ch3_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
```

How many reads lost as mitochondria?

```
ps = readRDS("p1p2_16S_ps.p.decontam.prev0.5.RDS")

# subset_taxa() removes instances of NA if part of match
ps.nonc =
  prune_samples(!(sample_names(ps) %>% grepl(pattern = "NC|NFW|zymo")),
                ps)
ps.nonc = na_to_unclassified_taxa(ps.nonc)

# starting with 2165 taxa and 166 samples
ps.filt = subset_taxa(ps.nonc, Kingdom == "Bacteria")           # 2158 taxa
ps.filt = subset_taxa(ps.filt, Phylum != "unclass. Bacteria")   # 2150 taxa
ps.filt = subset_taxa(ps.filt, Order != "Chloroplast")          # 2105 taxa
ps.filt = subset_taxa(ps.filt, Family != "Mitochondria")        # 2068 taxa

subset_taxa(ps.nonc, Family == "Mitochondria") %>%
  taxa_sums() %>%
  sort(decreasing=T) %>%
  head(n=3)
```

```
##   ASV5   ASV9  ASV25
## 132470  49385  23215
```

```
#   ASV5   ASV9  ASV25
# 132470  49385  23215
subset_taxa(ps.nonc, Order == "Chloroplast") %>%
  taxa_sums() %>%
  sort(decreasing = T) %>%
  head(n=3)
```

```
##   ASV1   ASV6 ASV192
## 327994 130263    921
```

```
#   ASV1   ASV6 ASV192
# 327994 130263    921

refseq(ps.nonc)[c("ASV5","ASV9","ASV25","ASV1","ASV6","ASV192")]
```

```
## DNAStringSet object of length 6:
##     width seq                                               names
## [1]   250 TACGAGAGGGGTAAGCATTATTC...CGAAAGTATGGGGAGCAAAACGG ASV5
## [2]   251 GACGGGGGGGGCAAGTGTTCTTC...CGAAAGCATGGGGAGCGAACAGG ASV9
## [3]   251 GACGGGGGGGGCAAGTGTTCTTC...CGAAAGCATGGGGAGCGAACGGG ASV25
## [4]   253 TACAGAGGATGCAAGCGTTATCC...CGAAAGCTAGGGGAGCGAATGGG ASV1
## [5]   253 GACAGAGGATGCAAGCGTTATCC...CGAAAGCTAGGGGAGCAAATGGG ASV6
## [6]   253 GACAGAGGATGCAAGCGTTATCC...CGAAAGCTAGGGGAGCAAATGGG ASV192
```

```
Biostrings::writeXStringSet(refseq(ps.nonc)[c("ASV5","ASV9","ASV25","ASV1","ASV6","ASV192")],
                            "top3mito_top3chlo.fasta")
```

Load Data

Load phyloseq objects for decon3, decon4, decon5, and tsinf bacterial 16Sv4 data. These objects were created using DADA2 v??. Decontam... Fastree, 500 boot, midpoint root.... Unclassified Phyla, mitochondria, and

chloroplast removed.

