# 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 - Decon4 and Decon5

Decon4 = infected whole trachy garden Decon5 = healthy whole trachy garden

### Setup

#### 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",
        "ggpubr"
    )

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
##   ggpubr version 0.6.0
```

**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

Based on this tutorial: https://benjjneb.github.io/decontam/vignettes/decontam_intro.html

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

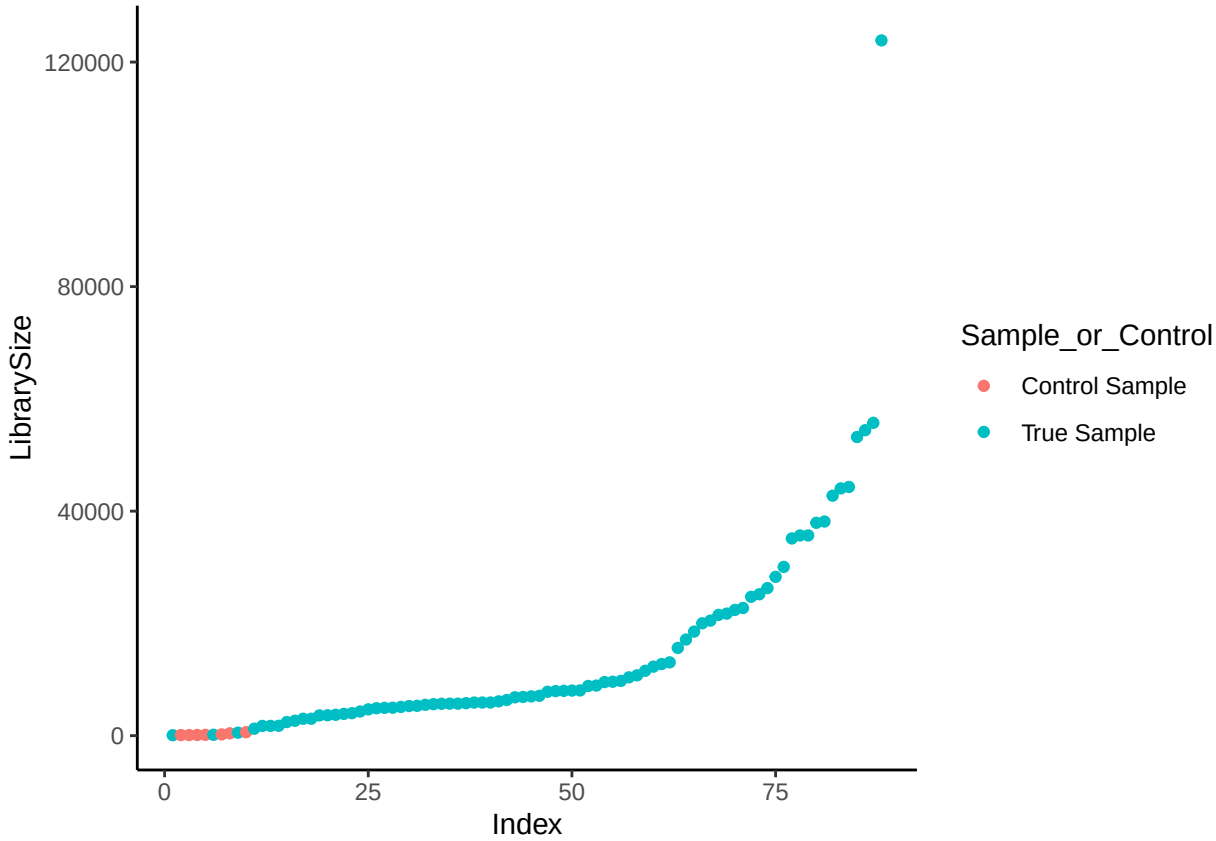
```
##  [1] "d4e01"  "d4e02"  "d4e03"  "d4e04"  "d4e05"  "d4e06"  "d4e07"  "d4e08"
##  [9] "d4e09"  "d4e10"  "d4e11"  "d4e12"  "d4e13"  "d4e14"  "d4e15"  "d4e16"
## [17] "d4e17"  "d4e18"  "d4e19"  "d4e20"  "d4e21"  "d4e22"  "d4e23"  "d4e24"
## [25] "d4e25"  "d4e26"  "d4e27"  "d4e28a" "d4e29"  "d4NC1"  "d4NC2"  "d5e01"
## [33] "d5e02"  "d5e03"  "d5e04"  "d5e05"  "d5e06"  "d5e07"  "d5e08"  "d5e09"
## [41] "d5e10"  "d5e11"  "d5e12"  "d5e13"  "d5e14"  "d5e15"  "d5e16"  "d5e17"
## [49] "d5e18"  "d5e19"  "d5e20"  "d5e21"  "d5e22"  "d5e23"  "d5e24"  "d5e25"
## [57] "d5e26"  "d5e27"  "d5e28"  "d5e29"  "d5e30"  "d5e31"  "d5e32"  "d5e33"
## [65] "d5e34"  "d5e35"  "d5e36"  "d5e37"  "d5e38"  "d5e39"  "d5e40"  "d5e41"
## [73] "d5e42a" "d5e43"  "d5e44"  "d5e45"  "d5e46"  "d5e47"  "d5e48"  "d5e49"
## [81] "d5e50"  "d5e51"  "d5e52"  "d5NC1"  "d5NC2"  "d5NC3"  "d5NC4"  "NFW2"
```

```
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"
```

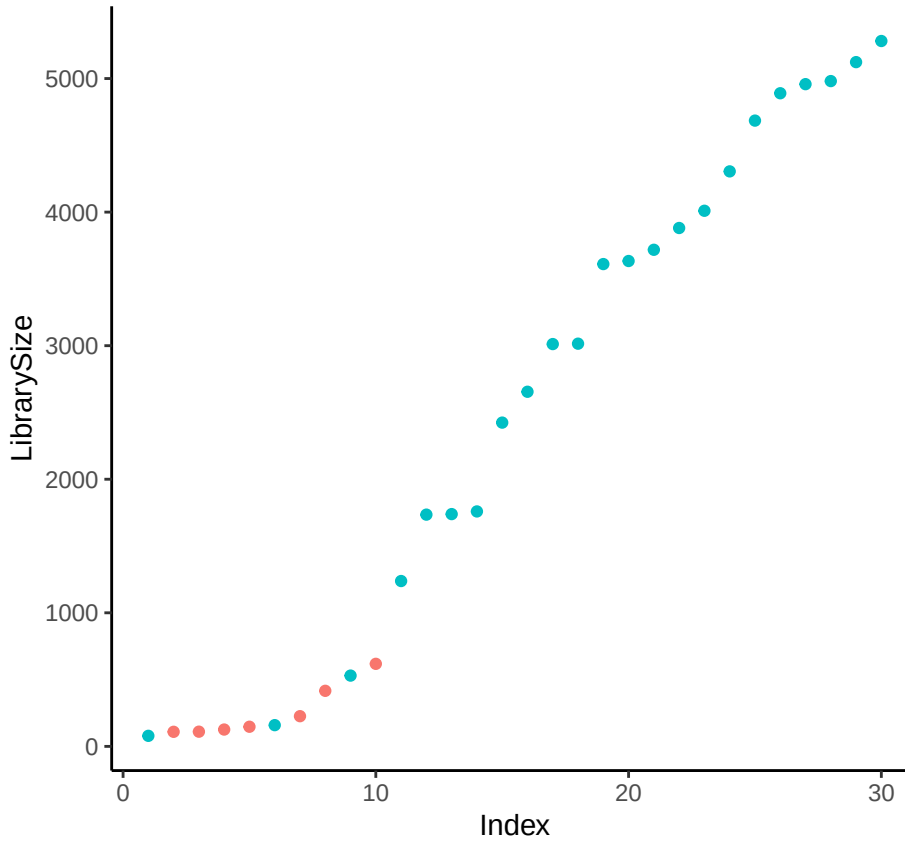
**Plot Lib Size vs. True Sample or Control Sample**

```
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()
```




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

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




**Zoom – Plot Lib Size vs. True or Control**

**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
##   339    21
##
## $prev0.25
## contaminant
## FALSE  TRUE
##   337    23
```

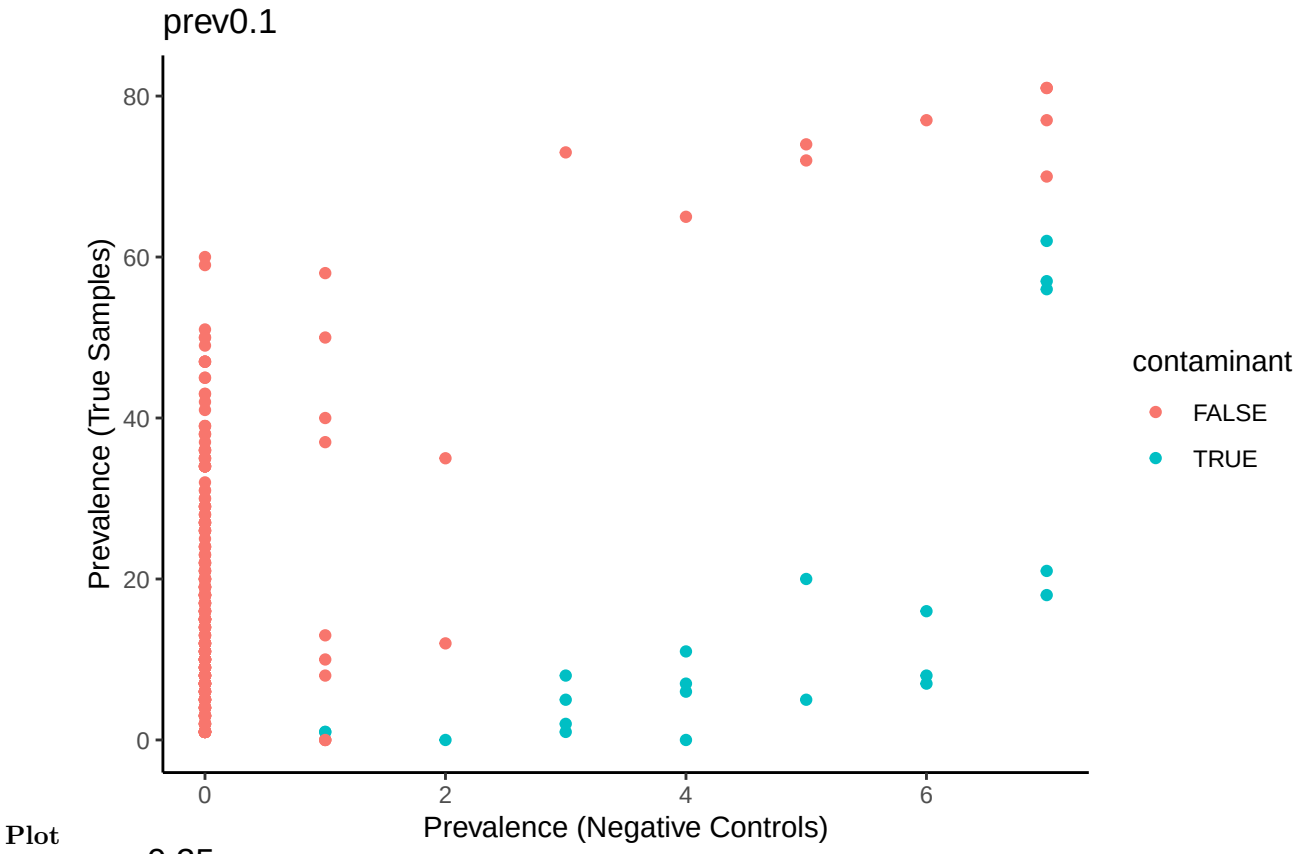
```
##
## $prev0.5
## contaminant
## FALSE  TRUE
##   334    26
```

### Pres-Abs

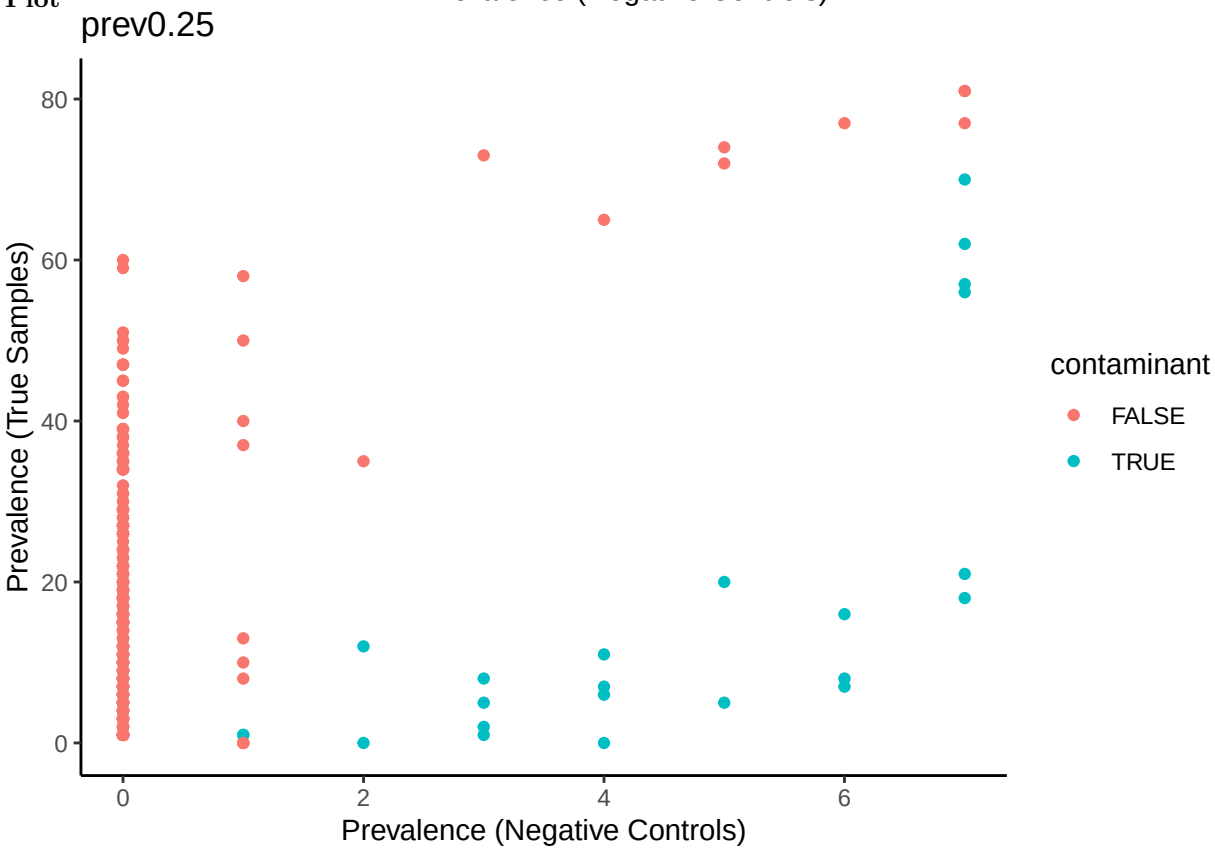
Make phyloseq object of presence-absence in negative controls and true samples

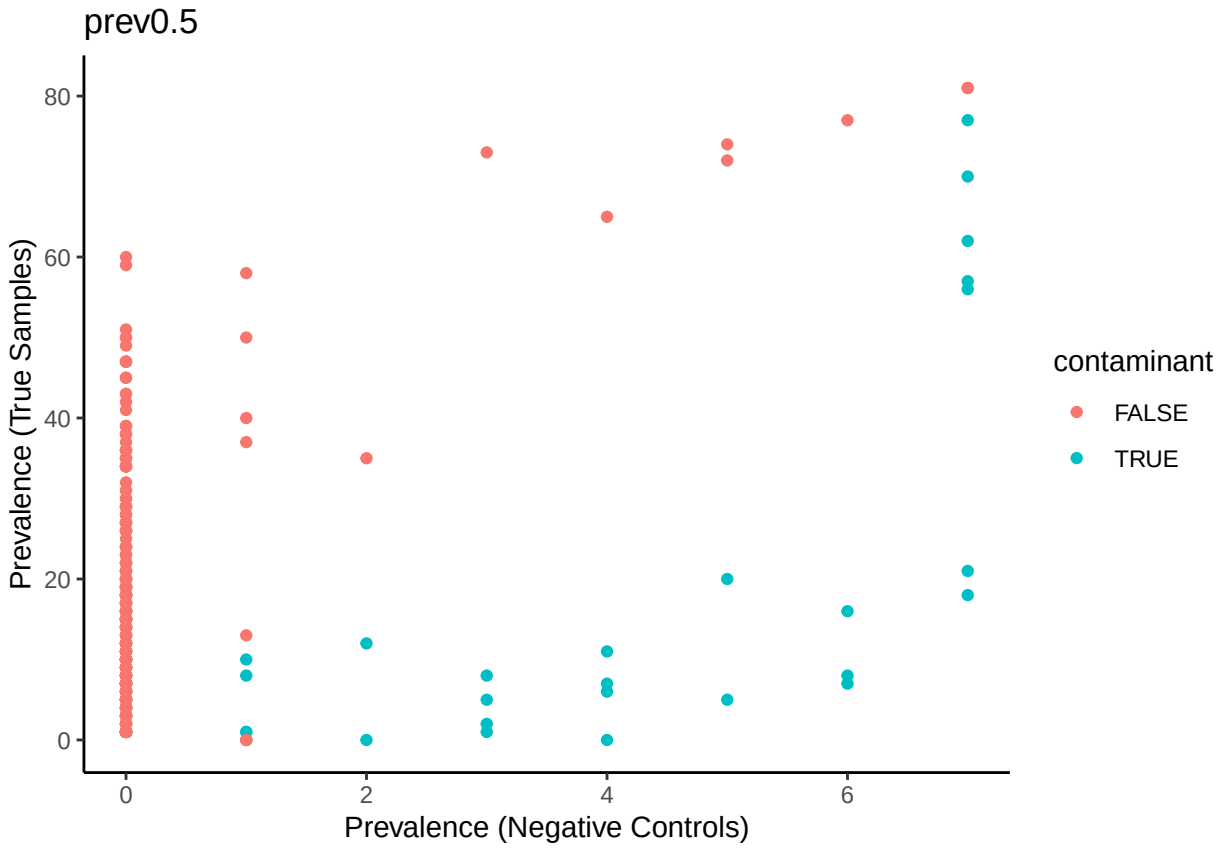
```
ps.p.pa <- transform_sample_counts(ps.p, function(abund) 1*(abund>0))

# 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)
contaminant
FALSE
TRUE
0
20
40
60
80
0
2
4
6

Plot


prev0.25
Prevalence (True Samples)
Prevalence (Negative Controls)
contaminant
FALSE
TRUE
0
20
40
60
80
0
2
4
6




**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)
}
```

**nums.csv**

Create file with stats from decontam like number of reads lost, % reads lost, etc.

```
## ---- 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 %>%
```

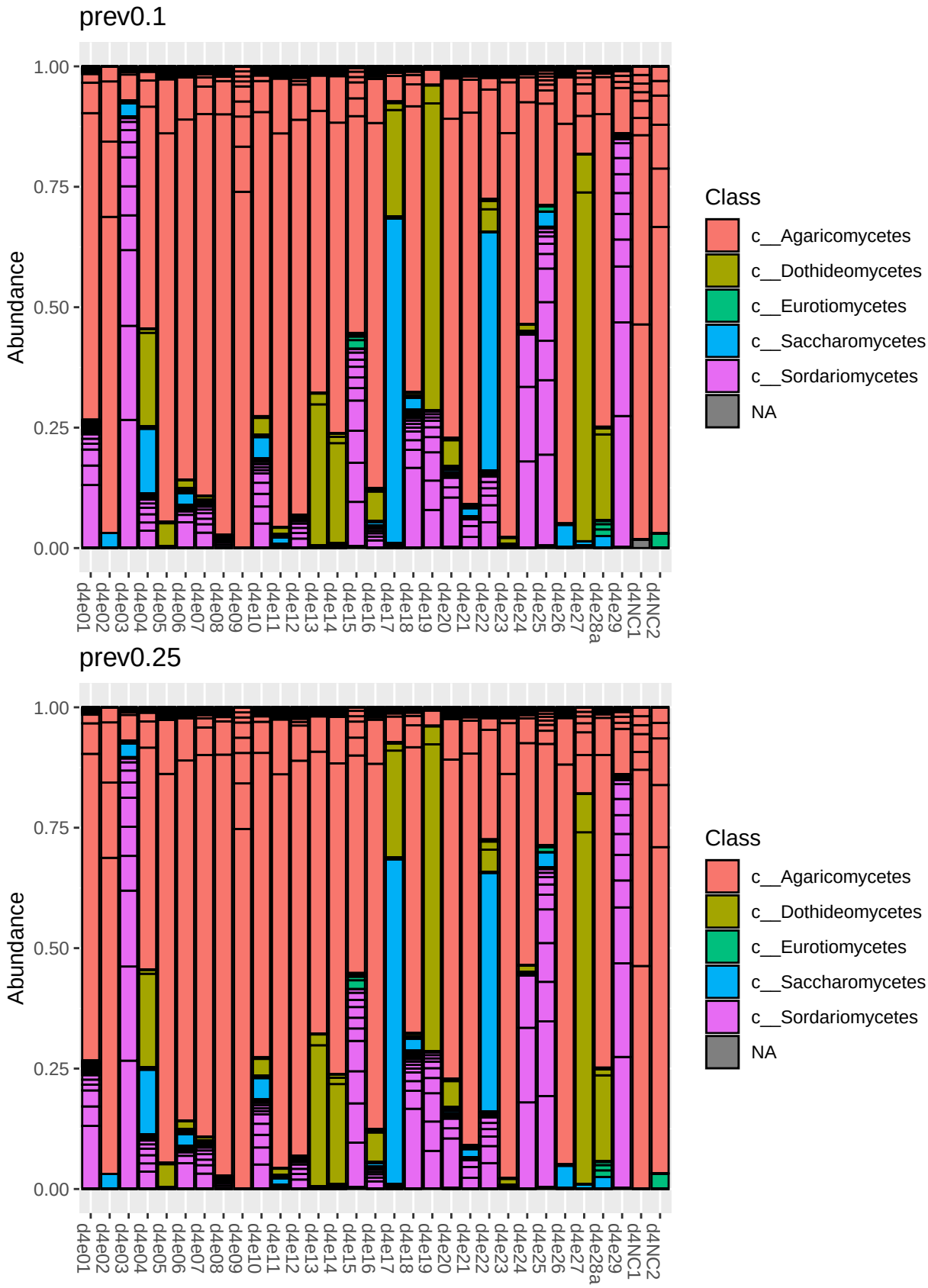
```
  mutate(prev0.1.percL = (ps.p.nreads-prev0.1.nreads)/ps.p.nreads*100) %>%
  mutate(prev0.25.percL = (ps.p.nreads-prev0.25.nreads)/ps.p.nreads*100) %>%
  mutate(prev0.5.percL = (ps.p.nreads-prev0.5.nreads)/ps.p.nreads*100)

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")
```

