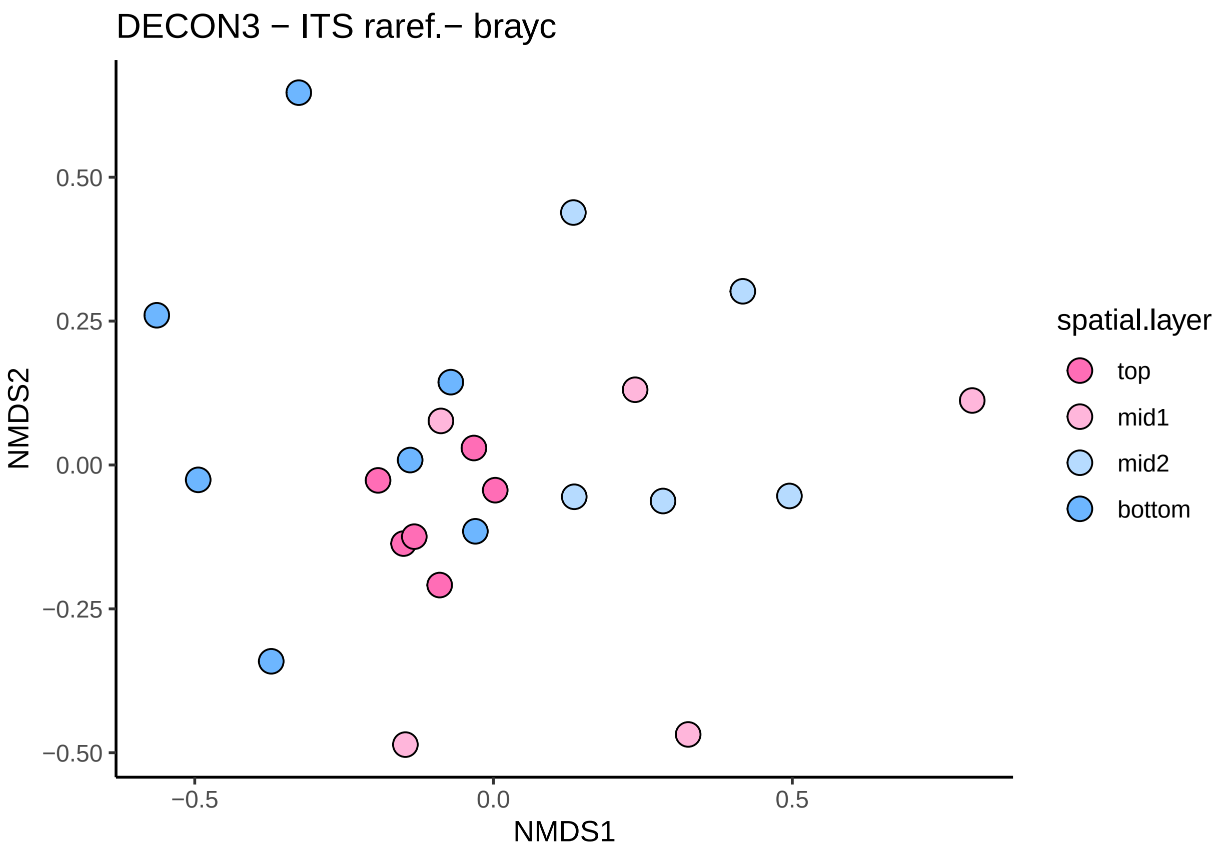
```
plot_ordination( ps,
                 ps.bcord,
                 color = "s.type",
                 # shape = "s.type"
                 ) +
  geom_point(size=3) +
  ggtitle("DECON3 - ITS raref.- brayc") +
  theme_classic()
```



```
## fungus garden samples only
# Bray Curtis NMDS
ps.fg.bcord = ordinate(ps.fg, method= "NMDS", distance = "bray")
```

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.1569728
## Run 1 stress 0.1843442
## Run 2 stress 0.1522857
## ... New best solution
## ... Procrustes: rmse 0.03726305  max resid 0.1097632
## Run 3 stress 0.1761945
## Run 4 stress 0.1569728
## Run 5 stress 0.1545552
## Run 6 stress 0.2036766
## Run 7 stress 0.1975732
## Run 8 stress 0.176198
## Run 9 stress 0.2516895
## Run 10 stress 0.1732829
## Run 11 stress 0.1569728
## Run 12 stress 0.1774534
## Run 13 stress 0.1522857
## ... New best solution
## ... Procrustes: rmse 2.2721e-05  max resid 7.881818e-05
## ... Similar to previous best
## Run 14 stress 0.1886761
## Run 15 stress 0.176169
## Run 16 stress 0.156982
## Run 17 stress 0.1545552
## Run 18 stress 0.1997154
## Run 19 stress 0.1800001
## Run 20 stress 0.1736402
## *** Best solution repeated 1 times
```