```
psList = readRDS("~/Documents/ch3_analyses/bact/psList.final.RDS")
#
# # update decons sample data
# decons.samdat = read.csv("~/Documents/ch3_analyses/decons_master_sampledata.csv")
# rownames(decons.samdat) = decons.samdat$seq.id
#
# for( i in c("decon3","decon4","decon5") ) {
#   sample_data(psList[[i]]) = decons.samdat
# }
#
# saveRDS(psList, "psList.final.RDS")
```

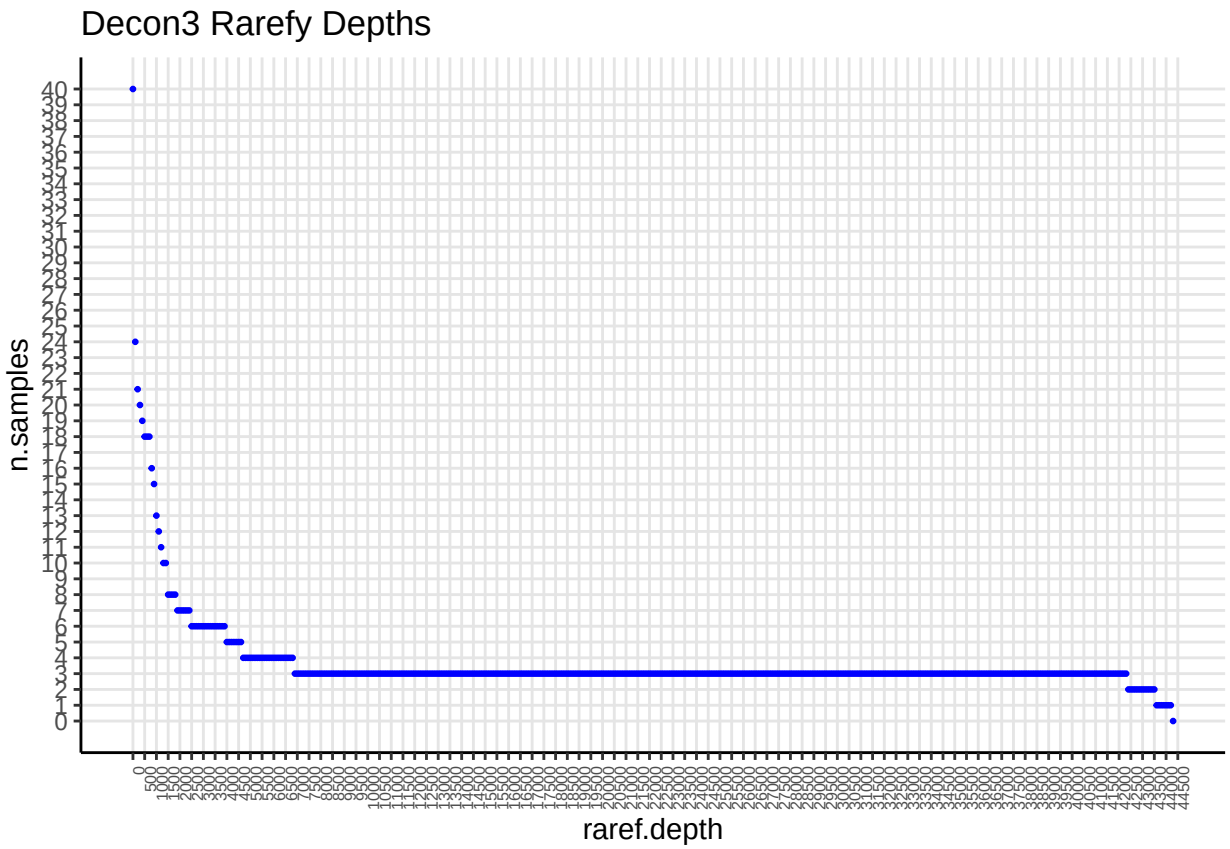
Rarefy Explore

The decon datasets unfortunately have pretty low reads :'(

Decon3

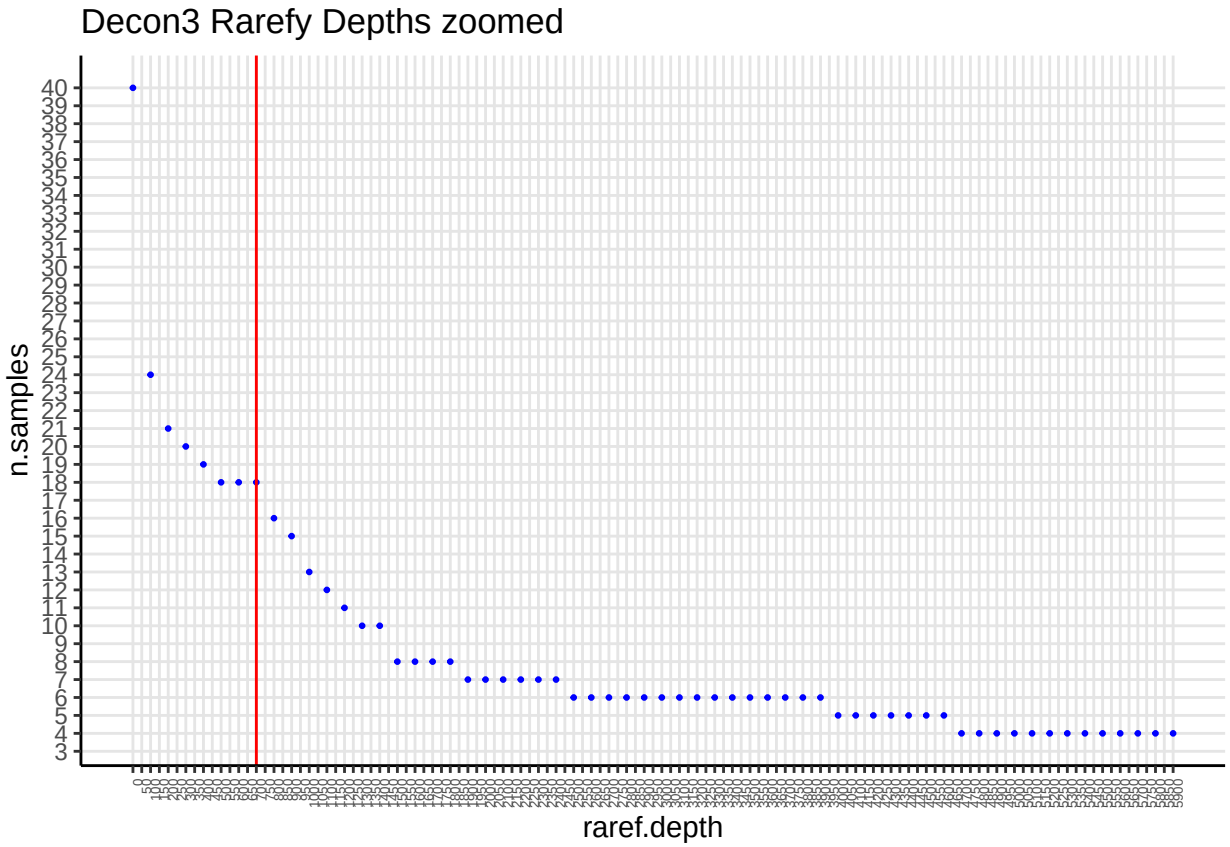
Looks like 700 reads might be an alright cutoff?