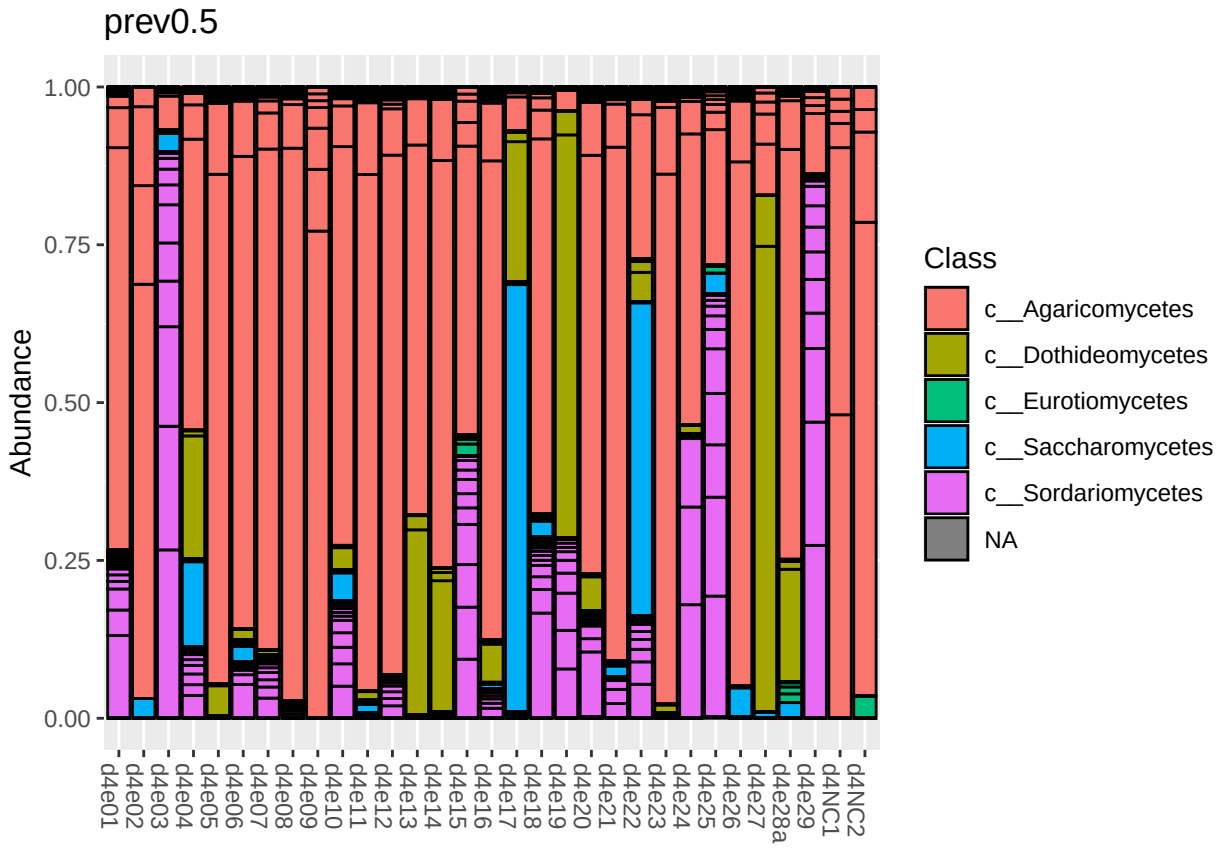
### decontam StackedBar

```
# 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, "^d4") ~ "decon4",
      str_detect(seq.id, "^d5") ~ "decon5",
      # 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 DECON4 by phylum

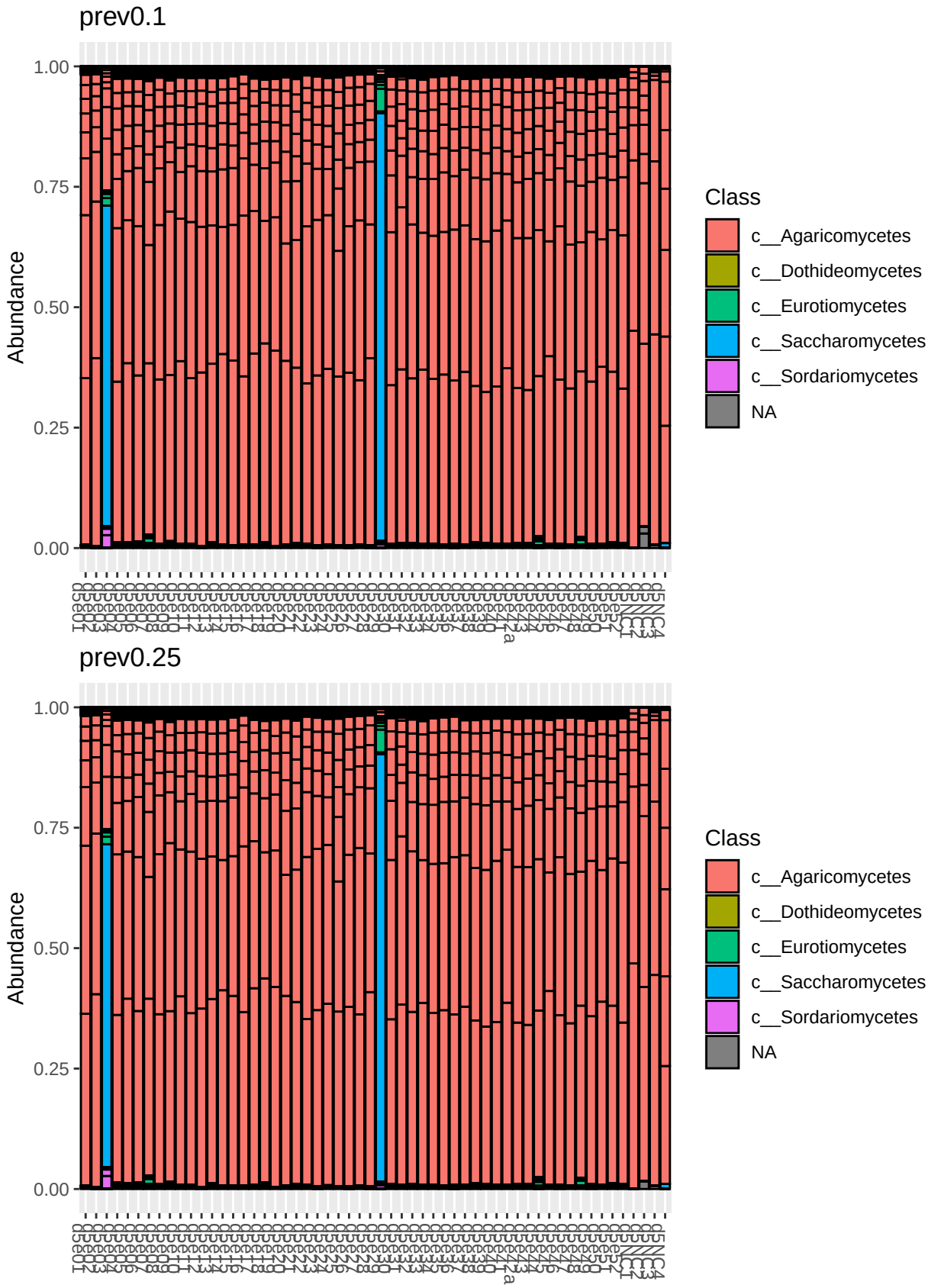
```
for( ln in names(rab.ps.nocontam.ls) ) {
  print( rab.ps.nocontam.ls[[ln]] %>%
    subset_samples(exp.name == "decon4") %>%
    plot_bar(fill="Class") +
    xlab(NULL) +
    ggtitle(ln) )
}
```

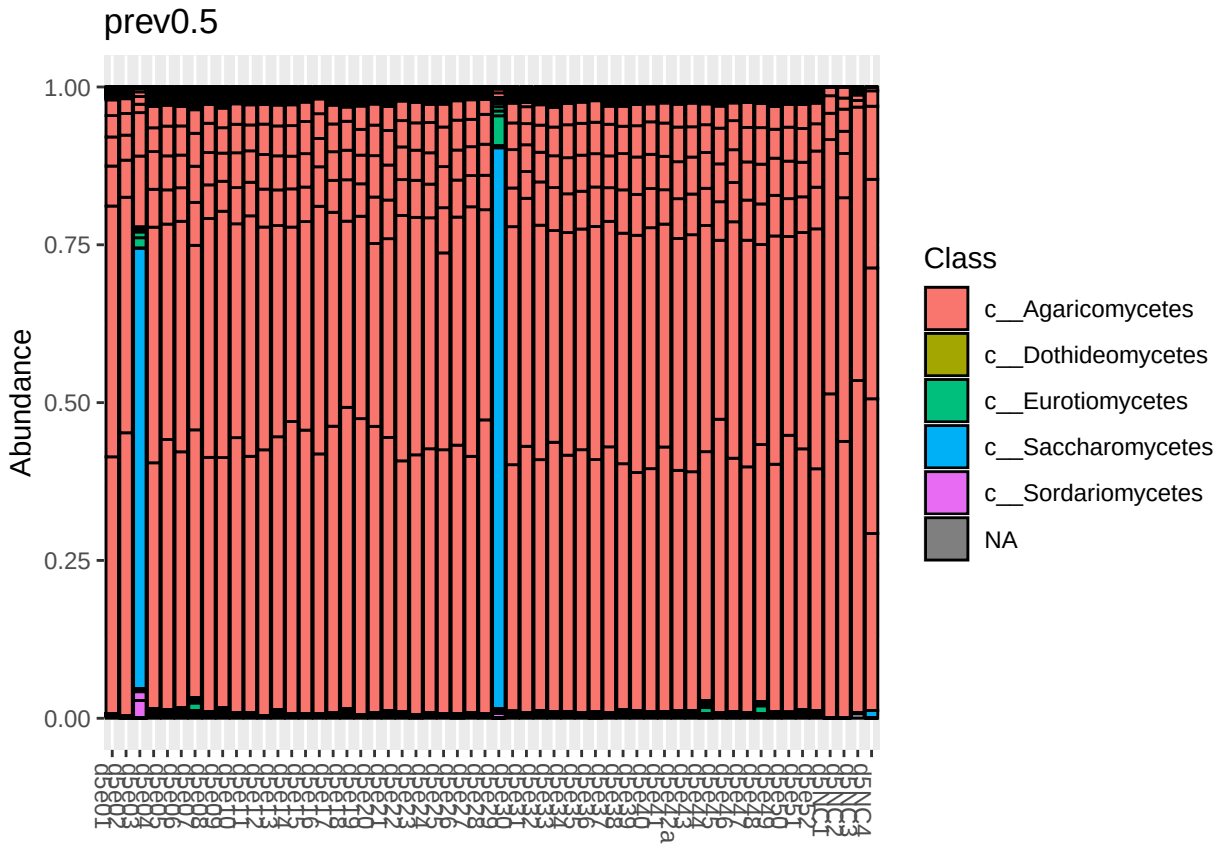

prev0.1
Abundance
1.00
0.75
0.50
0.25
0.00
d4e01
d4e02
d4e03
d4e04
d4e05
d4e06
d4e07
d4e08
d4e09
d4e10
d4e11
d4e12
d4e13
d4e14
d4e15
d4e16
d4e17
d4e18
d4e19
d4e20
d4e21
d4e22
d4e23
d4e24
d4e25
d4e26
d4e27
d4e28a
d4e29
d4NC1
d4NC2
Class
c__Agaricomycetes
c__Dothideomycetes
c__Eurotiomycetes
c__Saccharomycetes
c__Sordariomycetes
NA
prev0.25
Abundance
1.00
0.75
0.50
0.25
0.00
d4e01
d4e02
d4e03
d4e04
d4e05
d4e06
d4e07
d4e08
d4e09
d4e10
d4e11
d4e12
d4e13
d4e14
d4e15
d4e16
d4e17
d4e18
d4e19
d4e20
d4e21
d4e22
d4e23
d4e24
d4e25
d4e26
d4e27
d4e28a
d4e29
d4NC1
d4NC2
Class
c__Agaricomycetes
c__Dothideomycetes
c__Eurotiomycetes
c__Saccharomycetes
c__Sordariomycetes
NA




Plot noncontam DECON5 by phylum

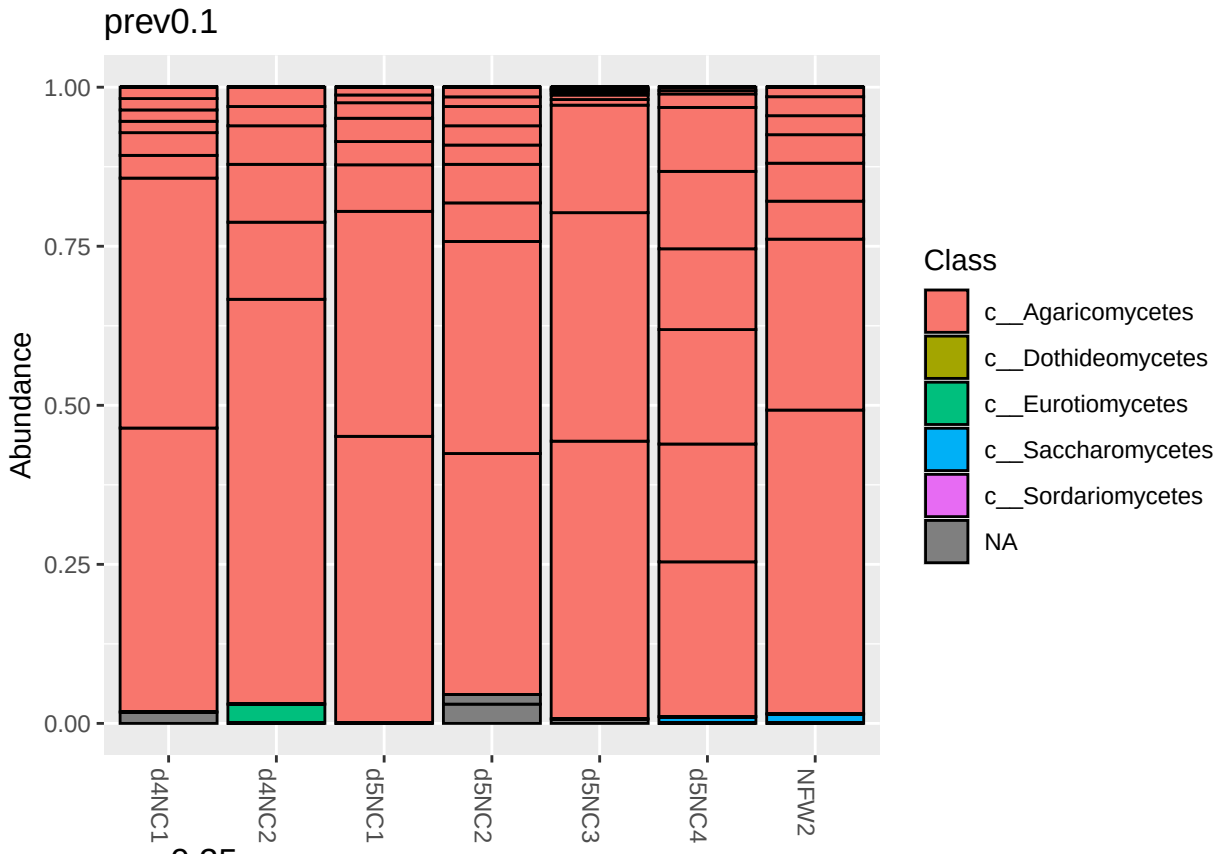
```
for( ln in names(rab.ps.nocontam.ls) ) {
  print( rab.ps.nocontam.ls[[ln]] %>%
    subset_samples(exp.name == "decon5") %>%
    plot_bar(fill="Class") +
    xlab(NULL) +
    ggtitle(ln) )
}
```

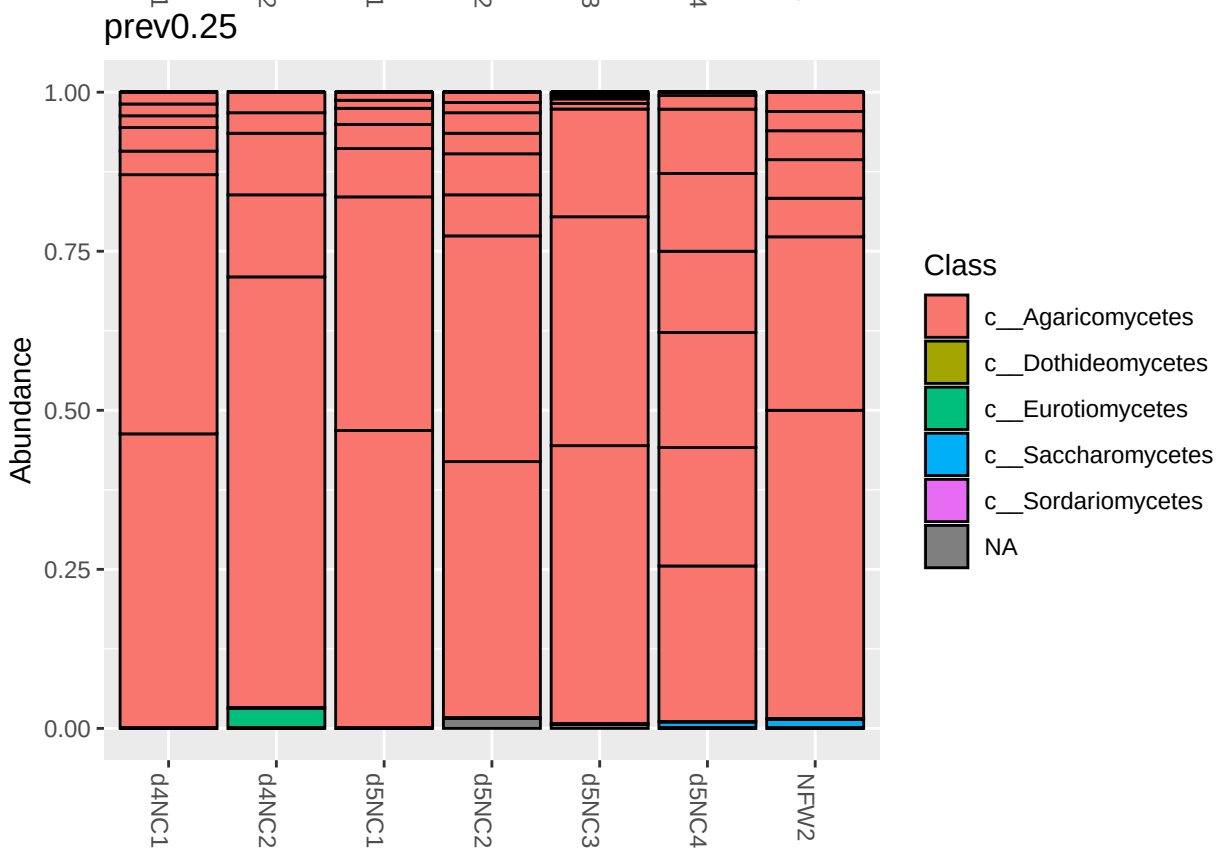

prev0.1
Abundance
1.00
0.75
0.50
0.25
0.00
Class
c__Agaricomycetes
c__Dothideomycetes
c__Eurotiomycetes
c__Saccharomycetes
c__Sordariomycetes
NA
prev0.25
Abundance
1.00
0.75
0.50
0.25
0.00
Class
c__Agaricomycetes
c__Dothideomycetes
c__Eurotiomycetes
c__Saccharomycetes
c__Sordariomycetes
NA

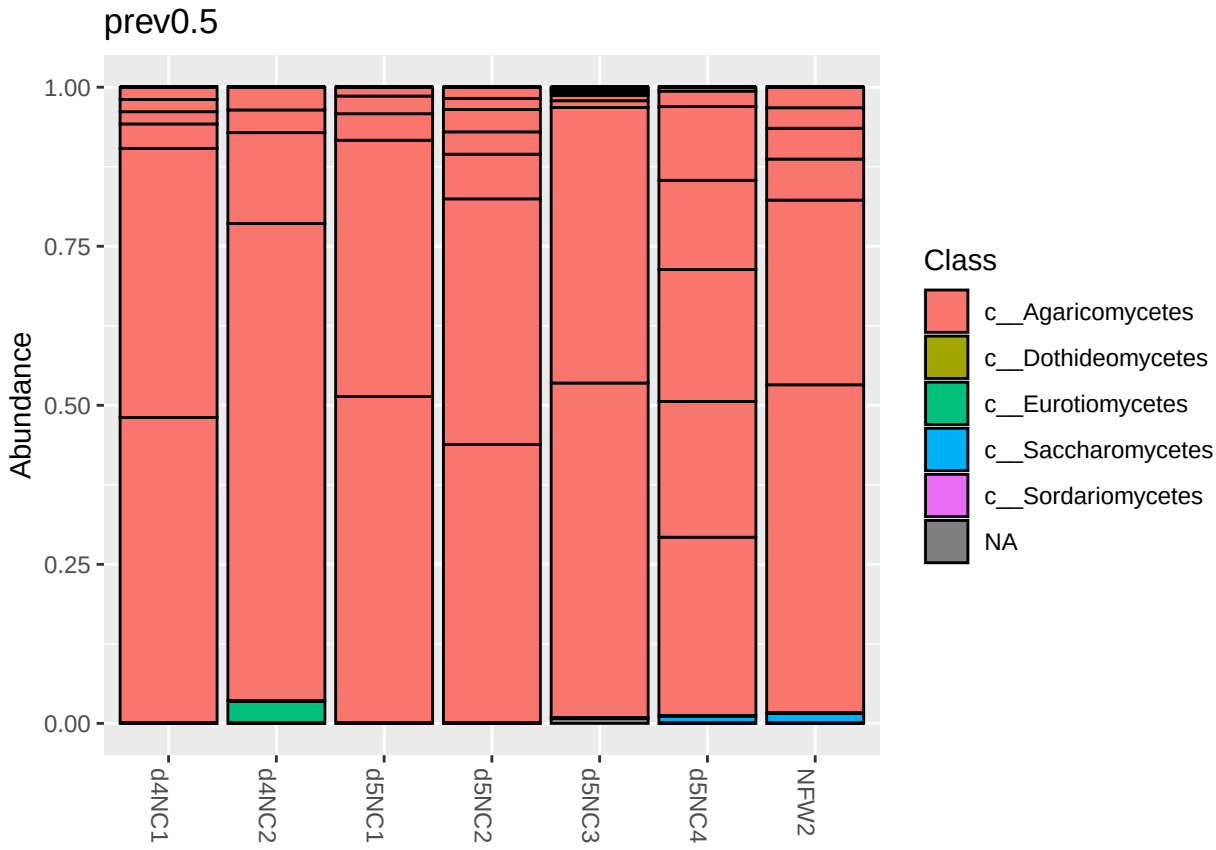



Check how controls look after decontam

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


prev0.1
Abundance
1.00
0.75
0.50
0.25
0.00
d4NC1
d4NC2
d5NC1
d5NC2
d5NC3
d5NC4
NFW2
Class
c__Agaricomycetes
c__Dothideomycetes
c__Eurotiomycetes
c__Saccharomycetes
c__Sordariomycetes
NA


prev0.25
Abundance
1.00
0.75
0.50
0.25
0.00
d4NC1
d4NC2
d5NC1
d5NC2
d5NC3
d5NC4
NFW2
Class
c__Agaricomycetes
c__Dothideomycetes
c__Eurotiomycetes
c__Saccharomycetes
c__Sordariomycetes
NA




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

```
## d4NC2 d4NC1 d5NC2  NFW2 d5NC1 d5NC4 d5NC3
##    28    52    57    62    72   164   471
```

## Basic Filtering

This custom function renames taxa that are "NA" by whatever their lowest prior classification was. For example: Genus: Leucoagaricus; Species: NA ==> Species: "unclass. Leucoagaricus". YOU MUST DO THIS BEFORE USING subset_taxa()!!! Phyloseq's subset_taxa() will also prune NA's. Example: subset_taxa(Phylum != "unclass. k__Fungi") –> this will remove anything that's unclassified kingdom Fungi, but it will also remove anything that is NA for phylum. This is more important when filtering at lower taxonomic levels, like chloroplast and mitochondria filtering in 16S data.