```
p <-
  plot_ordination( ps.fg,
                 ps.fg.bcord,
                 # color = "spatial.layer",
                 # label = "s.id"
                 ) +
  geom_point(
    aes(fill = spatial.layer),
    shape =21,
    color = "black",
    size=4
  ) +
  scale_fill_manual(
    values = unlist(spatial.layer.col.ls)
  ) +
  ggtitle("DECON3 - ITS raref.- brayc") +
  theme_classic()
p$data$spatial.layer = factor(p$data$spatial.layer, levels = c("top", "mid1", "mid2", "bottom"))
p
```



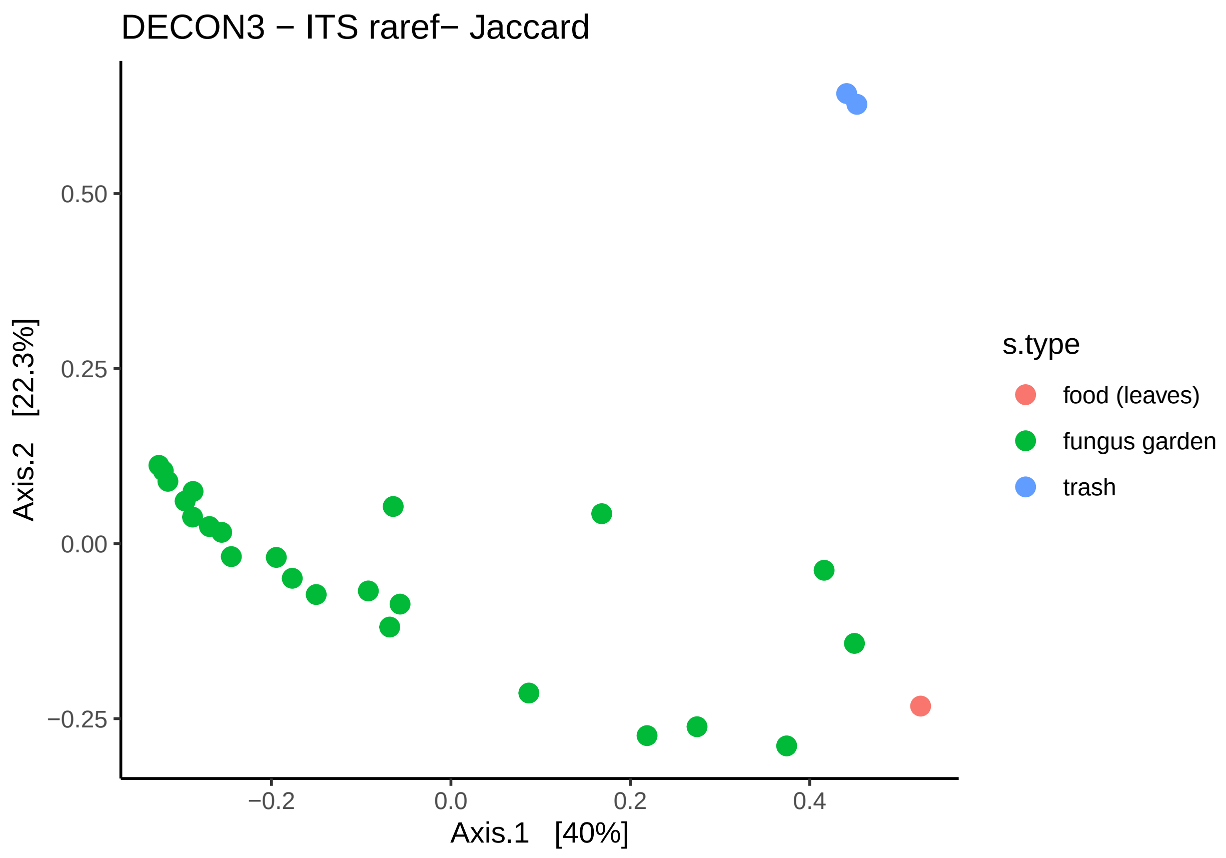
```r
# Permanova
ps.bcdist =
  distance(
    ps %>% subset_samples(s.type != "food (leaves)"),
    method ="bray")