```
## plot raref threshold increments of 50 reads
ps = 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 = 700, color="red")
```



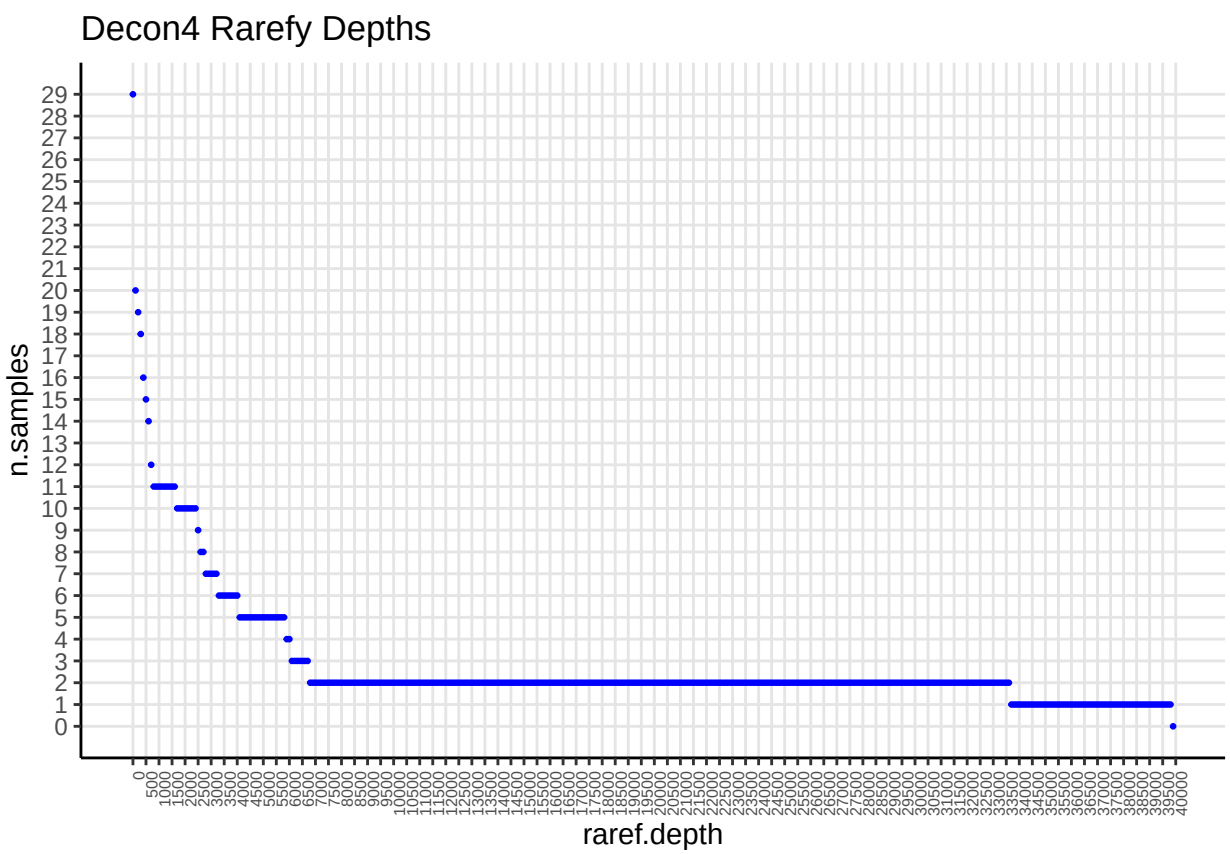