```
ps = na_to_unclassified_taxa(ps.nocontam.ls$prev0.5)
```

I'm choosing to use decontam prevalence threshold of 0.5. Removing negative controls samples. Filtering highly unclassified taxa

```
# remove negative controls
ps.nonc = prune_samples(!(sample_names(ps) %>% grepl(pattern = "NC|NFW|NFH2")),
                        ps)

# starting with 334 taxa and 87 samples
ps.filt = subset_taxa(ps.nonc, Kingdom == "k__Fungi")               # 334 taxa
ps.filt = subset_taxa(ps.filt, Phylum != "unclass. k__Fungi")   # 328 taxa

# split by experiment
psList.filt = list(
```

```
  decon4 = subset_samples(ps.filt, exp.name == "decon4"),
  decon5 = subset_samples(ps.filt, exp.name == "decon5")
)
```

## Make cultivar fungus

Custom function that defines "cultivar fungus" because the cultivar is obviously a trouble maker and doesn't want to be defined. So this will change anything that is genus Leucocoprinus or genus Leucoagaricus or genus unclassified family Agaricaceae and set the genus name to "Cultivar fungus". Also gloms by genus. It also outputs a fasta file of these potential cultivar fungus ASVs so that you can double check that they BLAST to an attine ant associated fungus.

```
# define cultivar fungus genus
psList.filt.cf = psList.filt
for( ln in names(psList.filt.cf) ) {
  psList.filt.cf[[ln]] = make_cultivar_fungus(psList.filt[[ln]], paste0(ln,"_") )
}
```

```
## >>> Potential cultivar fungus ASVs detected. Creating fasta ... ... ...
## >>> Writing file ./dada2/fasta/decon4_cultivar_fungus_ASVs.fasta  ... ... ...
## >>> Potential cultivar fungus ASVs detected. Creating fasta ... ... ...
## >>> Writing file ./dada2/fasta/decon5_cultivar_fungus_ASVs.fasta  ... ... ...
```

```
psList.filt.cf # d4: 23 taxa, d5: 23 taxa
```

```
## $decon4
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 23 taxa and 29 samples ]
## sample_data() Sample Data:       [ 29 samples by 3 sample variables ]
## tax_table()   Taxonomy Table:    [ 23 taxa by 7 taxonomic ranks ]
## refseq()      DNAStringSet:      [ 23 reference sequences ]
##
## $decon5
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 23 taxa and 52 samples ]
## sample_data() Sample Data:       [ 52 samples by 3 sample variables ]
## tax_table()   Taxonomy Table:    [ 23 taxa by 7 taxonomic ranks ]
## refseq()      DNAStringSet:      [ 23 reference sequences ]
```

## Save final ps

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

## 1% filter

per_sample_taxafilt: Custom function to do "present at >=X abundance in at least one sample" filtering. The default is 1% (0.01). But you can send different thresholds as decimals like 0.05 for 5%, etc. You can optionally glom by whatever taxa rank at the same time.

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

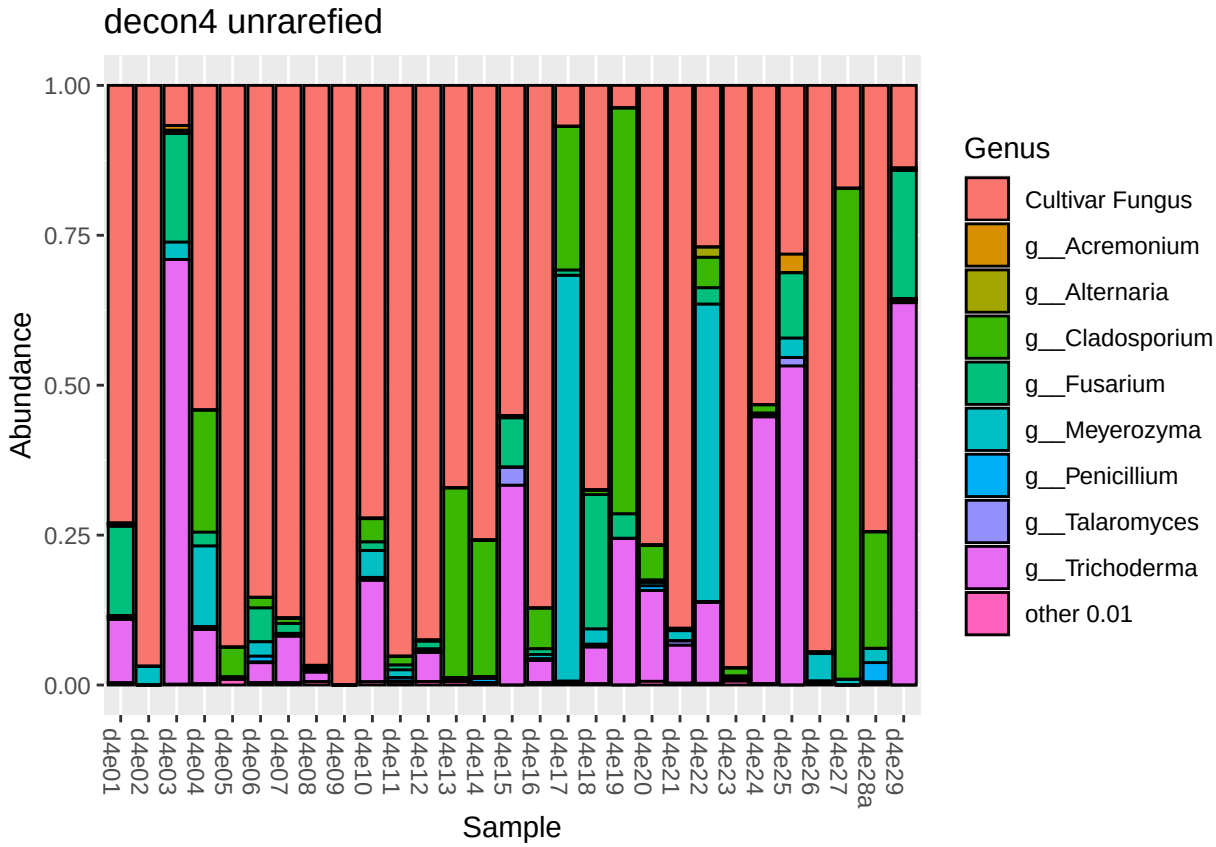
```
##
## >>> Glomming by Genus ... ... ...
##
## >>> Glomming by Genus ... ... ...
```

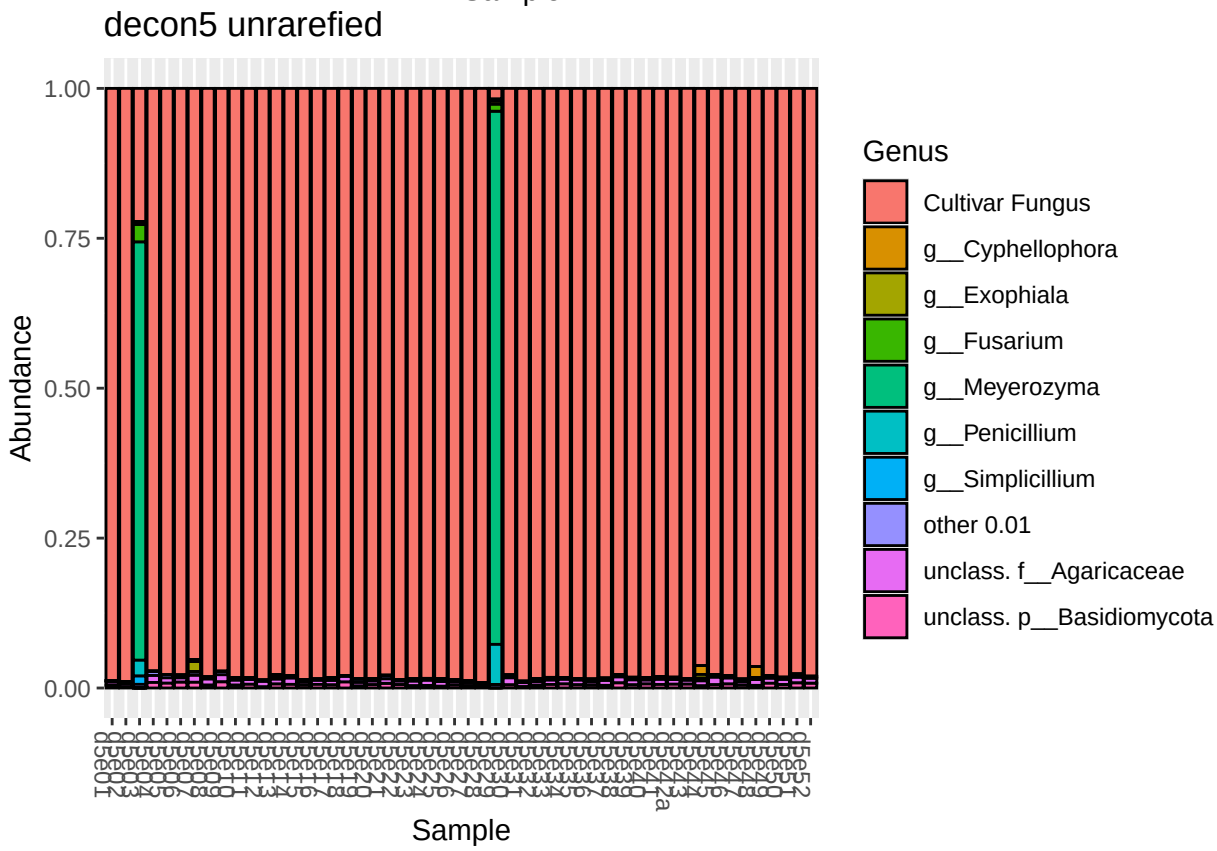
```
psList.filt.cf.0.01 # d4: 15 taxa, d5: 10 taxa
```

```
## $decon4
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 10 taxa and 29 samples ]
## sample_data() Sample Data:       [ 29 samples by 3 sample variables ]
## tax_table()   Taxonomy Table:    [ 10 taxa by 7 taxonomic ranks ]
## refseq()      DNAStringSet:      [ 10 reference sequences ]
##
## $decon5
## phyloseq-class experiment-level object
## otu_table()   OTU Table:         [ 10 taxa and 52 samples ]
## sample_data() Sample Data:       [ 52 samples by 3 sample variables ]
## tax_table()   Taxonomy Table:    [ 10 taxa by 7 taxonomic ranks ]
## refseq()      DNAStringSet:      [ 10 reference sequences ]
```

### stacked bar plot

```
fin.psList = psList.filt.cf.0.01

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


decon4 unrarefied
Abundance
1.00
0.75
0.50
0.25
0.00
d4e01
d4e02
d4e03
d4e04
d4e05
d4e06
d4e07
d4e08
d4e09
d4e10
d4e11
d4e12
d4e13
d4e14
d4e15
d4e16
d4e17
d4e18
d4e19
d4e20
d4e21
d4e22
d4e23
d4e24
d4e25
d4e26
d4e27
d4e28a
d4e29
Sample
Genus
Cultivar Fungus
g__Acremonium
g__Alternaria
g__Cladosporium
g__Fusarium
g__Meyerozyma
g__Penicillium
g__Talaromyces
g__Trichoderma
other 0.01


decon5 unrarefied
Abundance
1.00
0.75
0.50
0.25
0.00
Sample
Genus
Cultivar Fungus
g__Cyphellophora
g__Exophiala
g__Fusarium
g__Meyerozyma
g__Penicillium
g__Simplicillium
other 0.01
unclass. f__Agaricaceae
unclass. p__Basidiomycota

### Sample Data

Add relevant metadata to the phyloseq objects.

```
## ---- idk why this sample has an 'a' at the end...
w = which(sample_names(fin.psList$decon4) == "d4e28a")
sample_names(fin.psList$decon4)[w] = "d4e28"

## ---- idk why this sample has an 'a' at the end...
w = which(sample_names(fin.psList$decon5) == "d5e42a")
sample_names(fin.psList$decon5)[w] = "d5e42"
```

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

decon4.nonc.samdat =
  read.csv("~/Documents/ch3_analyses/decons_master_sampledata.csv")  %>%
  filter(grepl(pattern="^d4", seq.id)) %>%
  filter(!grepl(pattern="NC|NFW|NFH", seq.id))
decon5.nonc.samdat =
  read.csv("~/Documents/ch3_analyses/decons_master_sampledata.csv")  %>%
  filter(grepl(pattern="^d5", seq.id)) %>%
  filter(!grepl(pattern="NC|NFW|NFH", seq.id))

sdList = list(
        decon4 = decon4.nonc.samdat,
        decon5 = decon5.nonc.samdat
    )

# ---- check if sample names match between ps and samdat and set samdat if they do
for( i in seq_along(fin.psList) ){
    if(  all( sdList[[i]]$s.id %in% sample_names(fin.psList[[i]]) ) ) {
        cat(">>> sample names for",
            names(fin.psList)[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(fin.psList[[i]]) = sdList[[i]]

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

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

    } else {
        cat(">>> sample names for",
            names(fin.psList)[i],
            "phyloseq object do not match s.id's for",
            names(sdList)[i],
```

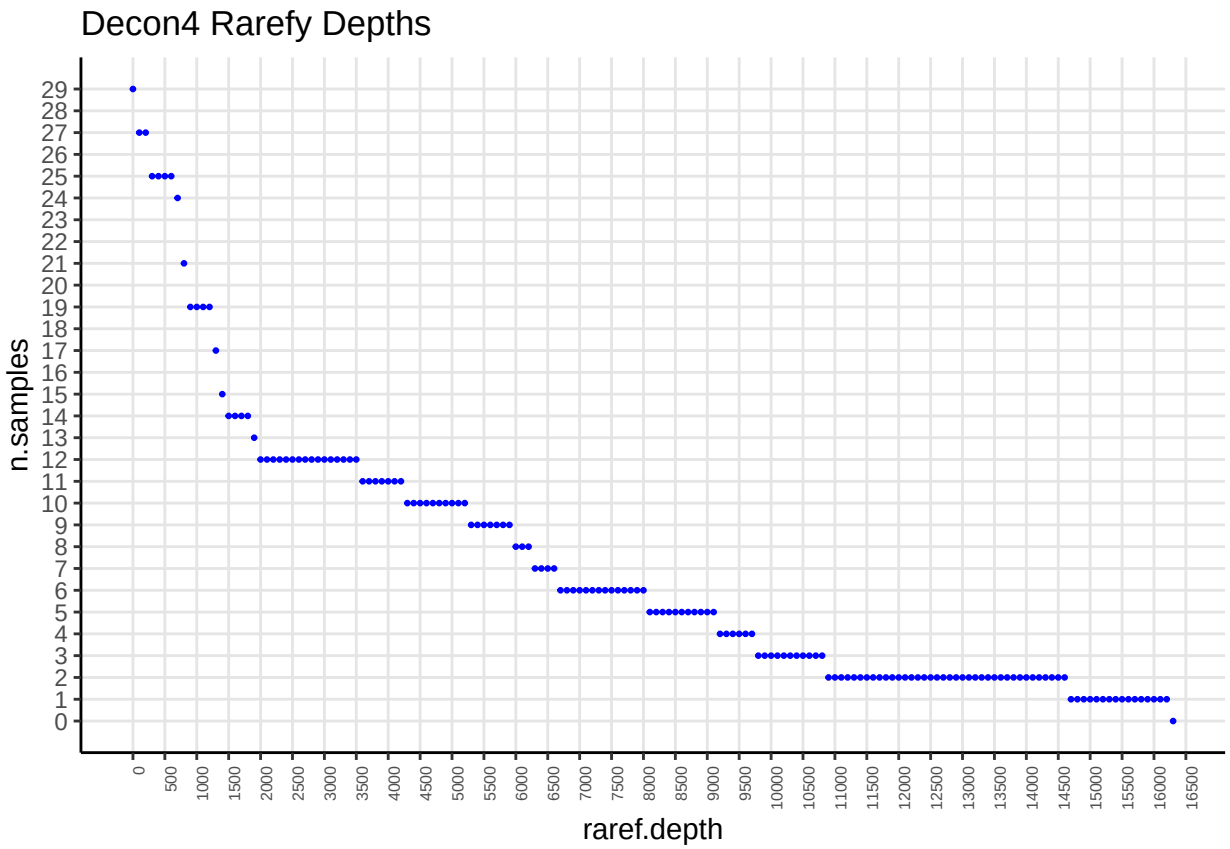
```
            "samdat! Please reorder.\n"
        )
    }
}
```

```
## >>> sample names for decon4 phyloseq object s.id are the same as s.id in decon4 samdat.
## >>> sample_data slot filled for phyloseq object named decon4 .
## >>> Saving file decon4.ps.filt.RDS ... ... ...
## >>> sample names for decon5 phyloseq object s.id are the same as s.id in decon5 samdat.
## >>> sample_data slot filled for phyloseq object named decon5 .
## >>> Saving file decon5.ps.filt.RDS ... ... ...
```

## Rarefy

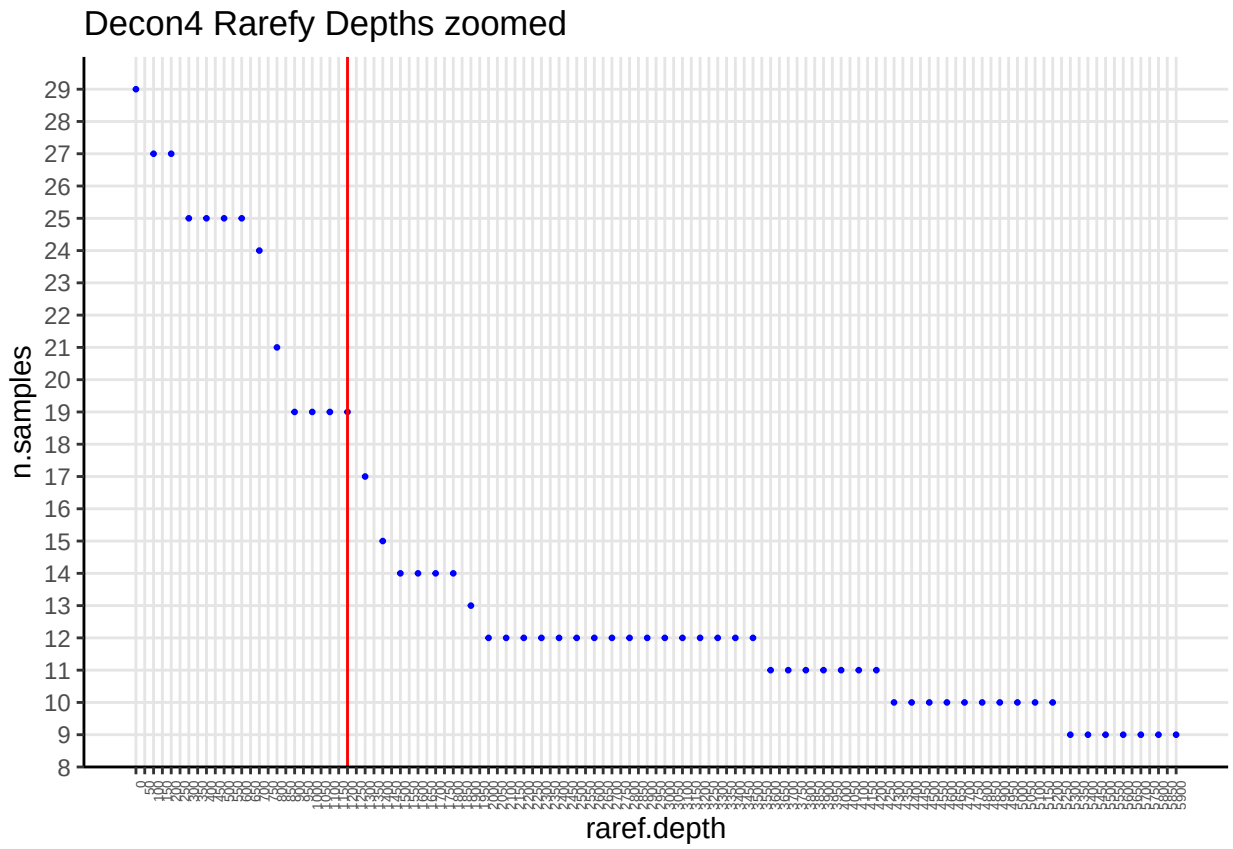
### Decon4 plot

```
## plot raref threshold increments of 50 reads
ps = fin.psList$decon4

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("Decon4 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")
    )
```