adonis2(ps.bcdist ~ s.type, data = ps.samdat.df %>% filter(s.type != "food (leaves)"))
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.bcdist ~ s.type, data = ps.samdat.df %>% filter(s.type != "food (leaves)"))
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1   1.3983 0.39094 14.763  0.002 **
## Residual 23   2.1784 0.60906
## Total    24   3.5767 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Permanova - FG only
ps.fg.bcdist = distance(ps.fg, method ="bray")
adonis2( ps.fg.bcdist ~ spatial.layer,
         data = ps.fg.samdat.df
)
```

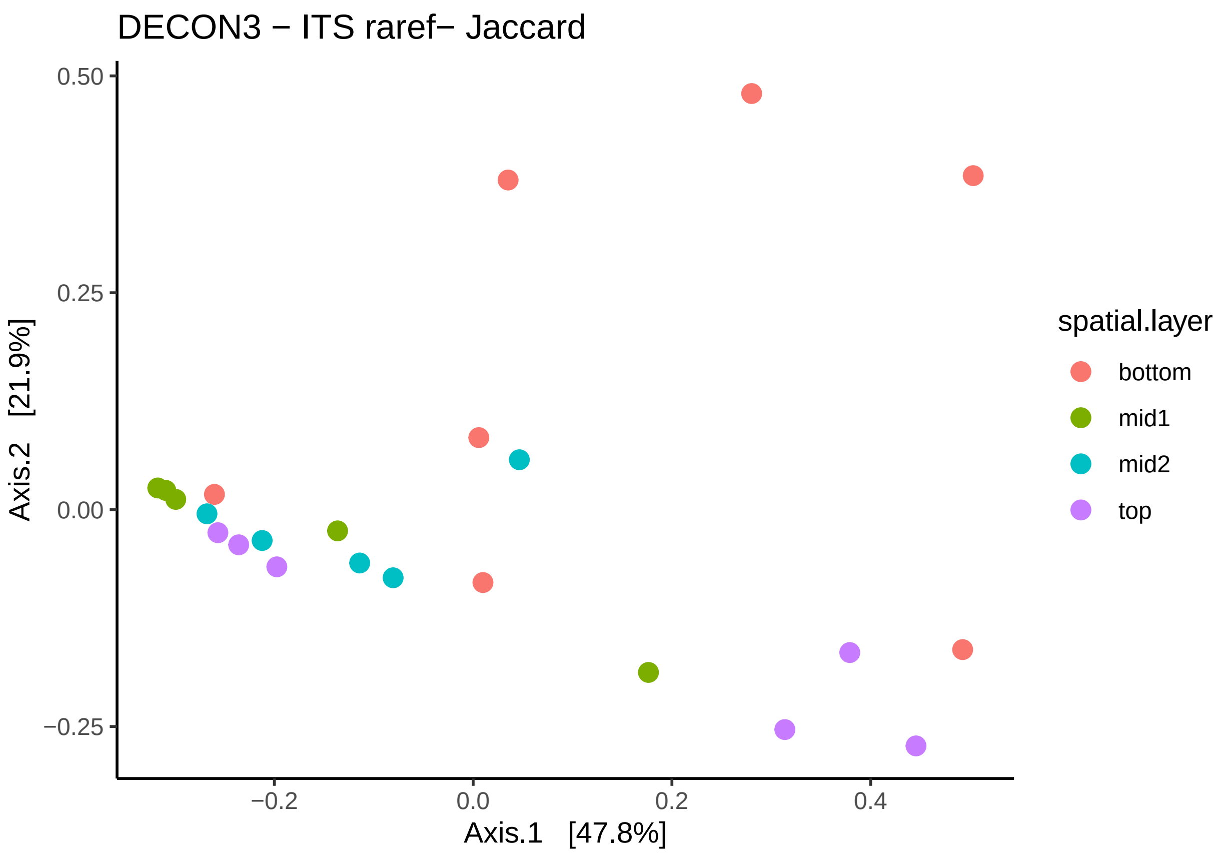
```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.fg.bcdist ~ spatial.layer, data = ps.fg.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     3  0.62282 0.28741 2.5545  0.034 *
## Residual 19  1.54417 0.71259
## Total    22  2.16699 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r