```
#ggsave("png/decon3_rarefyexplore.zoom_t700.png")
```

Decon4

500 reads keeps 50% of samples (n=15)... 700 reads removes 3 more samples (n=12).

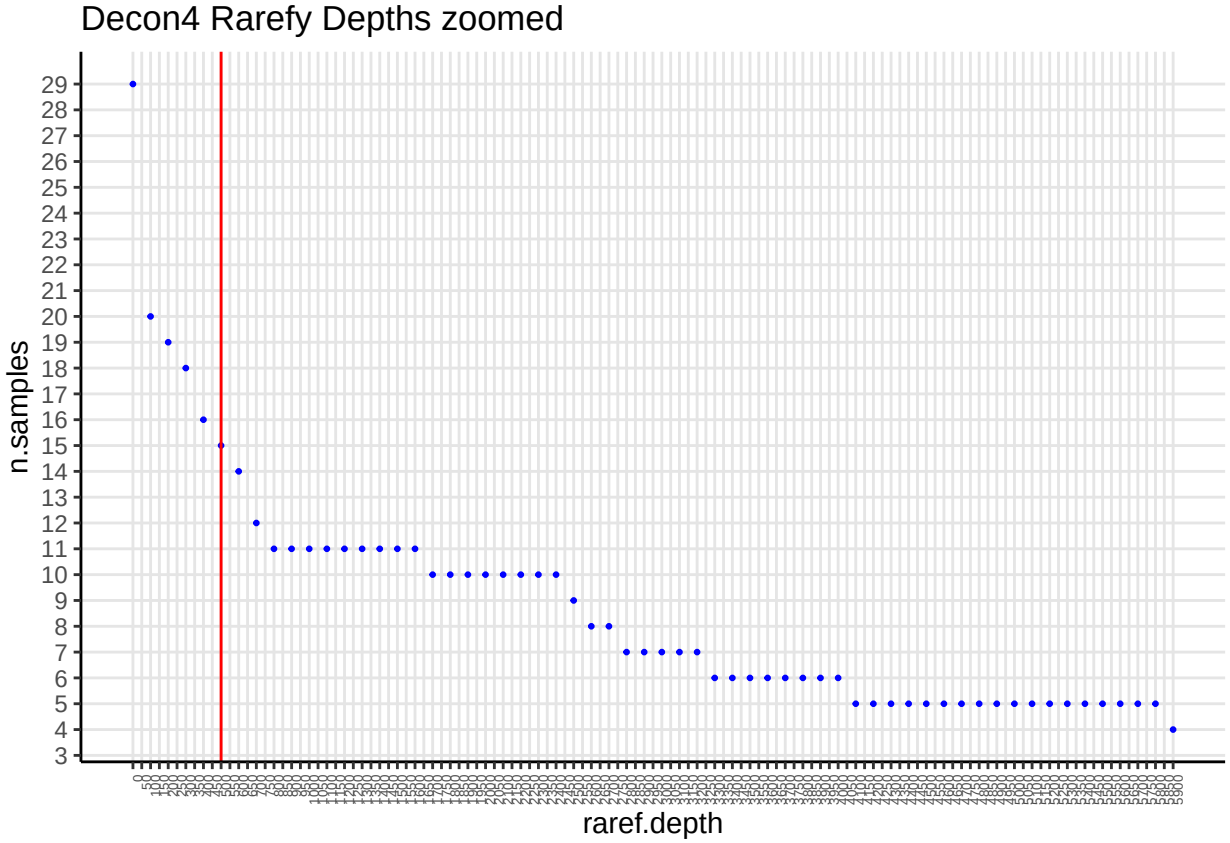
```
## plot raref threshold increments of 50 reads
ps = psList$decon4