```
## 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("Decon4 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")
    ) +
```

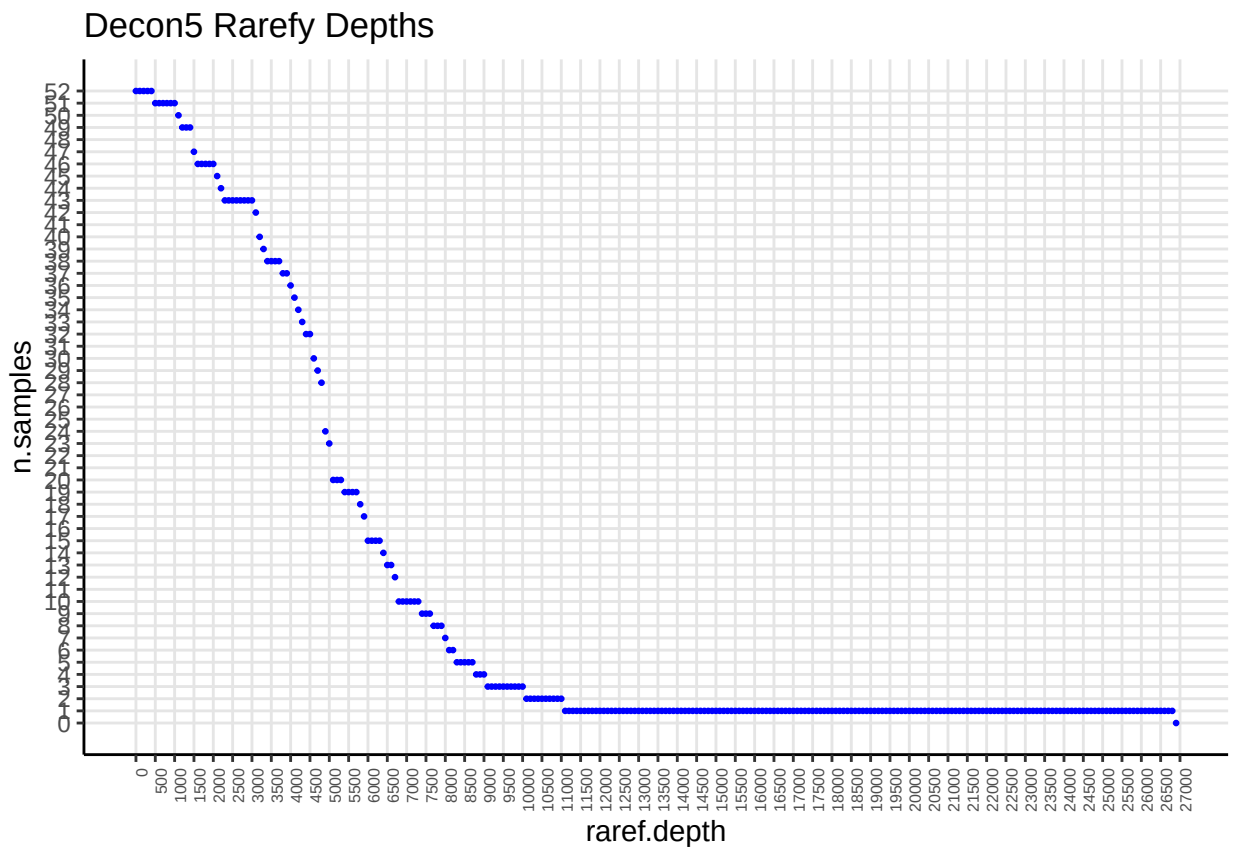
```
    geom_vline(xintercept = 1200, color="red")
```




**Decon5 plot**

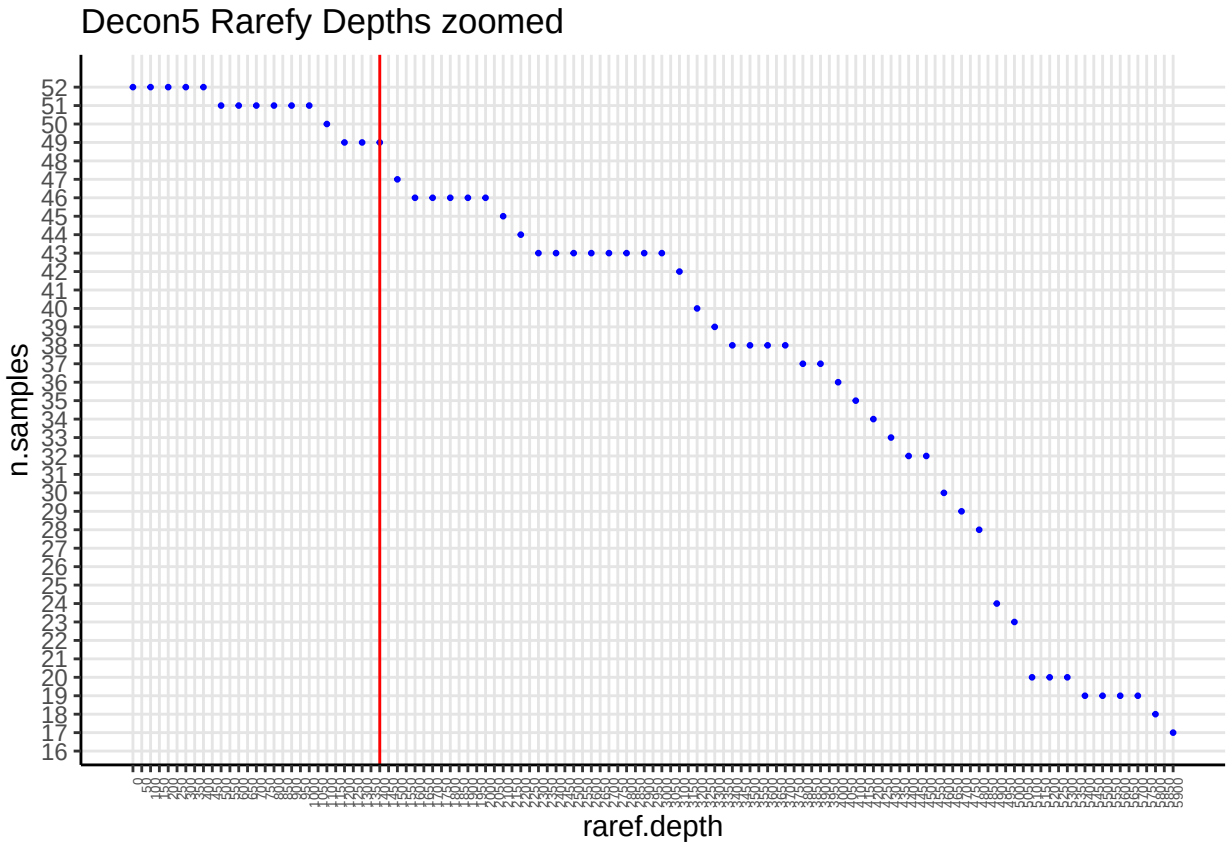
```
## plot raref threshold increments of 50 reads
ps = fin.psList$decon5

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("Decon5 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")
    )
```




```
## 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("Decon5 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 = 1400, color="red")
```




**do the raref**

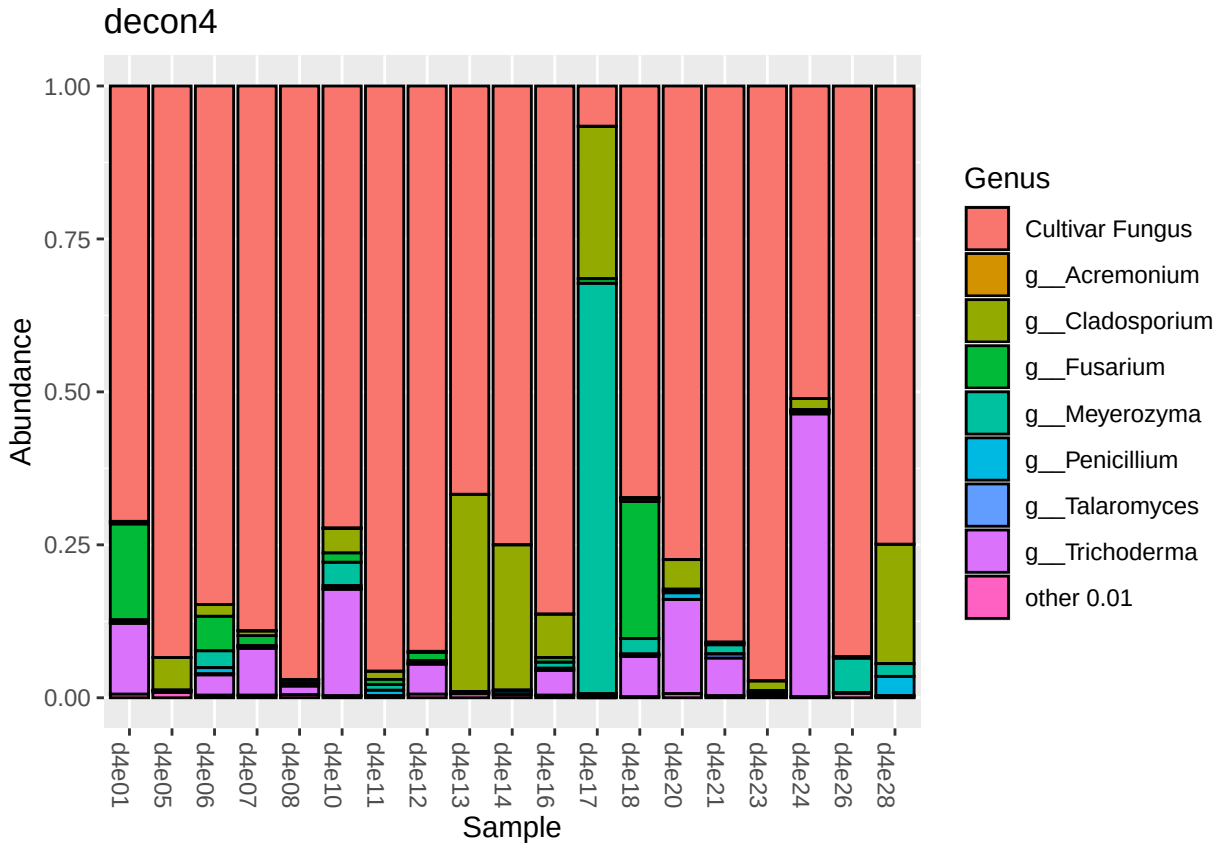
Rarefying decon4 to 1200 reads, decon5 to 1400 ... ... ...

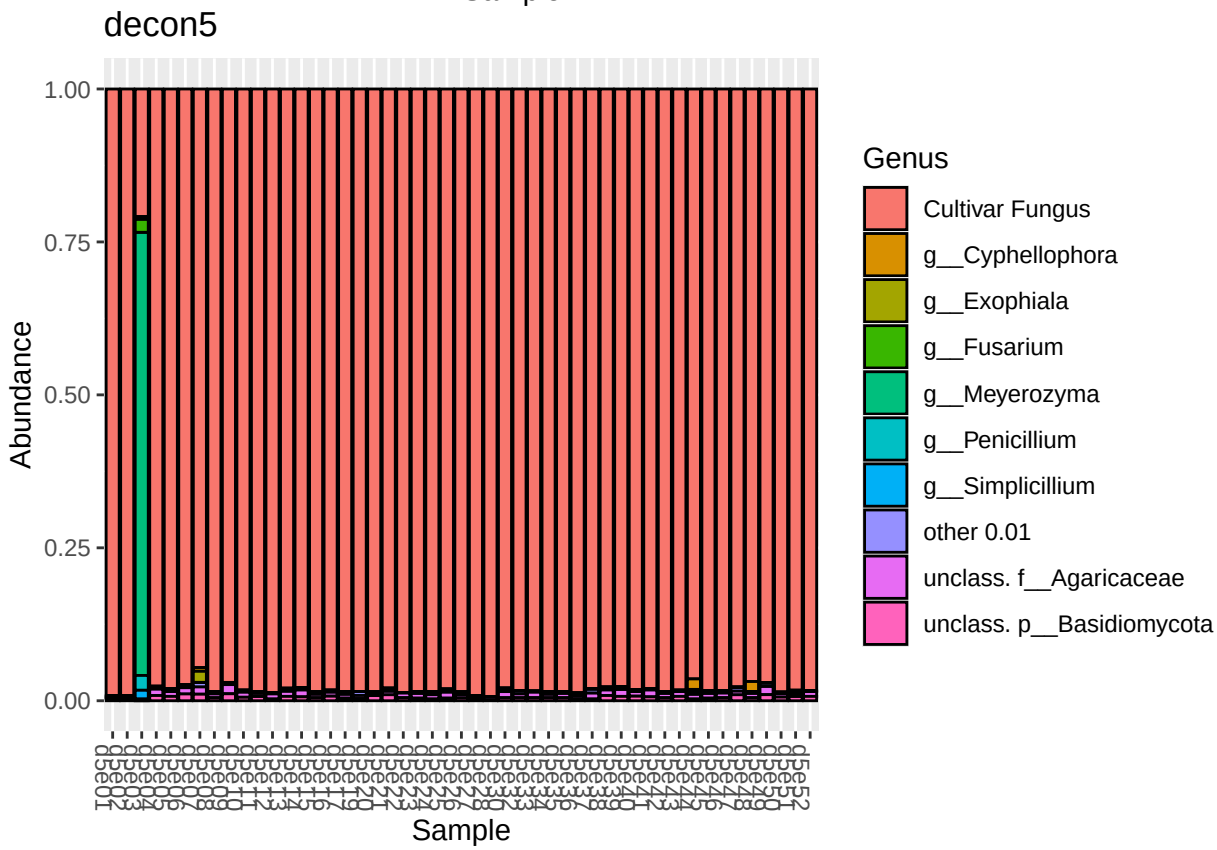
```
# d4 -- raref to 761, 883, 883, 1214, 1290, 1312 ... 1200
# d5 -- raref to 445, 1064, 1182, 1479, 1485, 1517, 2099 ... 1400
fin.psList.raref =
  list(
    decon4 = rarefy_even_depth(fin.psList$decon4, sample.size=1200, rngseed=1103, replace=F),
    decon5 = rarefy_even_depth(fin.psList$decon5, sample.size=1400, 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
```

```
## ...
## 10 samples removedbecause they contained fewer reads than `sample.size`.
## Up to first five removed samples are:
## d4e02d4e03d4e04d4e09d4e15
## ...
## 1OTUs were removed because they are no longer
## present in any sample after random subsampling
## ...
## `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
## ...
## 3 samples removedbecause they contained fewer reads than `sample.size`.
## Up to first five removed samples are:
## d5e18d5e29d5e31
## ...
```

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


decon4
1.00
0.75
0.50
0.25
0.00
Abundance
d4e01
d4e05
d4e06
d4e07
d4e08
d4e10
d4e11
d4e12
d4e13
d4e14
d4e16
d4e17
d4e18
d4e20
d4e21
d4e23
d4e24
d4e26
d4e28
Sample
Genus
Cultivar Fungus
g__Acremonium
g__Cladosporium
g__Fusarium
g__Meyerozyma
g__Penicillium
g__Talaromyces
g__Trichoderma
other 0.01


decon5
1.00
0.75
0.50
0.25
0.00
Abundance
Sample
Genus
Cultivar Fungus
g__Cyphellophora
g__Exophiala
g__Fusarium
g__Meyerozyma
g__Penicillium
g__Simplicillium
other 0.01
unclass. f__Agaricaceae
unclass. p__Basidiomycota

## Heatmaps

### Data Wrangling

Found this heatmap code that can do many side color bars on a heatmap here: https://www.biostars.org/p/18211/

```
## ---- 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 )
```

### Decon4

```
ps.m = psList.melt$decon4

ps.m %>% filter(trashed.yn == "maybe") %>% select(s.id, s.name) %>% unique() # d4e10:S24
```

### Data wrangling...

```
## [1] s.id   s.name
## <0 rows> (or 0-length row.names)
```

```
## change d4e10 (S24) from maybe trashed to yes trashed
ps.m <-
  ps.m %>%
  mutate(
    trashed.yn = replace(trashed.yn, trashed.yn == "maybe","yes")
  )
```

**Color wrangling** This is some nonsense because my "distance from inoculation" is a continuous variable, not discrete. So i wanted to quantitatively map by values to a color gradient because the values are not all the same distance apart. i found this function in a stackoverflow comment somewhere...here it is: https://stackoverflow.com/questions/15006211/how-do-i-generate-a-mapping-from-numbers-to-colors-in-r

```
# map numbers to a color scale in a continuous way instead of discrete
map2color<-function(x,pal,limits=NULL){
    if(is.null(limits)) limits=range(x)
    pal[findInterval(x,seq(limits[1],limits[2],length.out=length(pal)+1), all.inside=TRUE)]
}

# these are my values that need colors
x = sample_data(fin.psList$decon4)$inoc.dist.cm %>% sort() %>% unique()
# i think the number you choose here controls how fine or 'grainy' your colors are
# you need to have at the very least the same number as your number of values to map, but ideally more
mypal <- colorRampPalette( c( "darkgreen","white") )( length(x) + 20 )

# x[-1] bc setting -1 to black
map2color(x[-1], mypal)
```

```
##  [1] "#006400" "#257A25" "#2F812F" "#4A914A" "#4F944F" "#559755" "#64A164"
##  [8] "#74AB74" "#7AAE7A" "#7AAE7A" "#84B484" "#94BE94" "#9AC19A" "#9FC49F"
## [15] "#A4C8A4" "#AACBAA" "#AFCEAF" "#B4D1B4" "#B9D5B9" "#C9DEC9" "#C9DEC9"
## [22] "#C9DEC9" "#CFE1CF" "#CFE1CF" "#DFEBDF" "#E4EEE4" "#EFF5EF" "#FFFFFF"
```

```
# [1] "#006400" "#247A24" "#348334" "#489048" "#4E934E" "#539653" "#68A368" "#77AC77" "#7CAF7C" "#7CAF7
# [11] "#87B687" "#96BF96" "#9CC29C" "#A1C6A1" "#A1C6A1" "#ABCCAB" "#B0CFB0" "#B6D2B6" "#BBD5BB" "#C5DC
# [21] "#CADFCA" "#CADFCA" "#D0E2D0" "#D0E2D0" "#DFECDF" "#E4EFE4" "#EFF5EF" "#FFFFFF"

# save values to color map
full.inoc.dist.cols =
  data.frame(
    inoc.dist.cm = sample_data(fin.psList$decon4)$inoc.dist.cm %>% sort() %>% unique(),
    col = c("black",map2color(x[-1],mypal))
  )
```

Choose your colors or set them randomly from a palette. I like paletteMartin from package colorBlindness: https://cran.r-project.org/web/packages/colorBlindness/vignettes/colorBlindness.html#How_to_use_this_package

```
## ---- decon4 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 =
         data.frame(
           s.type = c("fungus garden","ants"),
           col = c("#db6d00","#920000")
         ),
       trashed.yn =
         data.frame(
           trashed.yn = c("yes","no"),
           col = c("#924900","#ffff6d")),
       inoc.dist.cm = full.inoc.dist.cols
       #  extr.batch =
       #   data.frame(
       #     extr.batch = c(1,2),
       #     col = c("#ff6db6","#006ddb")), # random cols for extr.batch
       # pcr.batch =
       #   data.frame(
       #     pcr.batch = c("1","2"),
       #     col = c("#490092","#b6dbff")) # random cols for pcr.batch
    )

metadat.cols
```

```
## $s.type
##          s.type     col
## 1 fungus garden #db6d00
## 2          ants #920000
##
## $trashed.yn
##   trashed.yn     col
```

```
## 1        yes #924900
## 2         no #ffff6d
##
## $inoc.dist.cm
##    inoc.dist.cm     col
## 1         -1.00   black
## 2          0.00 #006400
## 3          2.15 #257A25
## 4          2.77 #2F812F
## 5          4.09 #4A914A
## 6          4.30 #4F944F
## 7          4.64 #559755
## 8          5.54 #64A164
## 9          6.45 #74AB74
## 10         6.68 #7AAE7A
## 11         6.76 #7AAE7A
## 12         7.33 #84B484
## 13         8.18 #94BE94
## 14         8.31 #9AC19A
## 15         8.60 #9FC49F
## 16         8.77 #A4C8A4
## 17         9.28 #AACBAA
## 18         9.49 #AFCEAF
## 19         9.70 #B4D1B4
## 20        10.06 #B9D5B9
## 21        10.75 #C9DEC9
## 22        10.83 #C9DEC9
## 23        10.89 #C9DEC9
## 24        11.08 #CFE1CF
## 25        11.30 #CFE1CF
## 26        11.95 #DFEBDF
## 27        12.23 #E4EEE4
## 28        12.81 #EFF5EF
## 29        13.85 #FFFFFF
```

Basically turning sample_data table into colorcode_table here:

```
## ---- 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$s.type
        ) %>% select(s.id, s.type = col), # current table: [s.id, trashed.yn]
      left_join(
        ps.m %>% select(s.id, trashed.yn) %>% group_by(s.id) %>% distinct(),
        metadat.cols$trashed.yn
        ) %>% select(s.id, trashed.yn = col),
      join_by(s.id)
    ) # [s.id, s.type, trashed.yn]
```

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

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

```
tmp.j =
  full_join(
    tmp.j,
    left_join(
      ps.m %>% select(s.id, inoc.dist.cm) %>% group_by(s.id) %>% distinct(),
      metadat.cols$inoc.dist.cm
      ) %>% select(s.id, inoc.dist.cm = col),
    join_by(s.id)
    ) # [s.id, s.type, trashed.yn, inoc.dist]
```

```
## Joining with `by = join_by(inoc.dist.cm)`
```

```
# tmp.j =
#   full_join(
#     tmp.j,
#     left_join(
#       ps.m %>% select(s.id, extr.batch) %>% group_by(s.id) %>% distinct(),
#       metadat.cols$extr.batch
#       ) %>% select(s.id, extr.batch = col),
#     join_by(s.id)
#   ) # [s.id, s.type, trashed.yn, extr.batch, pcr.batch]
#
# tmp.j =
#   full_join(
#     tmp.j,
#     left_join(
#       ps.m %>% select(s.id, pcr.batch) %>% group_by(s.id) %>% distinct(),
#       metadat.cols$pcr.batch
#       ) %>% select(s.id, pcr.batch = col),
#     join_by(s.id)
#   ) # [s.id, s.type, trashed.yn, extr.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    trashed.yn inoc.dist.cm
## d4e23 "#db6d00" "#ffff6d"  "#9AC19A"
## d4e08 "#db6d00" "#924900"  "#84B484"
## d4e11 "#db6d00" "#ffff6d"  "#FFFFFF"
## d4e05 "#db6d00" "#ffff6d"  "#AFCEAF"
## d4e26 "#db6d00" "#ffff6d"  "#CFE1CF"
## d4e12 "#db6d00" "#924900"  "#A4C8A4"
## d4e21 "#db6d00" "#924900"  "#C9DEC9"
## d4e07 "#db6d00" "#924900"  "#559755"
## d4e16 "#db6d00" "#ffff6d"  "#AACBAA"
## d4e06 "#db6d00" "#ffff6d"  "#E4EEE4"
## d4e20 "#db6d00" "#ffff6d"  "#B4D1B4"
## d4e14 "#db6d00" "#ffff6d"  "#C9DEC9"
```