# calculate PCOA using Phyloseq package
ps.jccord = ordinate(ps, "PCoA", "jaccard")

plot_ordination(ps, ps.jccord, color = "s.type") +
  geom_point(size = 3) +
  ggtitle("DECON3 - ITS raref- Jaccard") +
  theme_classic()
```



d3Jacc

```
## fungus garden samples only
# Bray Curtis NMDS
ps.fg.jccord = ordinate(ps.fg, method= "PCoA", distance = "jaccard")
plot_ordination( ps.fg,
                 ps.fg.jccord,
                 color = "spatial.layer",
                 # label = "s.id"
                 ) +
  geom_point(size=3) +
  ggtitle("DECON3 - ITS raref- Jaccard") +
  theme_classic()
```



```r
# Permanova
ps.jccdist =
  distance(ps %>% subset_samples(s.type != "food (leaves)"), method ="jaccard")
adonis2(ps.jccdist ~ s.type, data = ps.samdat.df %>% filter(s.type != "food (leaves)"))
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.jccdist ~ s.type, data = ps.samdat.df %>% filter(s.type != "food (leaves)"))
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1   1.3829 0.27599 8.7677  0.005 **
## Residual 23   3.6277 0.72401
## Total    24   5.0106 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Permanova - FG only
ps.fg.jccdist = distance(ps.fg, method ="jaccard")
adonis2( ps.fg.jccdist ~ spatial.layer,
         data = ps.fg.samdat.df
)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.fg.jccdist ~ spatial.layer, data = ps.fg.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)  
## Model     3   0.8940 0.24882 2.0978  0.038 *
## Residual 19   2.6991 0.75118                
## Total    22   3.5932 1.00000                
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

TSinf

```
ps = fin.psList.raref$tsinf
ps.samdat.df <- data.frame(sample_data(ps))
```

TSinfBC

```

# Bray Curtis NMDS
ps.bcord = ordinate(ps, method= "NMDS", distance = "bray")
```