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")
    )
```




```
#ggsave("png/decon4_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("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 = 500, color="red")
```

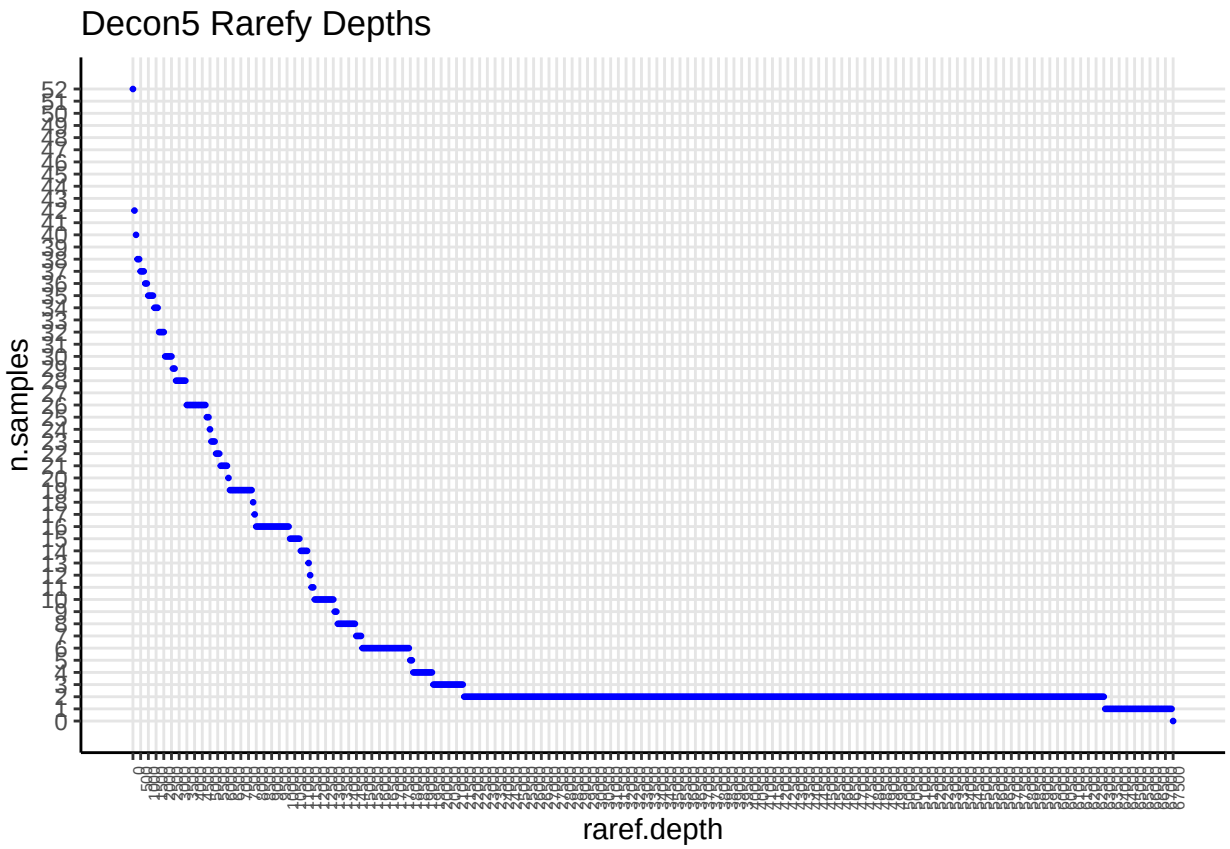