```
## d4e28 "#db6d00" "#ffff6d"  "#CFE1CF"
## d4e10 "#db6d00" "#924900"  "#94BE94"
## d4e01 "#db6d00" "#924900"  "#74AB74"
## d4e18 "#db6d00" "#924900"  "#7AAE7A"
## d4e17 "#920000" NA         "black"
## d4e13 "#db6d00" "#ffff6d"  "#B9D5B9"
## d4e24 "#db6d00" "#ffff6d"  "#DFEBDF"
```

Make plot legend for all your pretty colors:

```
## ---- 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,] "ants"          "#920000"
##  [4,] ""              "white"
##  [5,] "trashed.yn"    "white"
##  [6,] "yes"           "#924900"
##  [7,] "no"            "#ffff6d"
##  [8,] ""              "white"
##  [9,] "inoc.dist.cm"  "white"
## [10,] "-1"            "black"
## [11,] "0"             "#006400"
## [12,] "2.15"          "#257A25"
## [13,] "2.77"          "#2F812F"
## [14,] "4.09"          "#4A914A"
## [15,] "4.3"           "#4F944F"
## [16,] "4.64"          "#559755"
## [17,] "5.54"          "#64A164"
## [18,] "6.45"          "#74AB74"
## [19,] "6.68"          "#7AAE7A"
## [20,] "6.76"          "#7AAE7A"
## [21,] "7.33"          "#84B484"
## [22,] "8.18"          "#94BE94"
## [23,] "8.31"          "#9AC19A"
```

```
## [24,] "8.6"           "#9FC49F"
## [25,] "8.77"          "#A4C8A4"
## [26,] "9.28"          "#AACBAA"
## [27,] "9.49"          "#AFCEAF"
## [28,] "9.7"           "#B4D1B4"
## [29,] "10.06"         "#B9D5B9"
## [30,] "10.75"         "#C9DEC9"
## [31,] "10.83"         "#C9DEC9"
## [32,] "10.89"         "#C9DEC9"
## [33,] "11.08"         "#CFE1CF"
## [34,] "11.3"          "#CFE1CF"
## [35,] "11.95"         "#DFEBDF"
## [36,] "12.23"         "#E4EEE4"
## [37,] "12.81"         "#EFF5EF"
## [38,] "13.85"         "#FFFFFF"
## [39,] ""              "white"
```

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

```
# save.image()

## ---- plot ------------------------------------------------------------------------------- ##
# ps.m %>%
#   select(s.name, Abundance, Genus) %>%
#   tidyr::pivot_wider(names_from=Genus, values_from=Abundance) %>%
#   tibble::column_to_rownames("s.name") %>%
#   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 = 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
# )
```

**Decon5**

```
ps.m = psList.melt$decon5

## change NA's to "none" in spatial.layer
ps.m <-
  ps.m %>%
  mutate(
    spatial.layer = replace(spatial.layer, is.na(spatial.layer),"none")
  )

# ## try changing the 1-layer FG pieces as "top" instead of "bottom"
# ## H1, H2, H3, H4, H7, H20?, H28, H36
# tb.samples = c("H1", "H2", "H3", "H4", "H7", "H20", "H28", "H36")
# # welp only H20 is in here anyways
# ps.m %>% select(s.name, spatial.layer) %>% filter(s.name %in% tb.samples) %>% unique()
#
# ps.m <-
#   ps.m %>%
#   mutate(
#     spatial.layer = replace(spatial.layer, s.name %in% tb.samples, "bottom")
#   ) #%>% select(s.name, spatial.layer) %>% filter(s.name %in% tb.samples)
```

**Data wrangling...**

**Color wrangling...** Get color codes from desired palette: https://cran.r-project.org/web/packages/colorBlindness/vignettes/colorBlindness.html#How_to_use_this_package

```
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"
```

Choose your colors or set them randomly:

```
## ---- decon5 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","food (corn meal)"),
               col=c("#db6d00","#924900","#ffff6d"))), # set colors for s.type
```

```
       spatial.layer =
         as.data.frame(
           cbind(spatial.layer = c("top", "bottom","none"),
                 col=c("#ff6db6", "#6db6ff","black")))#,
       # extr.batch = ps.m %>%
       #   select(extr.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
## 3 food (corn meal) #ffff6d
##
## $spatial.layer
##   spatial.layer     col
## 1           top #ff6db6
## 2        bottom #6db6ff
## 3          none   black
```

Turning sample_data table into colorcode table:

```
## ---- 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$s.type
        ) %>% select(s.id, s.type = col), # current table: [s.id, s.type]
      left_join(
        ps.m %>% select(s.id, spatial.layer) %>% group_by(s.id) %>% distinct(),
        metadat.cols$spatial.layer
        ) %>% select(s.id, spatial.layer = col),
      join_by(s.id)
    ) # [s.id, s.type, spatial.layer]
```

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

```
# tmp.j =
#   full_join(
#     tmp.j,
#     left_join(
#       ps.m %>% select(s.id, extr.batch) %>% group_by(s.id) %>% distinct(),
#       metadat.cols$extr.batch
#       ) %>% select(s.id, extr.batch = col),
```

```
#     join_by(s.id)
#     ) # [s.id, s.type, spatial.layer, extr.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, spatial.layer, extr.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    spatial.layer
## d5e28 "#db6d00" "#6db6ff"
## d5e01 "#db6d00" "#ff6db6"
## d5e27 "#db6d00" "#ff6db6"
## d5e02 "#db6d00" "#6db6ff"
## d5e36 "#db6d00" "#6db6ff"
## d5e12 "#db6d00" "#6db6ff"
## d5e22 "#db6d00" "#ff6db6"
## d5e42 "#db6d00" "#ff6db6"
## d5e15 "#db6d00" "#6db6ff"
## d5e50 "#db6d00" "#6db6ff"
## d5e20 "#db6d00" "#6db6ff"
## d5e24 "#db6d00" "#6db6ff"
## d5e11 "#db6d00" "#ff6db6"
## d5e17 "#db6d00" "#6db6ff"
## d5e35 "#db6d00" "#6db6ff"
## d5e34 "#db6d00" "#ff6db6"
## d5e19 "#db6d00" "#6db6ff"
## d5e08 "#db6d00" "#ff6db6"
## d5e23 "#db6d00" "#ff6db6"
## d5e26 "#db6d00" "#6db6ff"
## d5e33 "#db6d00" "#6db6ff"
## d5e32 "#db6d00" "#ff6db6"
## d5e46 "#db6d00" "#ff6db6"
## d5e45 "#db6d00" "#6db6ff"
## d5e52 "#db6d00" "#ff6db6"
## d5e43 "#db6d00" "#ff6db6"
## d5e16 "#db6d00" "#ff6db6"
## d5e51 "#db6d00" "#6db6ff"
## d5e10 "#db6d00" "#6db6ff"
## d5e40 "#db6d00" "#ff6db6"
## d5e05 "#db6d00" "#6db6ff"
```

```
## d5e25 "#db6d00" "#6db6ff"
## d5e41 "#db6d00" "#6db6ff"
## d5e37 "#db6d00" "#6db6ff"
## d5e30 "#db6d00" "#ff6db6"
## d5e21 "#db6d00" "#6db6ff"
## d5e13 "#db6d00" "#6db6ff"
## d5e14 "#db6d00" "#6db6ff"
## d5e47 "#db6d00" "#ff6db6"
## d5e38 "#db6d00" "#ff6db6"
## d5e39 "#db6d00" "#ff6db6"
## d5e04 "#db6d00" "#ff6db6"
## d5e06 "#db6d00" "#6db6ff"
## d5e49 "#db6d00" "#6db6ff"
## d5e09 "#db6d00" "#6db6ff"
## d5e48 "#db6d00" "#6db6ff"
## d5e44 "#db6d00" "#6db6ff"
## d5e07 "#db6d00" "#6db6ff"
## d5e03 "#924900" "black"
```

Create plot legend for all your pretty colors:

```
## ---- 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,] "food (corn meal)" "#ffff6d"
##  [5,] ""                 "white"
##  [6,] "spatial.layer"    "white"
##  [7,] "top"              "#ff6db6"
##  [8,] "bottom"           "#6db6ff"
##  [9,] "none"             "black"
## [10,] ""                 "white"
```

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

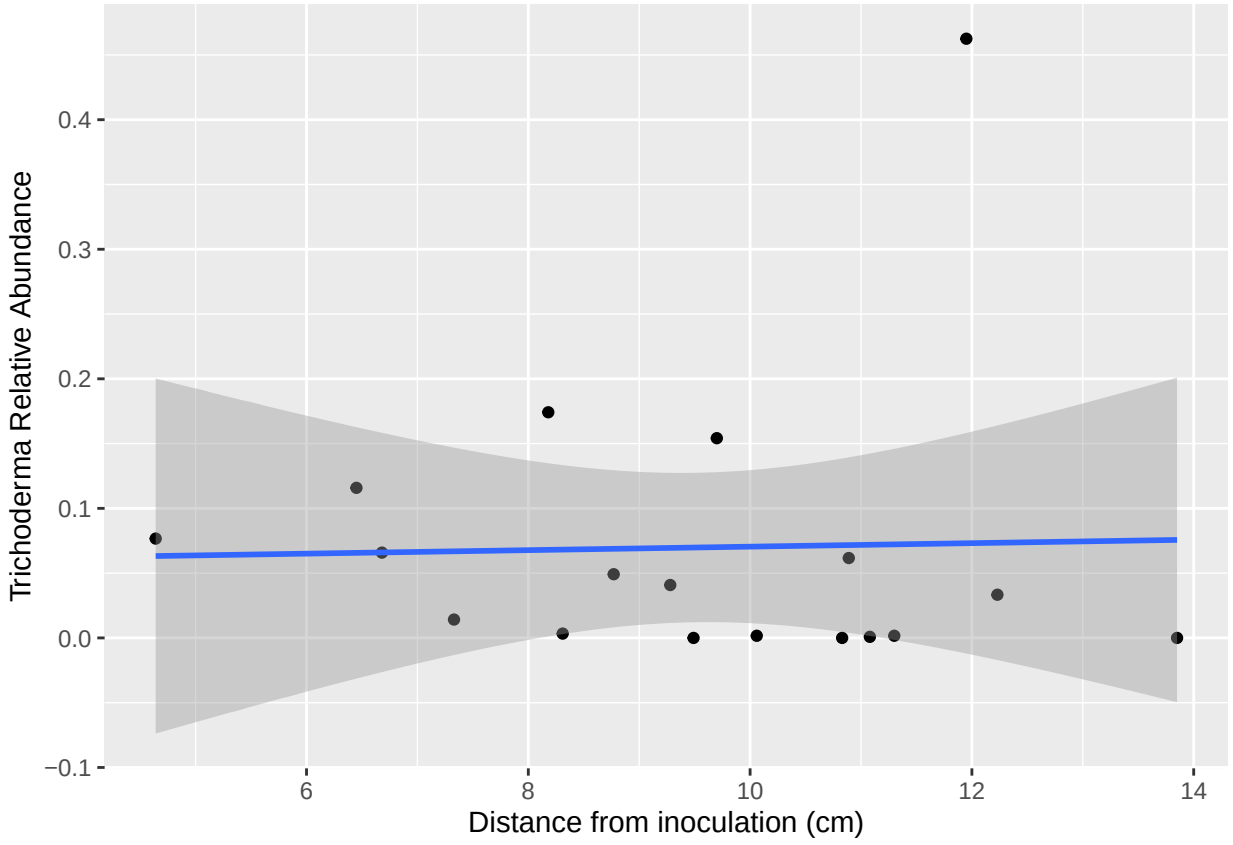
```
# save.image()
# ## ---- plot ------------------------------------------------------------------------------- ##
# ps.m %>%
#   select(s.name, Abundance, Genus) %>%
#   tidyr::pivot_wider(names_from=Genus, values_from=Abundance) %>%
#   tibble::column_to_rownames("s.name") %>%
#   as.matrix() %>%
#   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 = 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
# )
```

### Corr: %Trich x Inoc Dist

Does distance from inoculation actually correlate with %Trichoderma? Nope :(

```
ps.m = psList.melt$decon4
ps.m %>% filter(Genus == "g__Trichoderma") %>%
  filter(s.type == "fungus garden") %>%
  group_by(s.id) %>%
  select(s.id,Abundance,inoc.dist.cm,Genus) %>%
  ggplot(
    aes(x=inoc.dist.cm, y=Abundance)
  ) +
  geom_point() +
  # stat_summary(fun.data=mean_cl_normal) +
  geom_smooth(method='lm', formula= y~x) +
```

```
  ylab("Trichoderma Relative Abundance") +
  xlab("Distance from inoculation (cm)")
```




```
my_data =
  ps.m %>% filter(Genus == "g__Trichoderma") %>%
  filter(s.type == "fungus garden") %>%
  group_by(s.id) %>%
  select(s.id,Abundance,inoc.dist.cm,Genus)

shapiro.test(my_data$inoc.dist.cm)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  my_data$inoc.dist.cm
## W = 0.9891, p-value = 0.9979
```

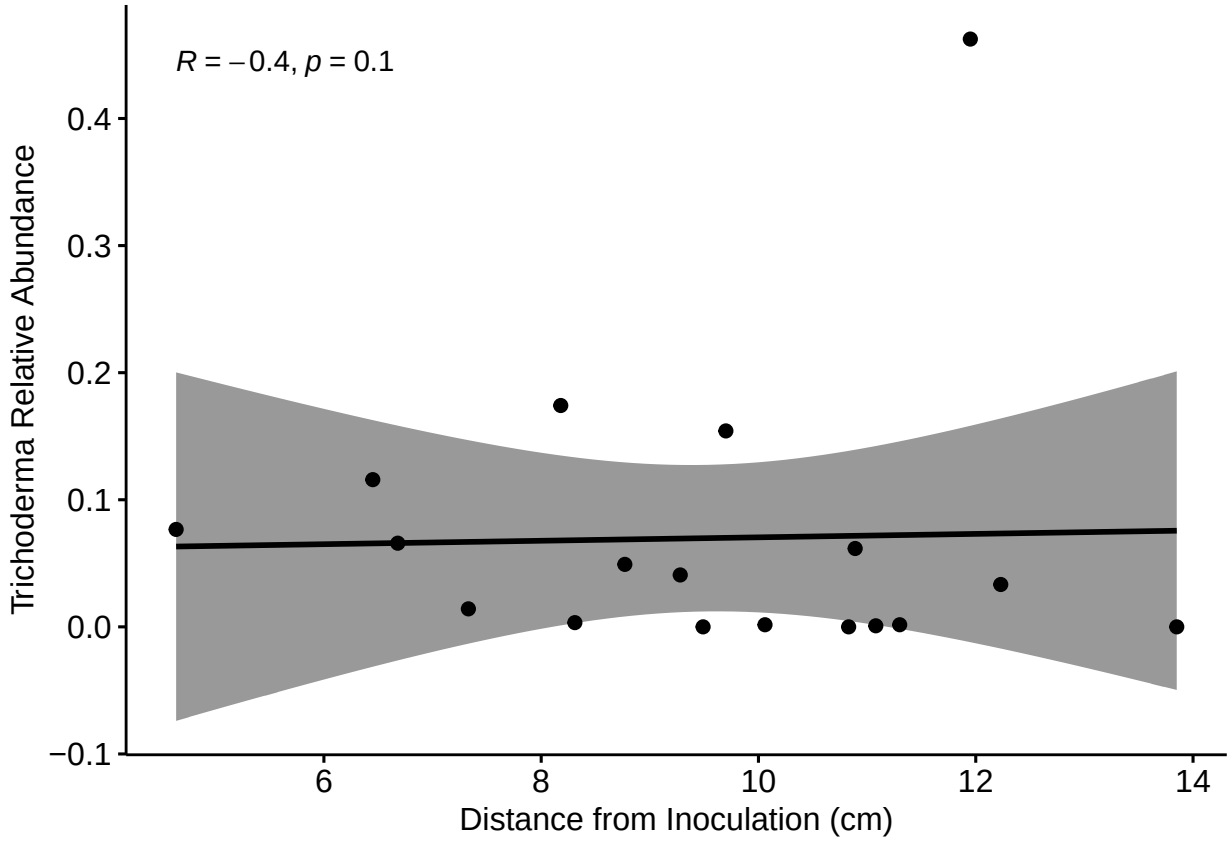
```
shapiro.test(my_data$Abundance)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  my_data$Abundance
## W = 0.64751, p-value = 2.082e-05
```

```
#   Shapiro-Wilk normality test
# data:  my_data$inoc.dist.cm
# W = 0.9891, p-value = 0.9979
```

```
#   Shapiro-Wilk normality test
# data:  my_data$Abundance
# W = 0.64751, p-value = 2.082e-05    # not normal :(
```

```
ggscatter(my_data,
          x = "inoc.dist.cm", y = "Abundance",
          add = "reg.line", conf.int = TRUE,
          cor.coef = TRUE, cor.method = "spearman",
          xlab = "Distance from Inoculation (cm)",
          ylab = "Trichoderma Relative Abundance")
```




```
cor.test(my_data$inoc.dist.cm, my_data$Abundance,  method = "spearman")
```

```
## Warning in cor.test.default(my_data$inoc.dist.cm, my_data$Abundance, method =
## "spearman"): Cannot compute exact p-value with ties
```

```
##
##  Spearman's rank correlation rho
##
## data:  my_data$inoc.dist.cm and my_data$Abundance
## S = 1353, p-value = 0.1035
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## -0.3962765
```

## Alpha Diversity

Helpful tutorials: https://www.bioconductor.org/packages/release/bioc/vignettes/phyloseq/inst/doc/phyloseq-analysis.html#easy-richness-estimates

https://joey711.github.io/phyloseq/plot_richness-examples.html

### d4

```
ps = psList.filt$decon4
sample_data(ps) = sdList$decon4

trashed.yn.col.ls = list( "yes" ="#924900", "no" = "#ffff6d" )
full.inoc.dist.cols
```

```
##    inoc.dist.cm     col
## 1         -1.00   black
## 2          0.00 #006400
## 3          2.15 #257A25
## 4          2.77 #2F812F
## 5          4.09 #4A914A
## 6          4.30 #4F944F
## 7          4.64 #559755
## 8          5.54 #64A164
## 9          6.45 #74AB74
## 10         6.68 #7AAE7A
## 11         6.76 #7AAE7A
## 12         7.33 #84B484
## 13         8.18 #94BE94
## 14         8.31 #9AC19A
## 15         8.60 #9FC49F
## 16         8.77 #A4C8A4
## 17         9.28 #AACBAA
## 18         9.49 #AFCEAF
## 19         9.70 #B4D1B4
## 20        10.06 #B9D5B9
## 21        10.75 #C9DEC9
## 22        10.83 #C9DEC9
## 23        10.89 #C9DEC9
## 24        11.08 #CFE1CF
## 25        11.30 #CFE1CF
## 26        11.95 #DFEBDF
## 27        12.23 #E4EEE4
## 28        12.81 #EFF5EF
## 29        13.85 #FFFFFF
```

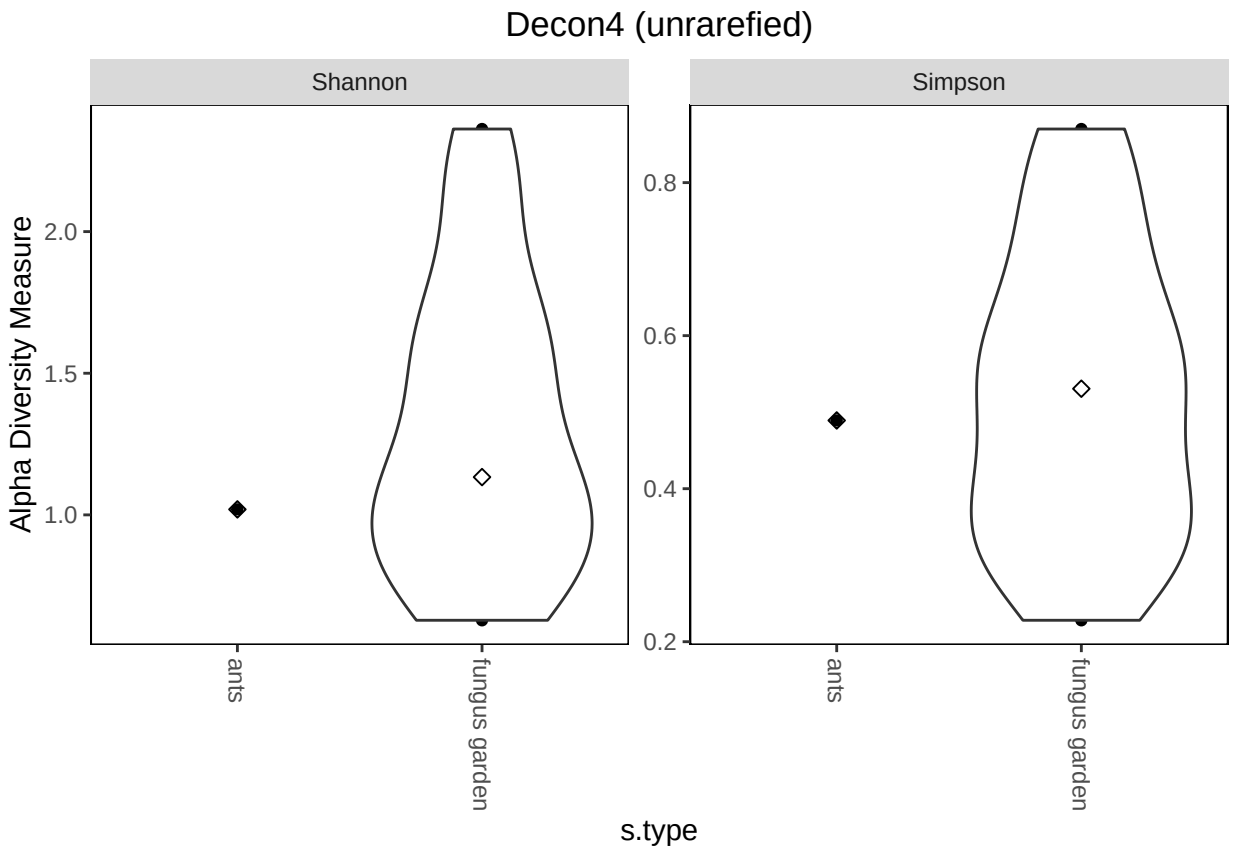
```
ps %>%
  plot_richness(., x = "s.type",
                measures = c("Shannon", "Simpson"),
  )  +
  geom_violin(