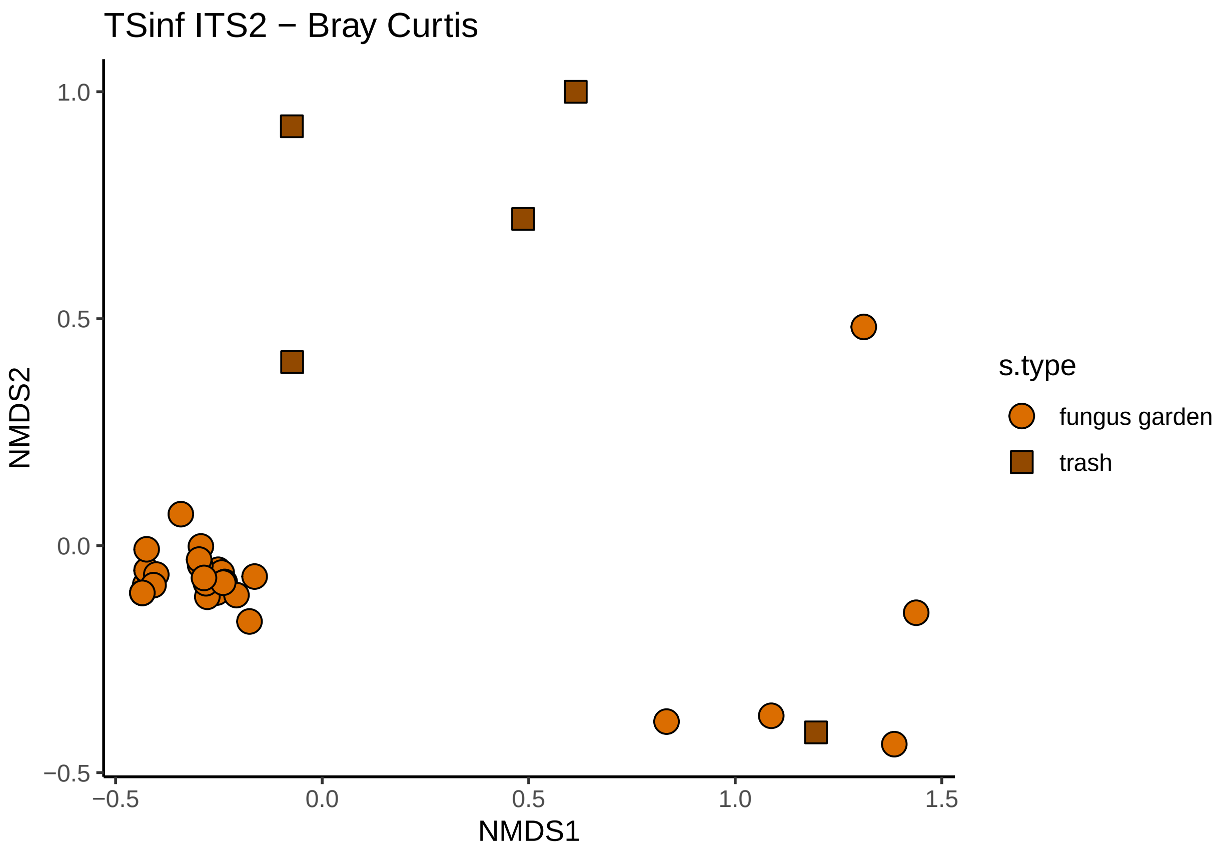
ord.s.type

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.067611 
## Run 1 stress 0.09289 
## Run 2 stress 0.09439933 
## Run 3 stress 0.08816931 
## Run 4 stress 0.0672771 
## ... New best solution
## ... Procrustes: rmse 0.00713883  max resid 0.0289319 
## Run 5 stress 0.07766994 
## Run 6 stress 0.08046065 
## Run 7 stress 0.0688424 
## Run 8 stress 0.08030722 
## Run 9 stress 0.08088235 
## Run 10 stress 0.0958091 
## Run 11 stress 0.09281064 
## Run 12 stress 0.09289001 
## Run 13 stress 0.08019417 
## Run 14 stress 0.08019421 
## Run 15 stress 0.06772943 
## ... Procrustes: rmse 0.007755164  max resid 0.03204925 
## Run 16 stress 0.06882329 
## Run 17 stress 0.09289 
## Run 18 stress 0.08115421 
## Run 19 stress 0.09164378 
## Run 20 stress 0.08019417 
## *** Best solution was not repeated -- monoMDS stopping criteria:
##     16: stress ratio > sratmax
##      4: scale factor of the gradient < sfgrmin
```

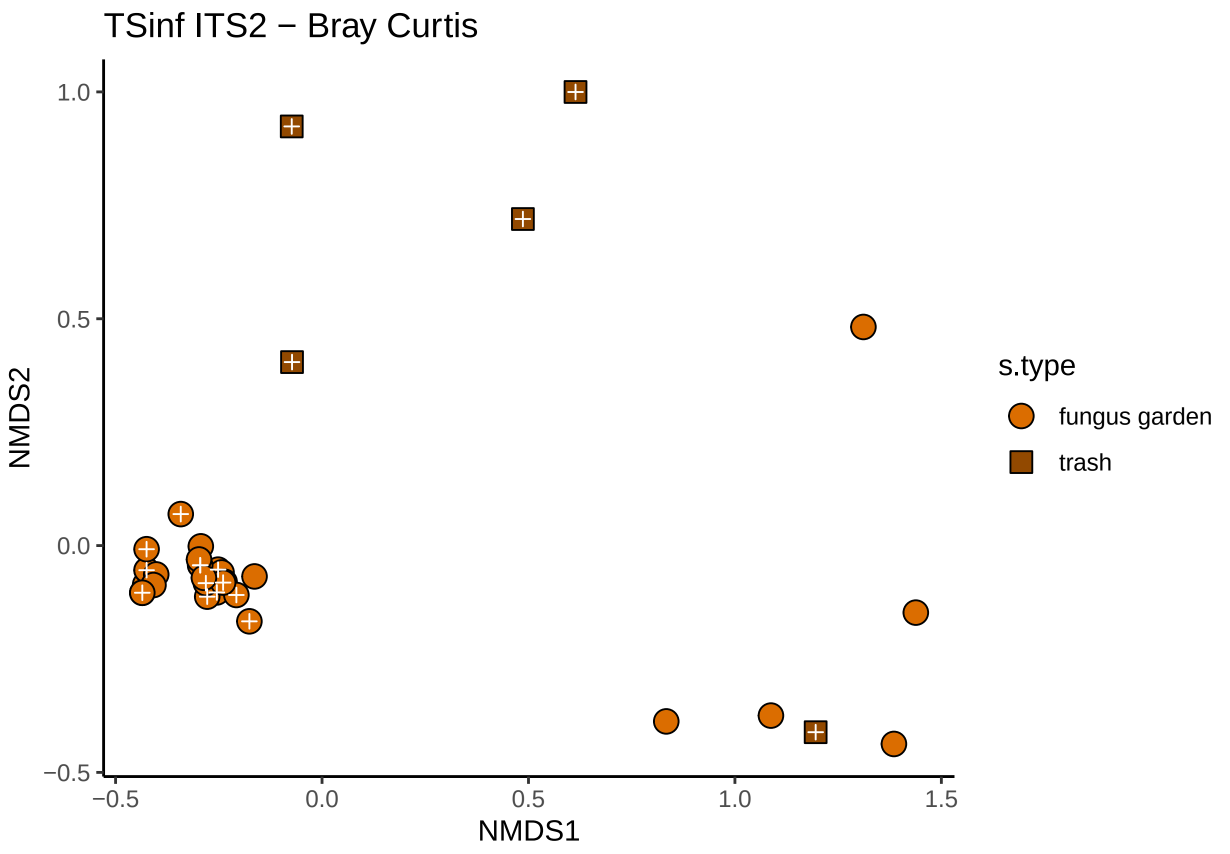
```
p <-
  plot_ordination( ps,
                  ps.bcord,
                  color = "black",
                  shape = "s.type"
  ) +
  geom_point(
    aes(fill = s.type),
    size = 4,
    # shape = 21
  ) +
  ggtitle("TSinf ITS2 - Bray Curtis") +
  scale_fill_manual(
    values = s.type.col.ls
  ) +
  scale_shape_manual(
    values = c(21, 22, 24)
  ) +
  theme_classic() +
  guides(fill = guide_legend(override.aes = list(shape = c(21,22))))
```