```
#ggsave("png/decon4_rarefyexplore.zoom_t500.png")
```

Decon5

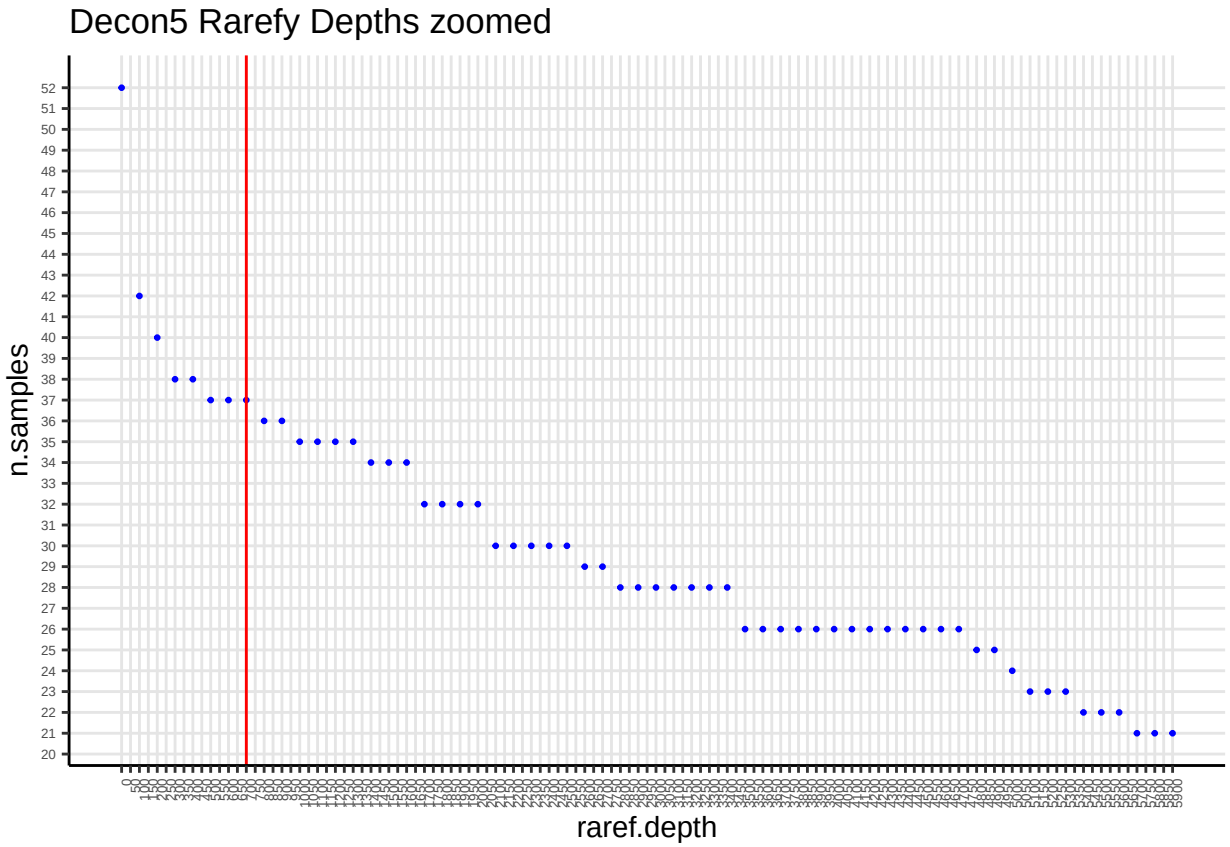
700 reads also seems good for DECON5, just like decon3.