  ) +
  ggtitle("Decon4 (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**

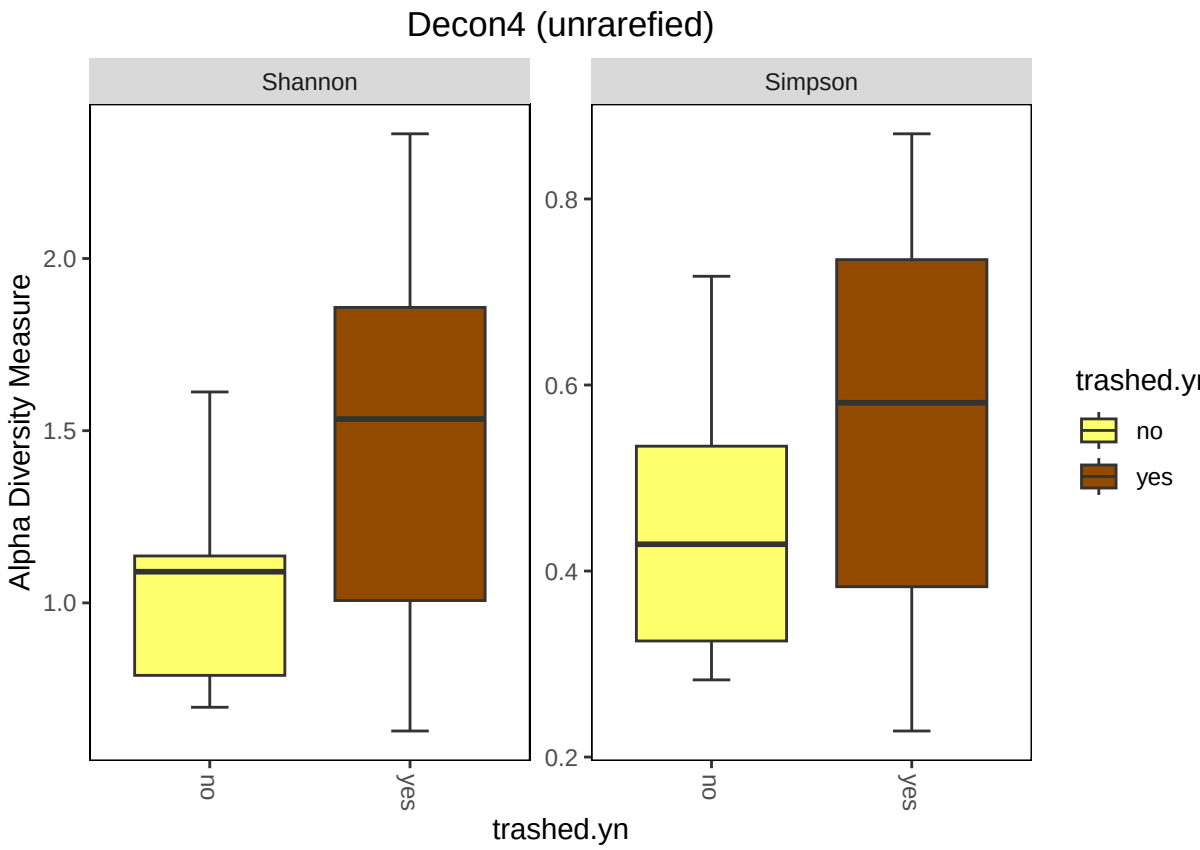
```
## Warning: The `fun.y` argument of `stat_summary()` is deprecated as of ggplot2 3.3.0.
## i Please use the `fun` argument instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

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




```
p <-
  ps %>%
```

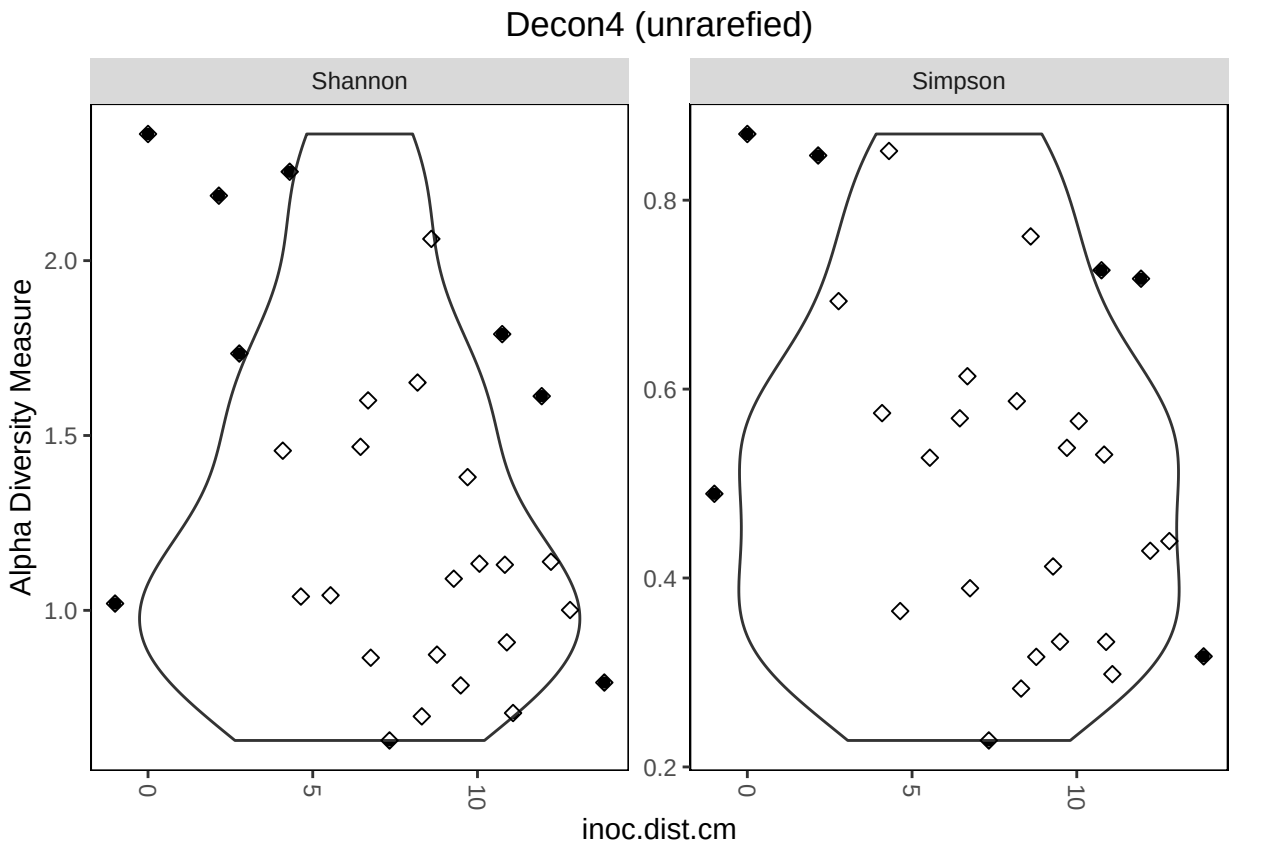
```
  subset_samples(s.type == "fungus garden") %>%
  plot_richness(., x = "trashed.yn",
                measures = c("Shannon", "Simpson"),
  )  +
  geom_boxplot(
    aes(fill = trashed.yn),
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25

  ) +
  scale_fill_manual(
    values = unlist(trashed.yn.col.ls)
  ) +
  ggtitle("Decon4 (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

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




**violin-trashed.yn**

#### violin-inoc.dist.cm

```
p <-
  ps %>%
  plot_richness(., x = "inoc.dist.cm",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_violin(
  ) +
  ggtitle("Decon4 (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
  )
# p$data$s.type = factor(p$data$s.type, levels=c("fungus garden","trash","food (leaves)"))
p
```




```
#Create dataframe of alpha diversity values for anova stats
```

```r
ps.adiv.df = estimate_richness(ps, measures = c("Shannon", "Simpson"))
ps.adiv.df$s.type = sample_data(ps)$s.type
ps.adiv.df$trashed.yn = sample_data(ps)$trashed.yn
ps.adiv.df$inoc.dist.cm = sample_data(ps)$inoc.dist.cm

#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 0.0819 0.081878  0.3147 0.5796
## Residuals 26 6.7644 0.260170
##
## $sim
## Analysis of Variance Table
##
## Response: Simpson
##           Df  Sum Sq  Mean Sq F value Pr(>F)
## s.type     1 0.00109 0.001089  0.0301 0.8637
## Residuals 26 0.94163 0.036216
```

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

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##            Df Sum Sq Mean Sq F value  Pr(>F)
## trashed.yn  1 1.3340 1.33399  6.1413 0.02031 *
## Residuals  25 5.4304 0.21722
## ---
## 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)
## trashed.yn  1 0.12105 0.121048  3.6879 0.06629 .
## Residuals  25 0.82058 0.032823
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Run anova on both Shannon and Simpson for inoc.dist
# (idk how much sense alpha div makes with continuous variable...)
ps.adiv.df.fg = ps.adiv.df %>% filter(s.type == "fungus garden")
inoc.dist.cm.aov = list(shan = anova(aov(Shannon ~ inoc.dist.cm, ps.adiv.df.fg)),
                sim = anova(aov(Simpson ~ inoc.dist.cm, ps.adiv.df.fg)))
inoc.dist.cm.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##              Df Sum Sq Mean Sq F value   Pr(>F)
## inoc.dist.cm  1 2.0600 2.06000  10.947 0.002844 **
## Residuals    25 4.7044 0.18818
## ---
## 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)
## inoc.dist.cm  1 0.23447 0.234467  8.2891 0.008058 **
## Residuals    25 0.70716 0.028286
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
TukeyHSD(aov(Shannon ~ trashed.yn, ps.adiv.df.fg))
```

**TukeyHSD**

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Shannon ~ trashed.yn, data = ps.adiv.df.fg)
##
## $trashed.yn
##             diff        lwr       upr     p adj
## yes-no 0.4523781 0.07641748 0.8283388 0.0203135
```

```r
TukeyHSD(aov(Simpson ~ trashed.yn, ps.adiv.df.fg))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Simpson ~ trashed.yn, data = ps.adiv.df.fg)
##
## $trashed.yn
##             diff         lwr       upr     p adj
## yes-no 0.1362714 -0.00987421 0.2824169 0.0662863
```

```r
# TukeyHSD(aov(Shannon ~ inoc.dist.cm, ps.adiv.df.fg))
# TukeyHSD(aov(Simpson ~ inoc.dist.cm, ps.adiv.df.fg))
```

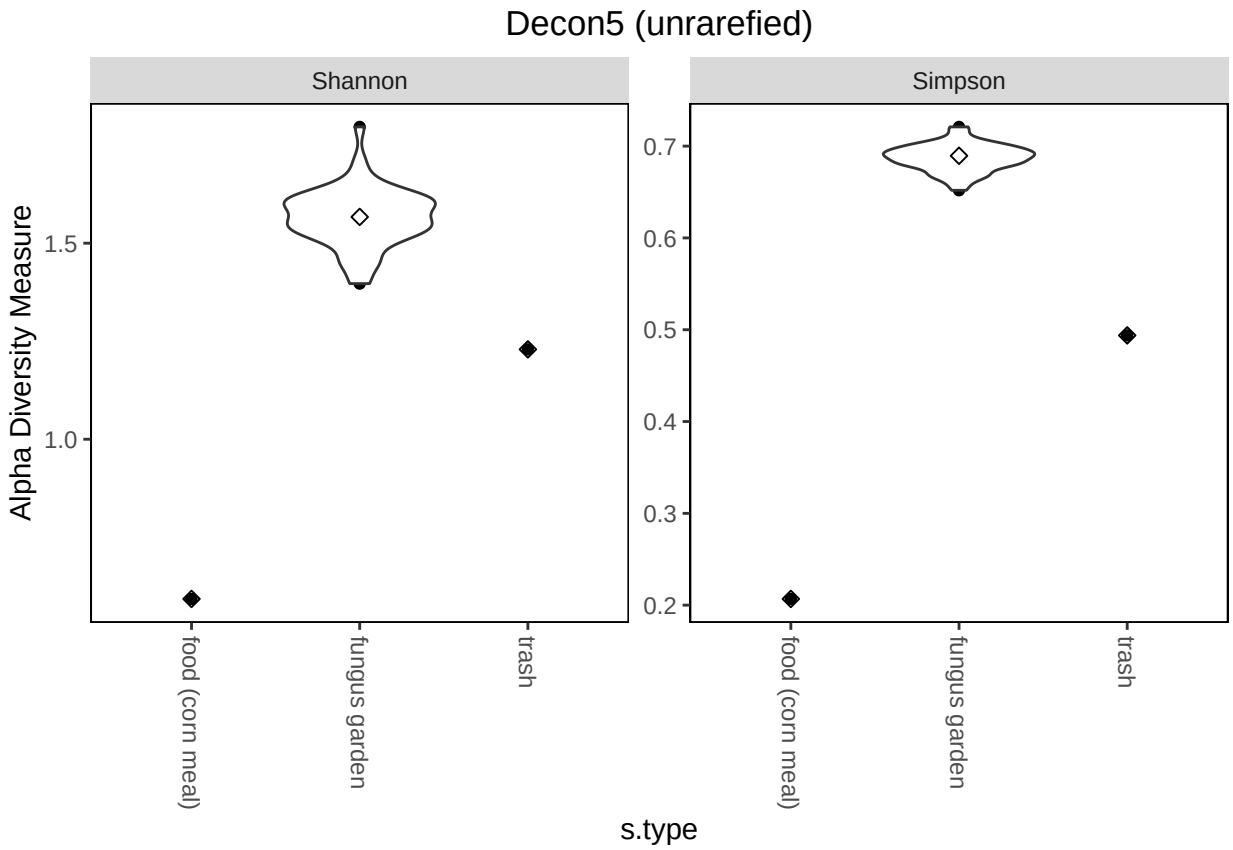
**d5**

```
ps = psList.filt$decon5
sample_data(ps) = sdList$decon5
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("Decon5 (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
  )
# p$data$s.type = factor(p$data$s.type, levels=c("fungus garden","trash","food (leaves)"))
p
```

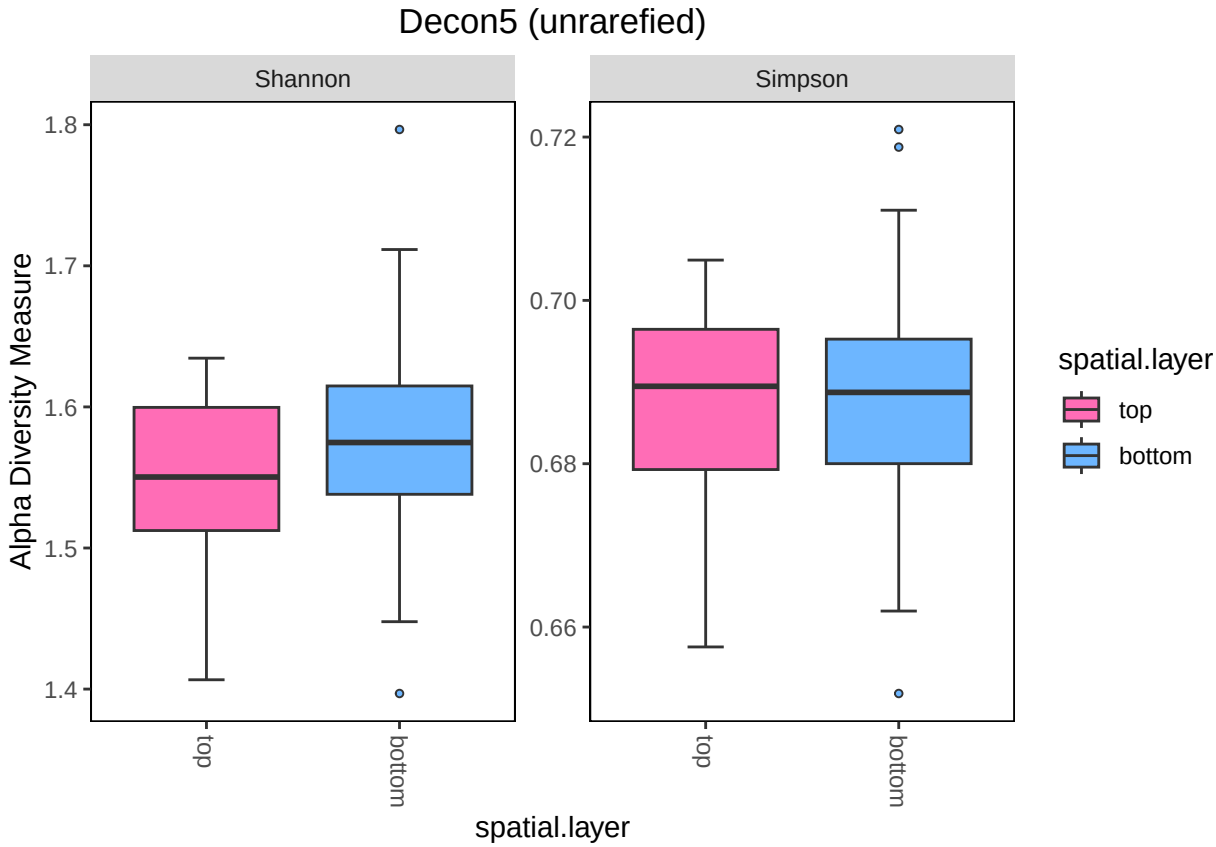
**violin-s.type**

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




#### violin-layer

```
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("Decon5 (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","bottom"))
p$layers = p$layers[-1]
p
```




#### anova

```
#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)),
                  sim = anova(aov(Simpson ~ s.type, ps.adiv.df)))
s.type.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## s.type     2 1.02763 0.51382  92.818 < 2.2e-16 ***
## Residuals 48 0.26572 0.00554
## ---
## 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     2 0.259592 0.129796  603.44 < 2.2e-16 ***
```

```
## Residuals 48 0.010324 0.000215
## ---
## 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")
spatial.layer.aov = list(shan = anova(aov(Shannon ~ spatial.layer, ps.adiv.df.fg)),
                 sim = anova(aov(Simpson ~ spatial.layer, ps.adiv.df.fg)))
spatial.layer.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##               Df   Sum Sq   Mean Sq F value Pr(>F)
## spatial.layer  1 0.012681 0.0126808  2.3554 0.1316
## Residuals     47 0.253035 0.0053837
##
## $sim
## Analysis of Variance Table
##
## Response: Simpson
##               Df    Sum Sq    Mean Sq F value Pr(>F)
## spatial.layer  1 0.0000347 3.4701e-05  0.1585 0.6923
## Residuals     47 0.0102897 2.1893e-04
```

## Beta Diversity

### d4

```
ps = fin.psList.raref$decon4
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")
```

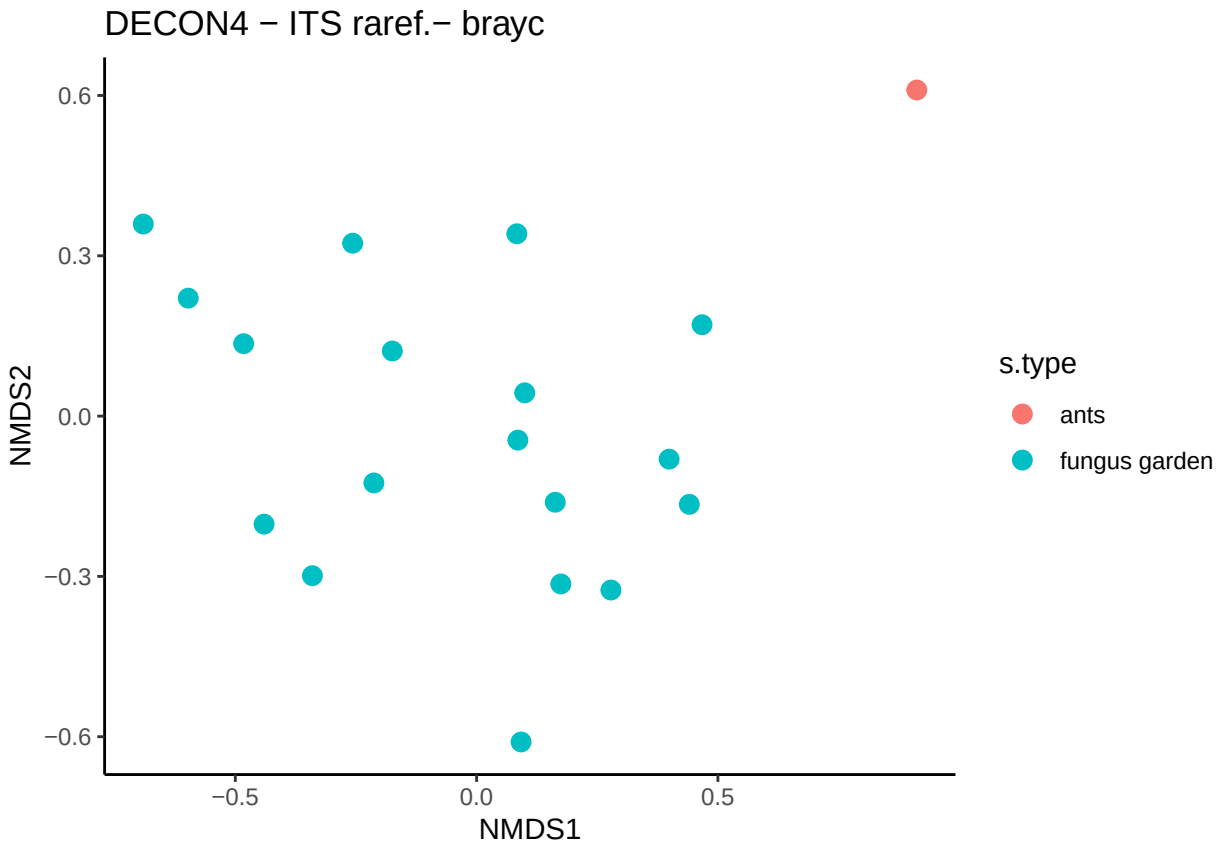
```
# Bray Curtis NMDS
ps.bcord = ordinate(ps, method= "NMDS", distance = "bray")
```

### d4BC

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.07723389
## Run 1 stress 0.1426242
## Run 2 stress 0.07723389
## ... Procrustes: rmse 2.690529e-06  max resid 8.227464e-06
## ... Similar to previous best
## Run 3 stress 0.07723389
## ... Procrustes: rmse 1.092856e-05  max resid 3.749479e-05
## ... Similar to previous best
## Run 4 stress 0.0772339
## ... Procrustes: rmse 1.084302e-05  max resid 3.171348e-05
## ... Similar to previous best
```