```
## Warning in plot_ordination(ps, ps.bcord, color = "black", shape = "s.type"):
## Color variable was not found in the available data you provided.No color
## mapped.
```

```
# order legend
p$data$s.type = factor(p$data$s.type, levels = c("fungus garden", "trash"))
p$data$treatment = factor(p$data$treatment, levels = c("Trichoderma", "PBS", "NT"))
p
```



```
# plot +ants on top
ants.y.bcord =
  p$data %>%
  filter(ants.yn == "yes")
p + geom_point(
  data = ants.y.bcord,
  aes(x=NMDS1, y=NMDS2),
  shape = 3,
  color = "white"
  # stroke = 1.1
)
```



```
# jaccard PCoA
ps.jccord = ordinate(ps, method= "PCoA", distance = "jaccard")
```

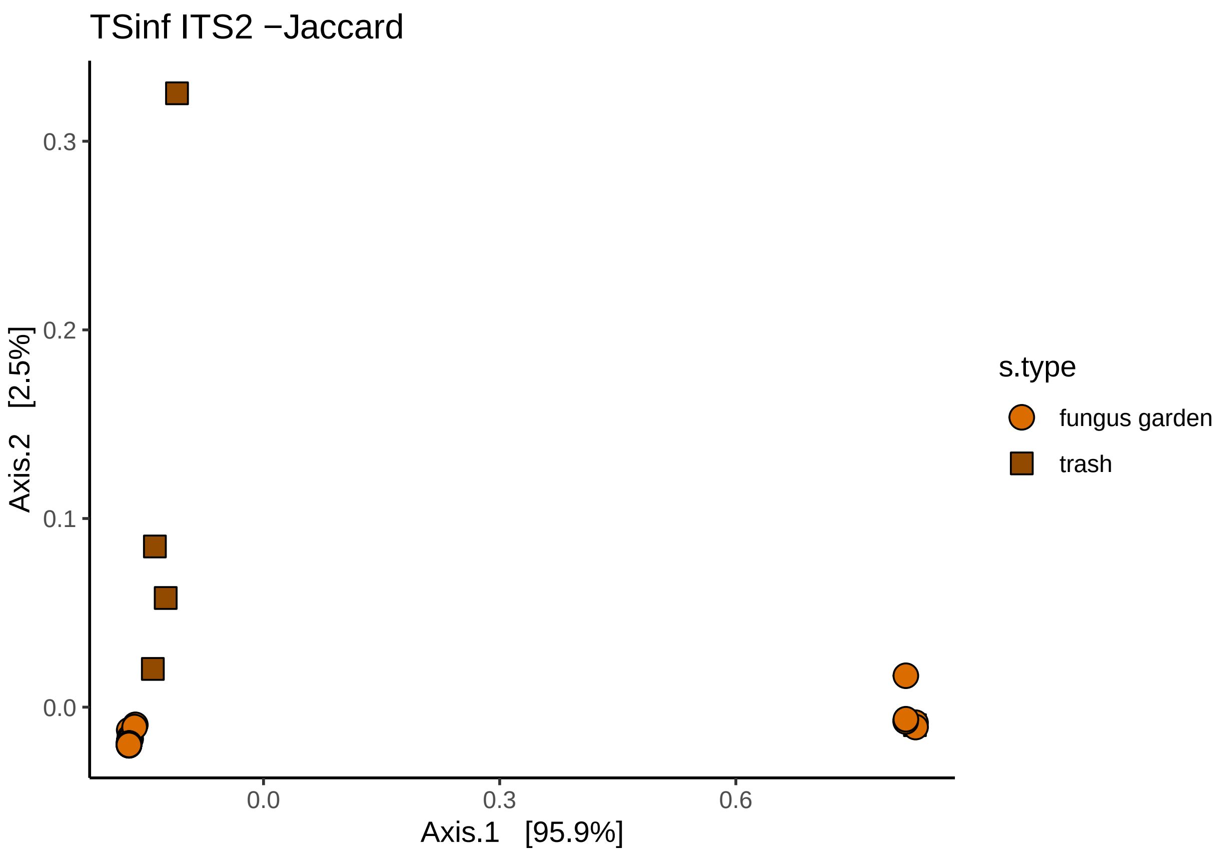
TSinfJCC