```r
## plot raref threshold increments of 50 reads
ps = 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")
    )
```




```
#ggsave("png/decon5_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("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),
        axis.text.y = element_text(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 = 700, color="red")
```



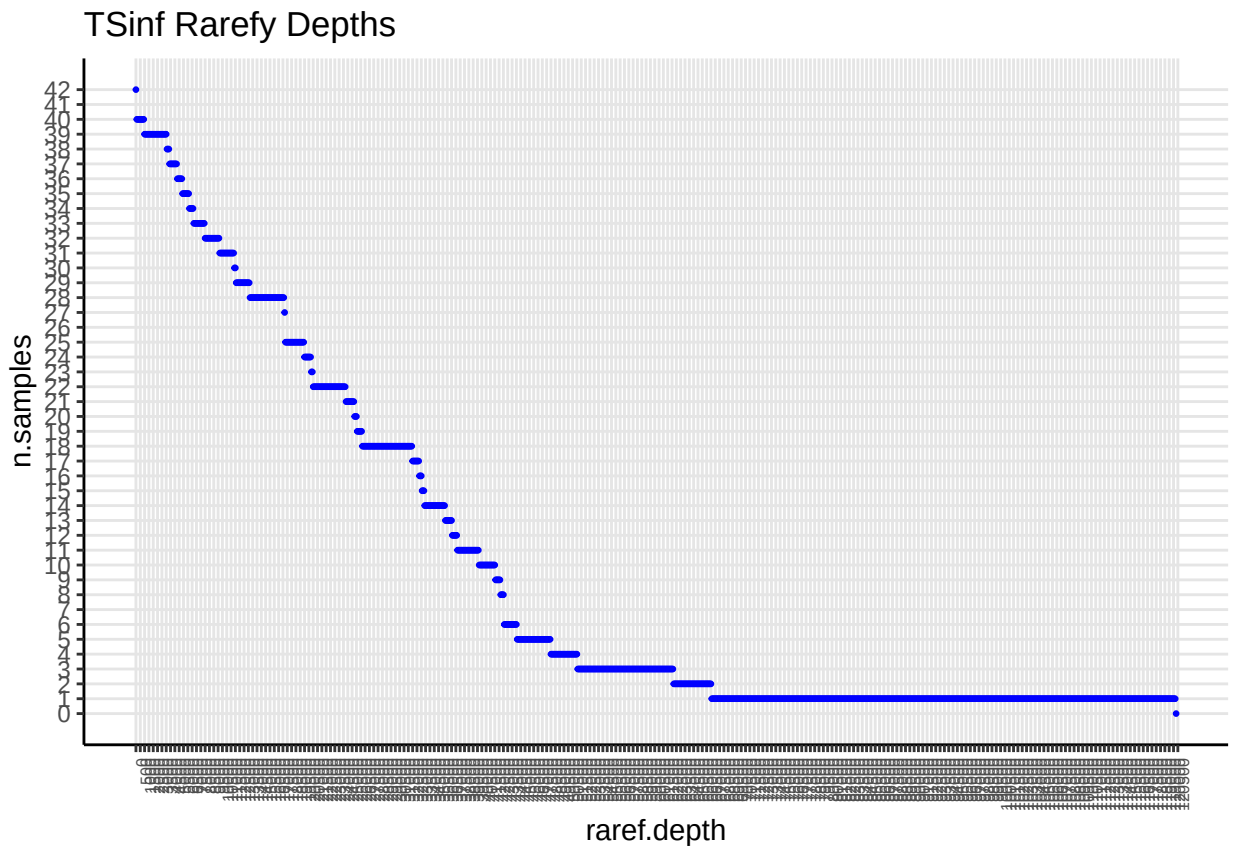