```
## Run 5 stress 0.1406008
## Run 6 stress 0.07723392
## ... Procrustes: rmse 1.900432e-05  max resid 5.472114e-05
## ... Similar to previous best
## Run 7 stress 0.07723389
## ... New best solution
## ... Procrustes: rmse 4.019691e-06  max resid 1.357103e-05
## ... Similar to previous best
## Run 8 stress 0.07723389
## ... Procrustes: rmse 8.206664e-06  max resid 2.826057e-05
## ... Similar to previous best
## Run 9 stress 0.07723389
## ... Procrustes: rmse 2.162423e-05  max resid 7.491857e-05
## ... Similar to previous best
## Run 10 stress 0.1426241
## Run 11 stress 0.0772339
## ... Procrustes: rmse 1.415094e-05  max resid 4.208332e-05
## ... Similar to previous best
## Run 12 stress 0.07723389
## ... Procrustes: rmse 3.999902e-06  max resid 1.340157e-05
## ... Similar to previous best
## Run 13 stress 0.07723389
## ... Procrustes: rmse 2.723637e-05  max resid 9.479628e-05
## ... Similar to previous best
## Run 14 stress 0.07723389
## ... Procrustes: rmse 5.180246e-06  max resid 1.764572e-05
## ... Similar to previous best
## Run 15 stress 0.07723389
## ... Procrustes: rmse 1.352981e-05  max resid 4.731958e-05
## ... Similar to previous best
## Run 16 stress 0.07723389
## ... Procrustes: rmse 1.14109e-05  max resid 3.787273e-05
## ... Similar to previous best
## Run 17 stress 0.07723389
## ... Procrustes: rmse 3.099945e-06  max resid 1.082504e-05
## ... Similar to previous best
## Run 18 stress 0.1612263
## Run 19 stress 0.07723389
## ... Procrustes: rmse 4.434205e-06  max resid 1.499831e-05
## ... Similar to previous best
## Run 20 stress 0.07723389
## ... Procrustes: rmse 4.980469e-06  max resid 1.554e-05
## ... Similar to previous best
## *** Best solution repeated 12 times
```

```
plot_ordination( ps,
                 ps.bcord,
                 color = "s.type",
                 # shape = "s.type"
                 ) +
  geom_point(size=3) +
  ggtitle("DECON4 - ITS raref.- brayc") +
  theme_classic()
```




```
# ## fungus garden samples only
# # Bray Curtis NMDS
# ps.fg.bcord = ordinate(ps.fg, method= "NMDS", distance = "bray")
# plot_ordination( ps.fg,
#                  ps.fg.bcord,
#                  color = "trashed.yn",
#                  # label = "s.id"
#                  ) +
#   geom_point(size=3) +
#   ggtitle("DECON4 - ITS raref.- brayc") +
#   theme_classic()

# ## fungus garden samples only
# # Bray Curtis NMDS
# ps.fg.bcord = ordinate(ps.fg, method= "NMDS", distance = "bray")
# plot_ordination( ps.fg,
#                  ps.fg.bcord,
#                  color = "inoc.dist.cm",
#                  # label = "s.id"
#                  ) +
#   geom_point(size=3) +
#   ggtitle("DECON4 - ITS raref.- brayc") +
#   theme_classic()

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

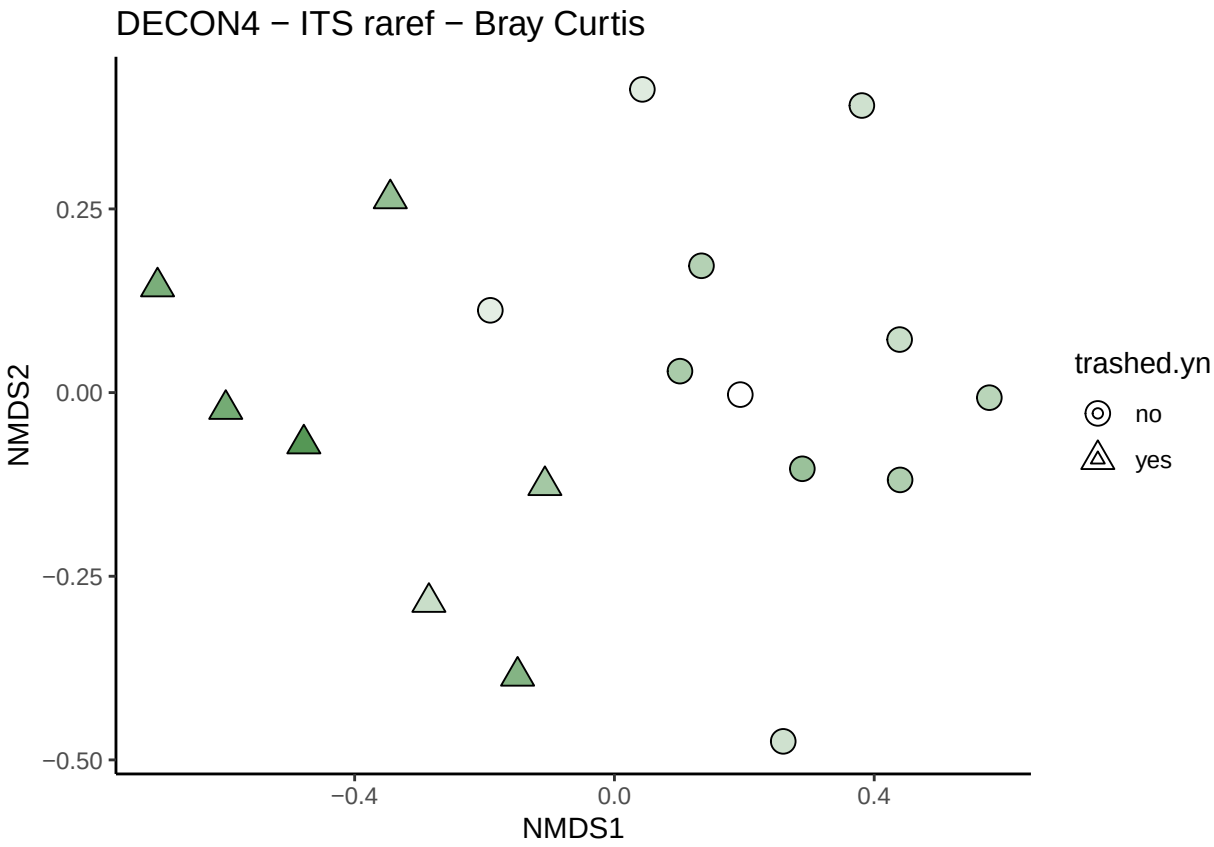
```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.09647714
## Run 1 stress 0.1548914
## Run 2 stress 0.1623778
## Run 3 stress 0.155842
## Run 4 stress 0.1563038
## Run 5 stress 0.155842
## Run 6 stress 0.09647714
## ... New best solution
## ... Procrustes: rmse 6.270757e-06  max resid 2.0097e-05
## ... Similar to previous best
## Run 7 stress 0.09647714
## ... New best solution
## ... Procrustes: rmse 1.104798e-06  max resid 3.197572e-06
## ... Similar to previous best
## Run 8 stress 0.09647714
## ... Procrustes: rmse 5.157969e-06  max resid 1.449314e-05
## ... Similar to previous best
## Run 9 stress 0.2853333
## Run 10 stress 0.09647714
## ... New best solution
## ... Procrustes: rmse 4.714454e-06  max resid 1.461879e-05
## ... Similar to previous best
## Run 11 stress 0.1642094
## Run 12 stress 0.09647714
## ... Procrustes: rmse 2.045594e-06  max resid 6.073614e-06
## ... Similar to previous best
## Run 13 stress 0.2627748
## Run 14 stress 0.1553217
## Run 15 stress 0.09647714
## ... Procrustes: rmse 8.911476e-06  max resid 2.76383e-05
## ... Similar to previous best
## Run 16 stress 0.09647714
## ... Procrustes: rmse 1.64706e-06  max resid 4.410806e-06
## ... Similar to previous best
## Run 17 stress 0.1553217
## Run 18 stress 0.155842
## Run 19 stress 0.1553217
## Run 20 stress 0.09647714
## ... New best solution
## ... Procrustes: rmse 1.167744e-06  max resid 2.773778e-06
## ... Similar to previous best
## *** Best solution repeated 1 times
```

```
p <-
  plot_ordination( ps.fg,
                   ps.fg.bcord,
                   shape = "trashed.yn",
                   color = "black"
  ) +
  geom_point(
    aes( fill = as.factor(inoc.dist.cm)),
    size=4,
```

```
  ) +
  scale_shape_manual(
    values = c(21,24)
  ) +
  scale_fill_manual(
    values = setNames(full.inoc.dist.cols$col, as.factor(full.inoc.dist.cols$inoc.dist.cm)),
    guide = "none"
  ) +
  ggtitle("DECON4 - ITS raref - Bray Curtis") +
  theme_classic()
```

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

```
p
```




```
# 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  0.68055 0.59002 24.465   0.06 .
```

```
## Residual 17  0.47288 0.40998
## Total    18  1.15343 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

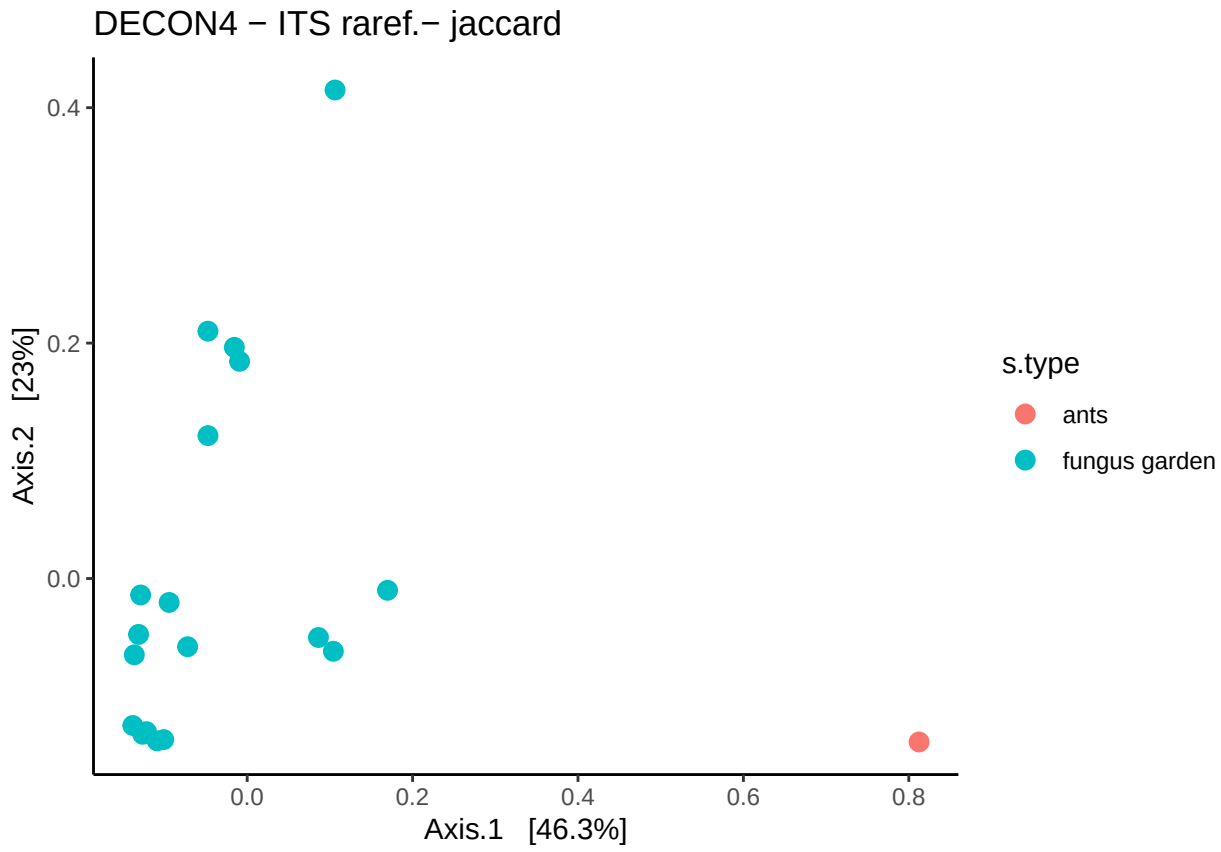
```
# Permanova - FG only
ps.fg.bcdist = distance(ps.fg, method ="bray")
adonis2( ps.fg.bcdist ~ trashed.yn,
         data = ps.fg.samdat.df
)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.fg.bcdist ~ trashed.yn, data = ps.fg.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1  0.04895 0.10351 1.8475  0.115
## Residual 16  0.42393 0.89649
## Total    17  0.47288 1.00000
```

```
adonis2( ps.fg.bcdist ~ inoc.dist.cm,
         data = ps.fg.samdat.df
)
```

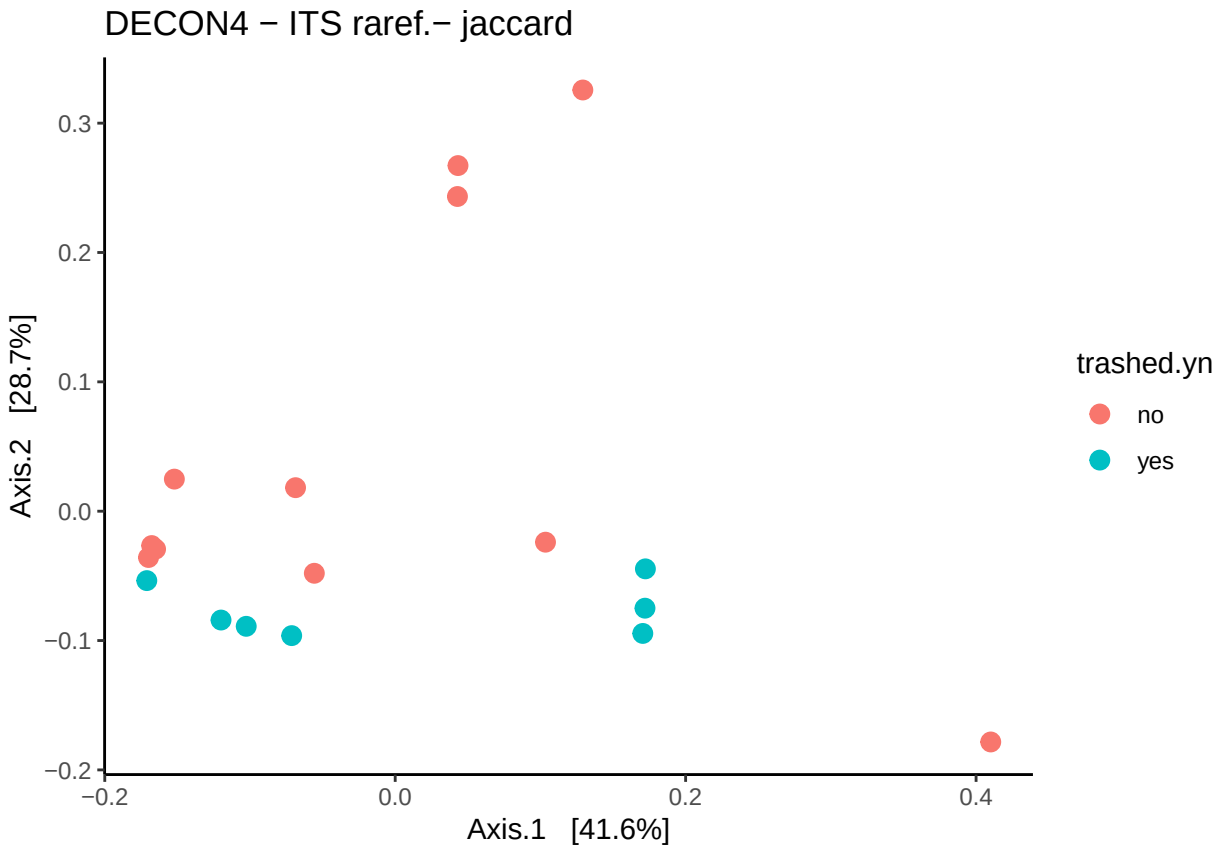
```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.fg.bcdist ~ inoc.dist.cm, data = ps.fg.samdat.df)
##          Df SumOfSqs     R2      F Pr(>F)
## Model     1  0.02568 0.0543 0.9187   0.45
## Residual 16  0.44720 0.9457
## Total    17  0.47288 1.0000
```

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

plot_ordination( ps,
                 ps.jccord,
                 color = "s.type",
                 # shape = "s.type"
                 ) +
  geom_point(size=3) +
  ggtitle("DECON4 - ITS raref.- jaccard") +
  theme_classic()
```

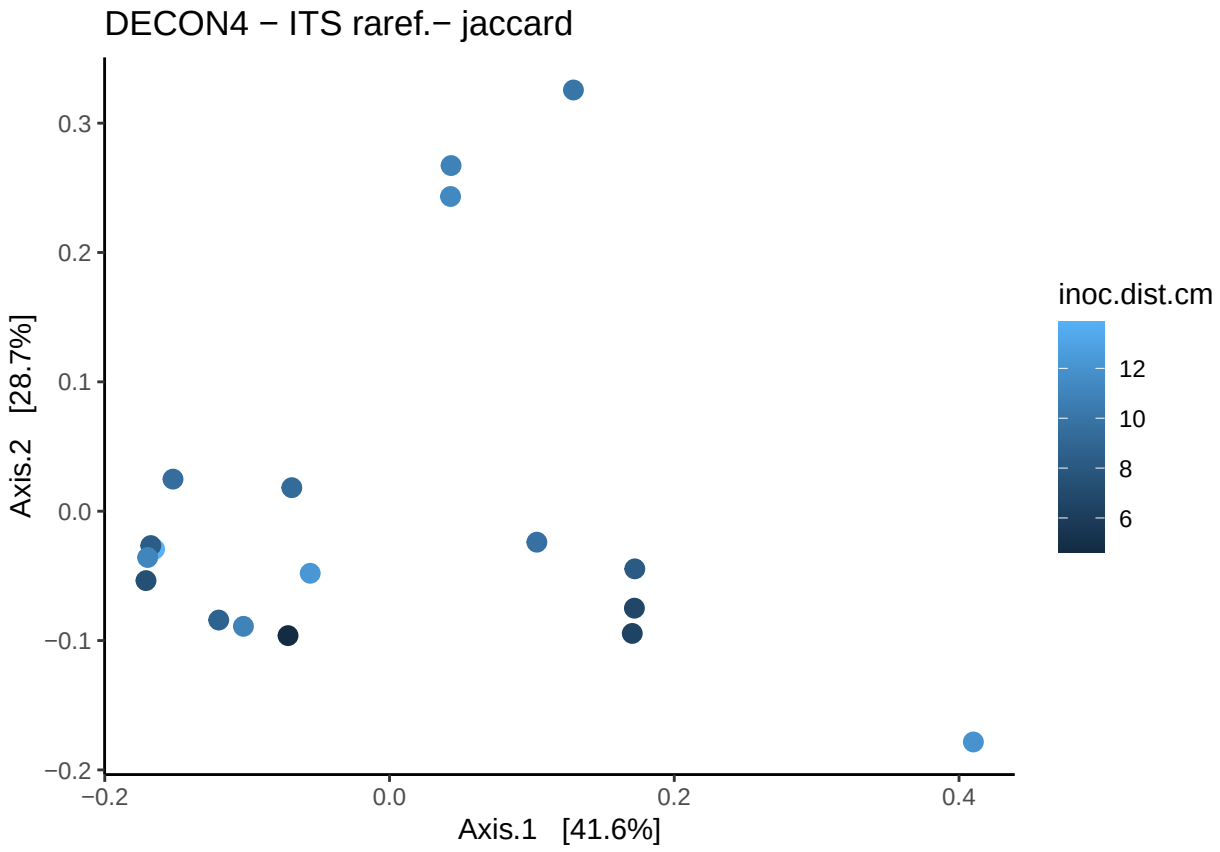



**d4JCC**

```
## fungus garden samples only
# Jaccard PCoA
ps.fg.jccord = ordinate(ps.fg, method= "PCoA", distance = "jaccard")
plot_ordination( ps.fg,
                 ps.fg.jccord,
                 color = "trashed.yn",
                 # label = "s.id"
                 ) +
  geom_point(size=3) +
  ggtitle("DECON4 - ITS raref.- jaccard") +
  theme_classic()
```




```
## fungus garden samples only
# Jaccard PCoA
ps.fg.bcord = ordinate(ps.fg, method= "PCoA", distance = "jaccard")
plot_ordination( ps.fg,
                 ps.fg.jccord,
                 color = "inoc.dist.cm",
                 # label = "s.id"
                 ) +
  geom_point(size=3) +
  ggtitle("DECON4 - ITS raref.- jaccard") +
  theme_classic()
```