```
p <-
  plot_ordination( ps,
                   ps.jccord,
                   color = "black",
                   shape = "s.type"
  ) +
  geom_point(
    aes(fill = s.type),
    size = 4,
    # shape = 21
  ) +
  ggtitle("TSinf ITS2 -Jaccard") +
  scale_fill_manual(
    values = s.type.col.ls
  ) +
  scale_shape_manual(
    values = c(21, 22, 24)
  ) +
  theme_classic() +
  guides(fill = guide_legend(override.aes = list(shape = c(21,22))))
```

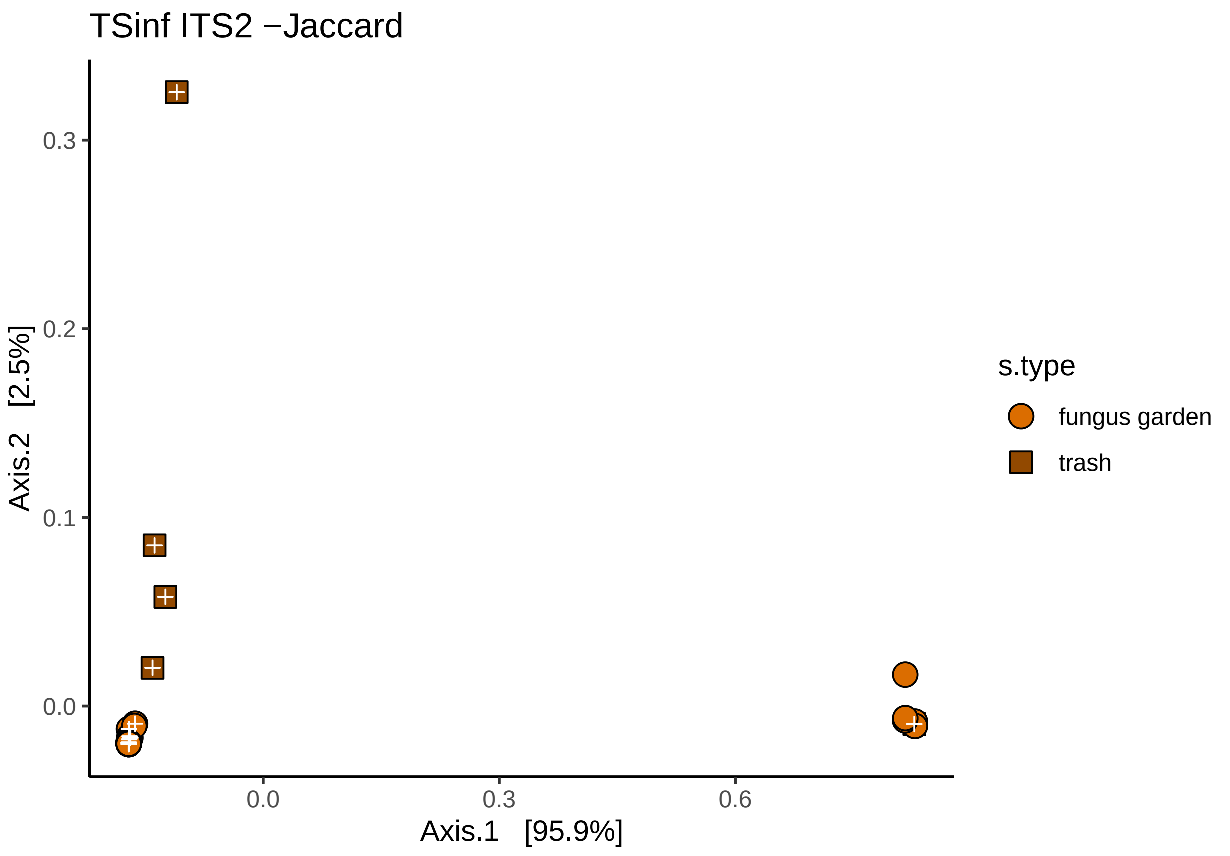
ord.s.type