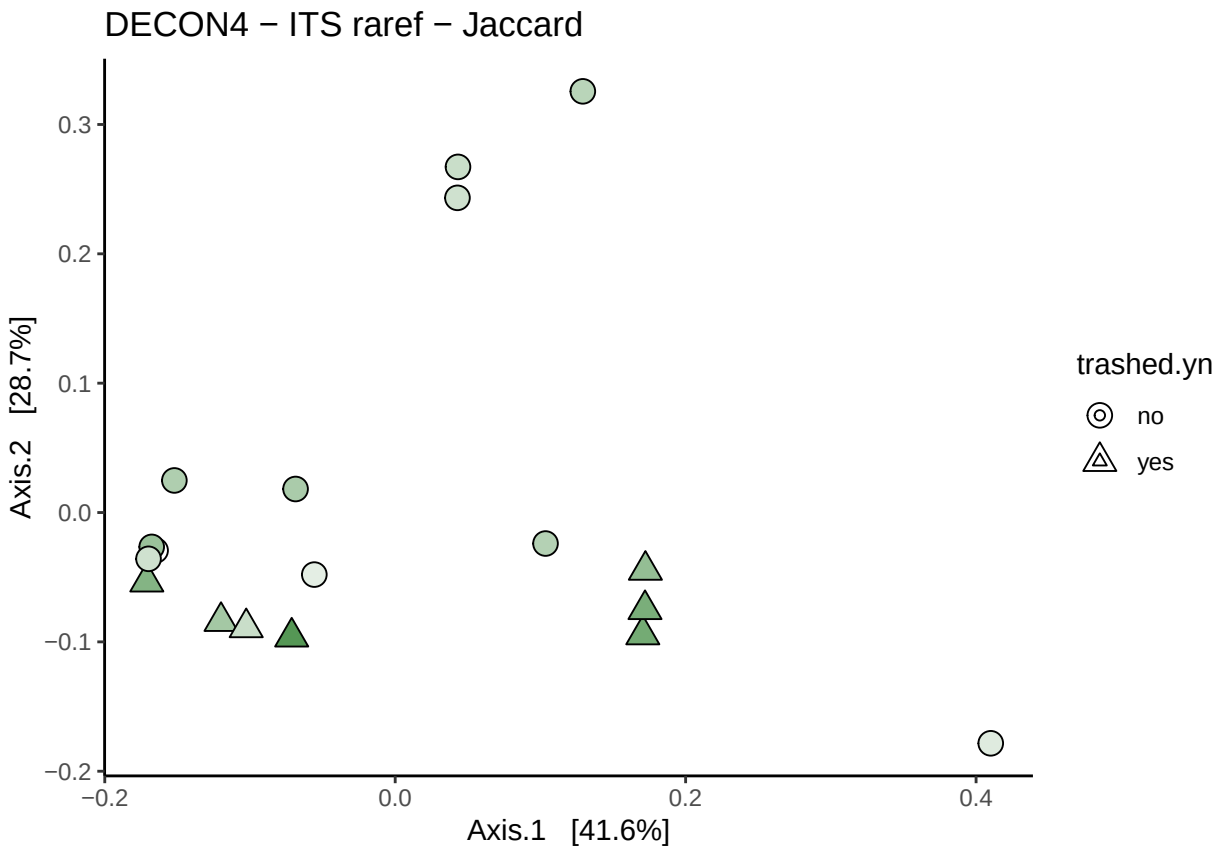



```
ps.fg.jccord = ordinate(ps.fg, method= "PCoA", distance = "jaccard")
p <-
  plot_ordination( ps.fg,
                   ps.fg.jccord,
                   shape = "trashed.yn",
                   color = "black"
  ) +
  geom_point(
    aes( fill = as.factor(inoc.dist.cm)),
    size=4,
  ) +
  scale_shape_manual(
    values = c(21,24)
  ) +
  scale_fill_manual(
    values = setNames(full.inoc.dist.cols$col, as.factor(full.inoc.dist.cols$inoc.dist.cm)),
    guide = "none"
  ) +
  ggtitle("DECON4 - ITS raref - Jaccard") +
  theme_classic()
```

**d4JCC trashed.yn and inoc.dist**

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

```
p
```




```
# 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   0.7497 0.40289 11.47  0.063 .
## Residual 17   1.1111 0.59711
## Total    18   1.8608 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
```

```
## adonis2(formula = ps.fg.jccdist ~ trashed.yn, data = ps.fg.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1  0.11987 0.10788 1.9348    0.1
## Residual 16  0.99125 0.89212
## Total    17  1.11112 1.00000
```

```
adonis2( ps.fg.jccdist ~ inoc.dist.cm,
         data = ps.fg.samdat.df
)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.fg.jccdist ~ inoc.dist.cm, data = ps.fg.samdat.df)
##          Df SumOfSqs     R2      F Pr(>F)
## Model     1  0.07056 0.0635 1.0849  0.344
## Residual 16  1.04056 0.9365
## Total    17  1.11112 1.0000
```

**d5**

```
ps = fin.psList.raref$decon5
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")
```

**d5BC**

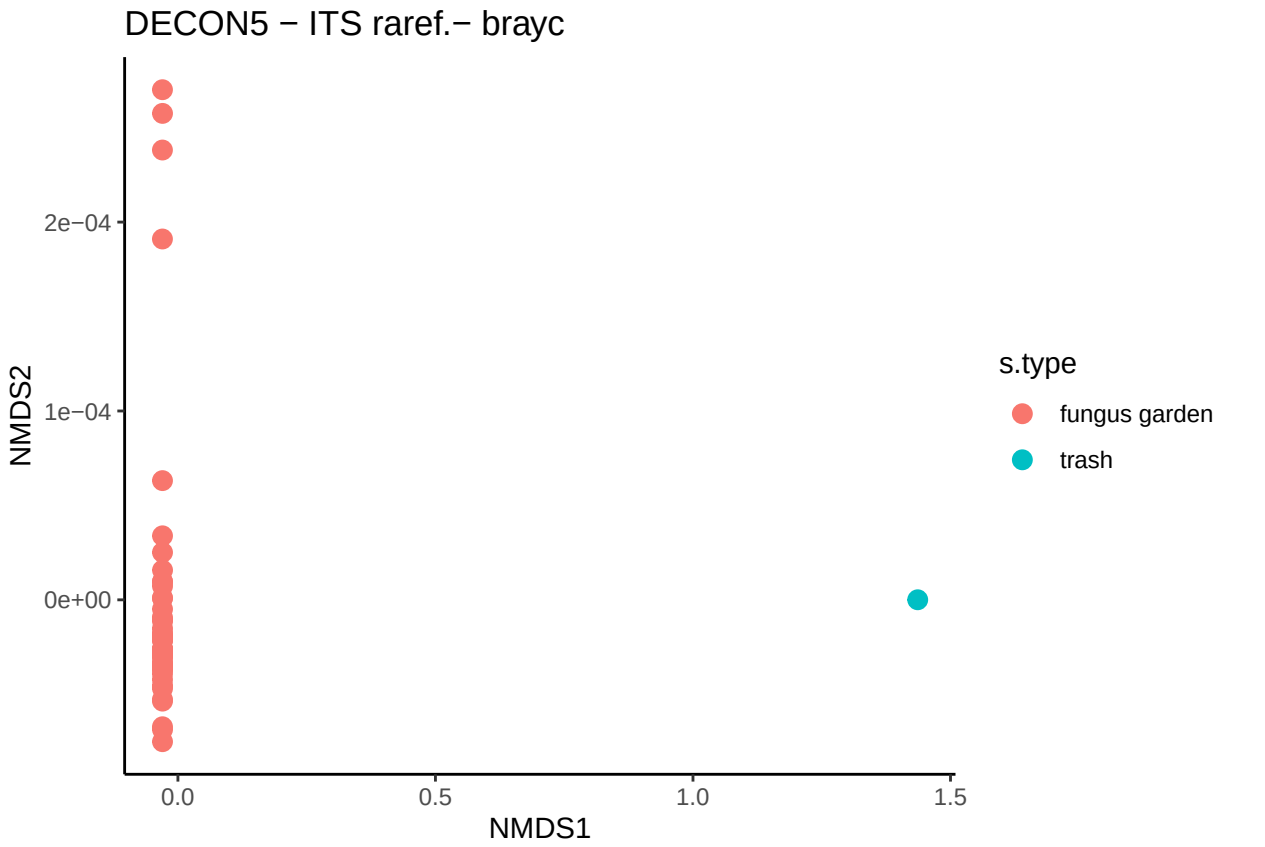
```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 9.953293e-05
## Run 1 stress 9.692247e-05
## ... New best solution
## ... Procrustes: rmse 9.403928e-05  max resid 0.0002552021
## ... Similar to previous best
## Run 2 stress 9.683911e-05
## ... New best solution
## ... Procrustes: rmse 1.789734e-05  max resid 7.081417e-05
## ... Similar to previous best
## Run 3 stress 9.803809e-05
## ... Procrustes: rmse 9.664803e-05  max resid 0.0004008584
## ... Similar to previous best
## Run 4 stress 9.728535e-05
## ... Procrustes: rmse 9.348319e-05  max resid 0.0003806453
## ... Similar to previous best
## Run 5 stress 9.433012e-05
## ... New best solution
## ... Procrustes: rmse 8.009263e-05  max resid 0.0003387141
## ... Similar to previous best
```

```
## Run 6 stress 9.854994e-05
## ... Procrustes: rmse 5.649027e-05  max resid 0.0003282367
## ... Similar to previous best
## Run 7 stress 9.714434e-05
## ... Procrustes: rmse 8.46395e-05  max resid 0.0004552926
## ... Similar to previous best
## Run 8 stress 8.643515e-05
## ... New best solution
## ... Procrustes: rmse 9.740003e-05  max resid 0.00034845
## ... Similar to previous best
## Run 9 stress 9.482124e-05
## ... Procrustes: rmse 0.000120325  max resid 0.0004251261
## ... Similar to previous best
## Run 10 stress 9.986095e-05
## ... Procrustes: rmse 9.81361e-05  max resid 0.0003687143
## ... Similar to previous best
## Run 11 stress 9.781896e-05
## ... Procrustes: rmse 0.0001097843  max resid 0.000468463
## ... Similar to previous best
## Run 12 stress 9.833339e-05
## ... Procrustes: rmse 0.0001018289  max resid 0.0005207519
## ... Similar to previous best
## Run 13 stress 8.956919e-05
## ... Procrustes: rmse 9.645764e-05  max resid 0.0005041369
## ... Similar to previous best
## Run 14 stress 9.628905e-05
## ... Procrustes: rmse 0.0001105789  max resid 0.000435329
## ... Similar to previous best
## Run 15 stress 9.969761e-05
## ... Procrustes: rmse 0.0001165857  max resid 0.0004178461
## ... Similar to previous best
## Run 16 stress 9.789057e-05
## ... Procrustes: rmse 0.0001083193  max resid 0.0004947965
## ... Similar to previous best
## Run 17 stress 0.3994062
## Run 18 stress 9.562651e-05
## ... Procrustes: rmse 9.556637e-05  max resid 0.0003029397
## ... Similar to previous best
## Run 19 stress 9.547996e-05
## ... Procrustes: rmse 0.0001013713  max resid 0.0004815545
## ... Similar to previous best
## Run 20 stress 9.817895e-05
## ... Procrustes: rmse 0.0001082799  max resid 0.0005106343
## ... Similar to previous best
## *** Best solution repeated 12 times
```

```
## Warning in metaMDS(veganifyOTU(physeq), distance, ...): stress is (nearly)
## zero: you may have insufficient data
```

```
plot_ordination( ps,
                 ps.bcord,
                 color = "s.type",
                 # shape = "s.type"
                 ) +
  geom_point(size=3) +
```

```
  ggtitle("DECON5 - ITS raref.- brayc") +
  theme_classic()
```

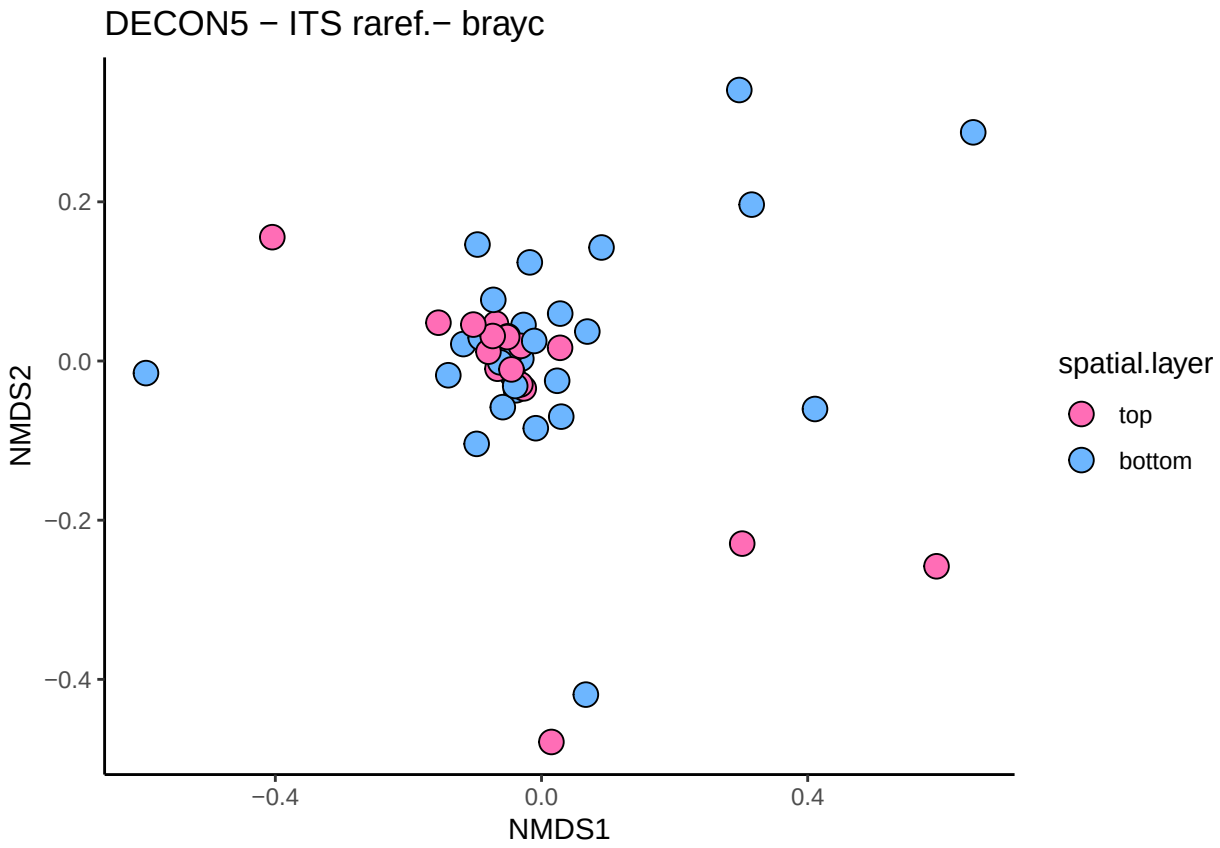



```
## fungus garden samples only
# Bray Curtis NMDS
ps.fg.bcord = ordinate(ps.fg, method= "NMDS", distance = "bray")
```

```
## Square root transformation
## Wisconsin double standardization
## Run 0 stress 0.1086804
## Run 1 stress 0.133344
## Run 2 stress 0.1354336
## Run 3 stress 0.1123719
## Run 4 stress 0.1798987
## Run 5 stress 0.1220326
## Run 6 stress 0.1270933
## Run 7 stress 0.1164596
## Run 8 stress 0.1503703
## Run 9 stress 0.106431
## ... New best solution
## ... Procrustes: rmse 0.02077204  max resid 0.1082731
## Run 10 stress 0.1406898
## Run 11 stress 0.1444305
## Run 12 stress 0.127347
## Run 13 stress 0.1273067
## Run 14 stress 0.1358101
## Run 15 stress 0.1487221
```

```
## Run 16 stress 0.1286499
## Run 17 stress 0.1814309
## Run 18 stress 0.1382041
## Run 19 stress 0.1426847
## Run 20 stress 0.1064308
## ... New best solution
## ... Procrustes: rmse 0.0002167309  max resid 0.00103193
## ... Similar to previous best
## *** 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),
    color = "black",
    shape = 21,
    size=4
  ) +
  scale_fill_manual(
    values = unlist(spatial.layer.col.ls)
  ) +
  ggtitle("DECON5 - ITS raref.- brayc") +
  theme_classic()
p$data$spatial.layer = factor(p$data$spatial.layer, levels=c("top","bottom"))
p
```




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

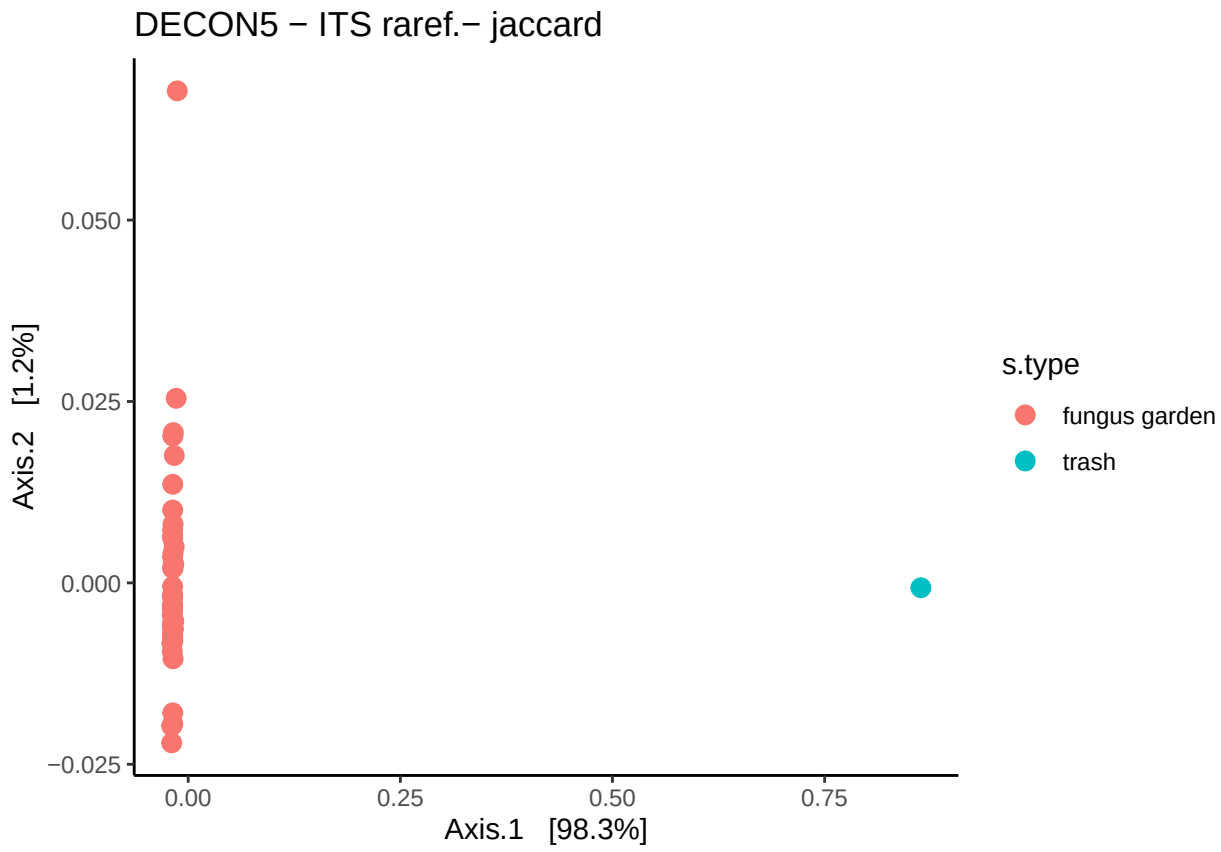
```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.bcdist ~ s.type, data = ps.samdat.df)
##          Df SumOfSqs     R2      F Pr(>F)
## Model     1  0.60856 0.9943 8201.7  0.022 *
## Residual 47  0.00349 0.0057
## Total    48  0.61204 1.0000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# 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     1 0.0000987 0.02831 1.3403   0.28
## Residual 46 0.0033886 0.97169
## Total    47 0.0034874 1.00000
```

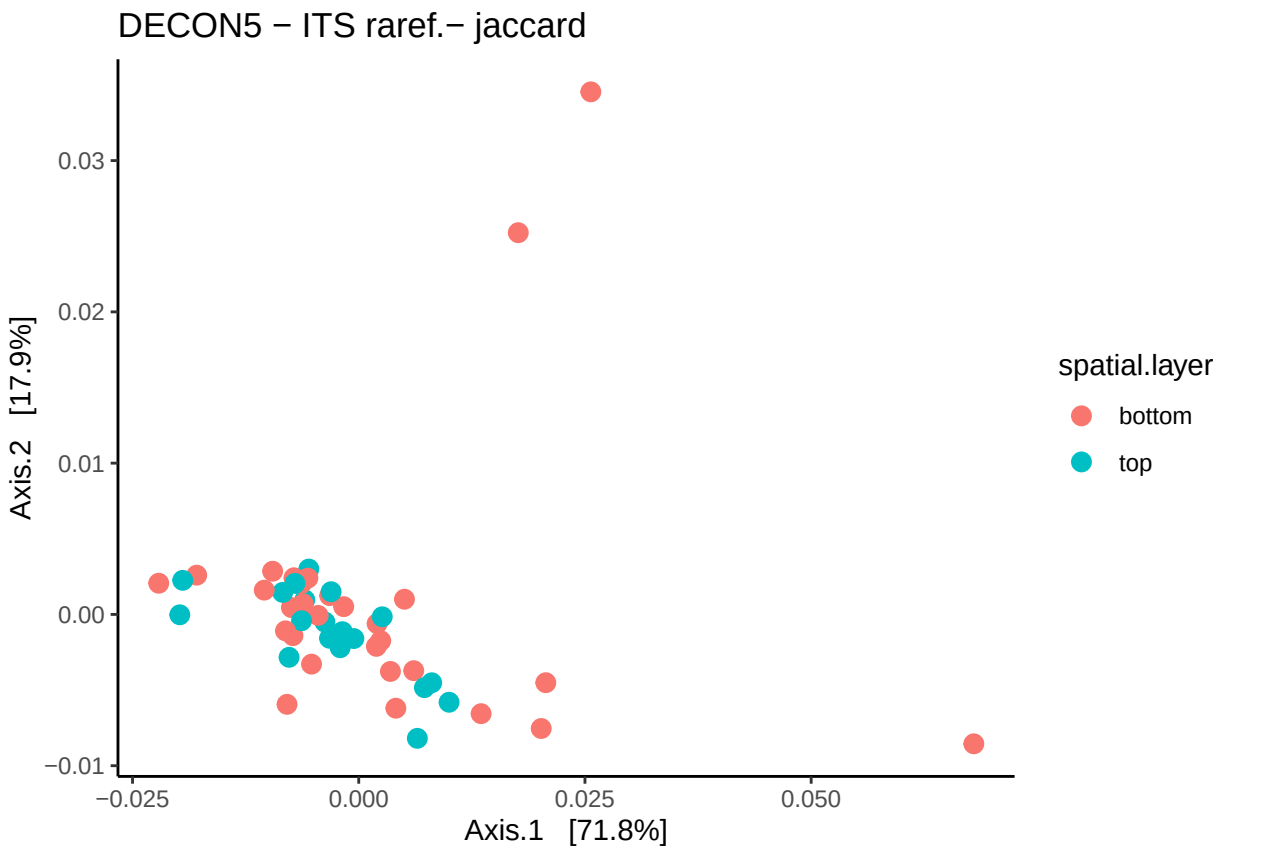
```
# Jaccard PCoA
ps.jccord = ordinate(ps, method= "PCoA", distance = "jaccard")

plot_ordination( ps,
                 ps.jccord,
                 color = "s.type",
                 # shape = "s.type"
                 ) +
  geom_point(size=3) +
  ggtitle("DECON5 - ITS raref.- jaccard") +
  theme_classic()
```




**d5JCC**

```
## fungus garden samples only
# Jaccard PCoA
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("DECON5 - ITS raref.- jaccard") +
```

```
  theme_classic()
```




```
# 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  0.76104 0.98283 2690.7  0.022 *
## Residual 47  0.01329 0.01717
## Total    48  0.77434 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# 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     1 0.0003726 0.02803 1.3266  0.257
## Residual 46 0.0129211 0.97197
## Total    47 0.0132937 1.00000
```