```
## Warning in plot_ordination(ps, ps.jccord, color = "black", shape = "s.type"):
## Color variable was not found in the available data you provided.No color
## mapped.
```

```
# order legend
p$data$s.type = factor(p$data$s.type, levels = c("fungus garden", "trash"))
p$data$treatment = factor(p$data$treatment, levels = c("Trichoderma", "PBS", "NT"))
p
```



```
# plot +ants on top
ants.y.jccord =
  p$data %>%
  filter(ants.yn == "yes")
p + geom_point(
  data = ants.y.jccord,
  aes(x=Axis.1, y=Axis.2),
  shape = 3,
  color = "white"
  # stroke = 1.1
)
```



```
# Permanova
ps.bcdist = distance(ps, method ="bray")
adonis2(ps.bcdist ~ s.type, data = ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.bcdist ~ s.type, data = ps.samdat.df)
##          Df SumOfSqs     R2      F Pr(>F)
## Model     1   0.0395 0.0081 0.2775  0.588
## Residual 34   4.8433 0.9919
## Total    35   4.8828 1.0000
```

```
adonis2(ps.bcdist ~ expect.inf, data = ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.bcdist ~ expect.inf, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     2   4.8336 0.98991 1618.9  0.001 ***
## Residual 33   0.0493 0.01009
## Total    35   4.8828 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis2(ps.bcdist ~ treatment, data = ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.bcdist ~ treatment, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     2   1.1023 0.22576 4.8112  0.023 *
## Residual 33   3.7805 0.77424
## Total    35   4.8828 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis2(ps.bcdist ~ ants.yn, data = ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.bcdist ~ ants.yn, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1   0.3937 0.08064 2.9821  0.197
## Residual 34   4.4891 0.91936
## Total    35   4.8828 1.00000
```

```
adonis2(ps.bcdist ~ treatment*ants.yn, data = ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.bcdist ~ treatment * ants.yn, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     5   1.9109 0.39134 3.8578  0.027 *
## Residual 30   2.9720 0.60866
## Total    35   4.8828 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Permanova
ps.jccdist = distance(ps, method ="jaccard")
adonis2(ps.jccdist ~ s.type, data = ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.jccdist ~ s.type, data = ps.samdat.df)
##          Df SumOfSqs     R2      F Pr(>F)
## Model     1   0.0889 0.0175 0.6058  0.545
## Residual 34   4.9894 0.9825
## Total    35   5.0783 1.0000
```

```r
adonis2(ps.jccdist ~ expect.inf, data = ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.jccdist ~ expect.inf, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     2   4.9338 0.97155 563.39  0.001 ***
## Residual 33   0.1445 0.02845
## Total    35   5.0783 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
adonis2(ps.jccdist ~ treatment, data = ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.jccdist ~ treatment, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     2   1.1237 0.22127 4.6884  0.026 *
## Residual 33   3.9546 0.77873
## Total    35   5.0783 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
adonis2(ps.jccdist ~ ants.yn, data = ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.jccdist ~ ants.yn, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1   0.4073 0.08021 2.9649  0.177
## Residual 34   4.6710 0.91979
## Total    35   5.0783 1.00000
```

```r
adonis2(ps.jccdist ~ treatment*ants.yn, data = ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.jccdist ~ treatment * ants.yn, data = ps.samdat.df)
##          Df SumOfSqs     R2      F Pr(>F)
## Model     5   1.9511 0.3842 3.7435  0.021 *
## Residual 30   3.1272 0.6158
## Total    35   5.0783 1.0000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
save.image()
```

When you save the notebook, an HTML file containing the code and output will be saved alongside it (click the *Preview* button or press *Cmd+Shift+K* to preview the HTML file).