```
#ggsave("png/decon5_rarefyexplore.zoom_t700.png")
```

TSinf

This dataset has a lot more reads. 3500 reads seems like a good value.

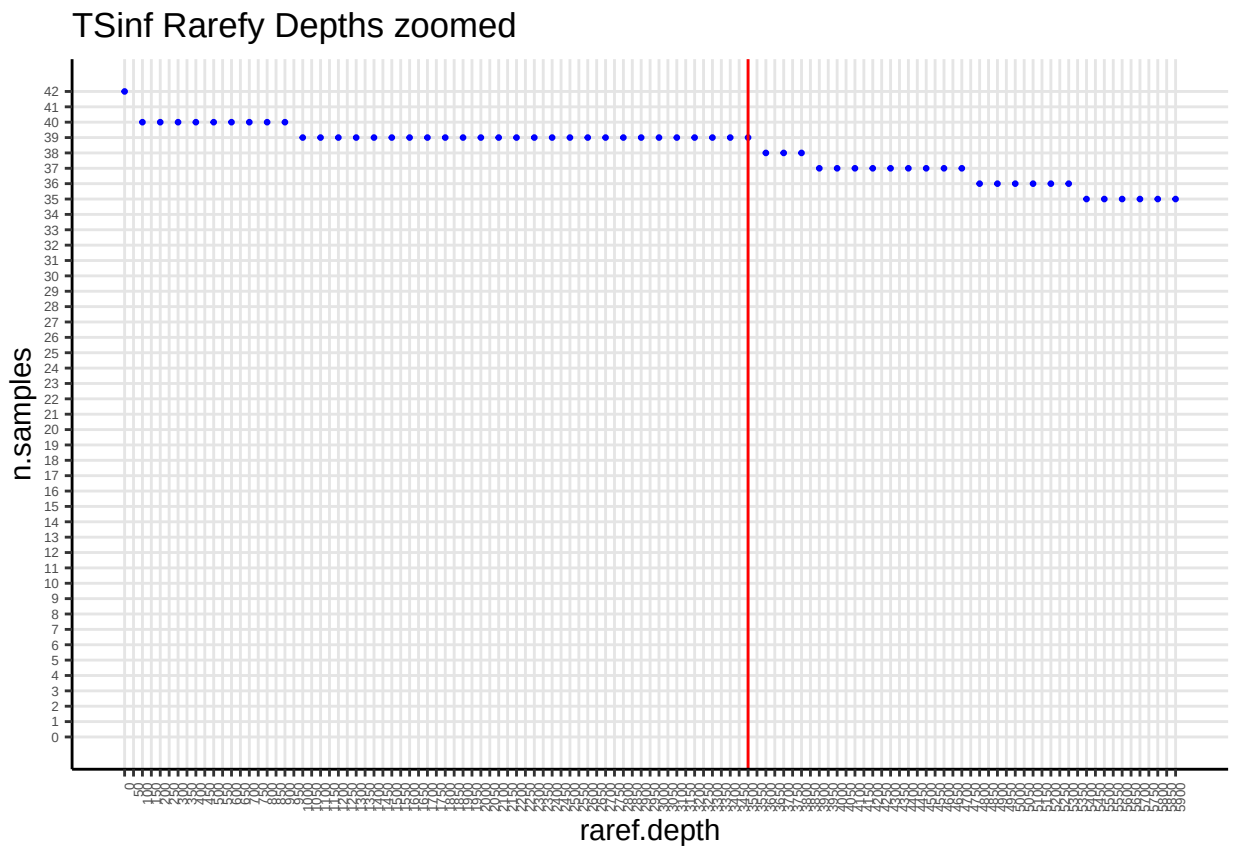
```
## plot raref threshold increments of 50 reads
ps = psList$tsinf

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("TSinf Rarefy Depths") +
    scale_x_continuous(
        breaks = seq(0, m+500, by=500)
    ) +
    scale_y_continuous(
      limits = c(0, nsamples(ps)),
        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/tsinf_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("TSinf Rarefy Depths zoomed") +
  scale_x_continuous(
    limits = c(0, x.max),
      breaks = seq(0, x.max, by=50)
  ) +
  scale_y_continuous(
    limits = c (0, y.max),
      breaks = seq(0, y.max, by=1)
  ) +
  theme_classic() +
  theme(
      axis.text.x = element_text(angle = 90, hjust=1, size=5),
      axis.text.y = element_text(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 = 3500, color="red")
```



```
  # ggsave("png/tsinf_rarefyexplore.zoom_t3500.png")
```

Rarefy

Rarefying decon3 to 700 reads, decon4 to 500, decon5 to 700, and tsinf to 3500.

```
# d3 -- raref to 182, 294, 329, 490, 702... 700
# d4 -- raref to 209, 335, 352, 412, 558 ...500
# d5 -- raref to 233, 268, 490, 701... 700
# ts -- raref to 909, 3510 ... 3500

psList.raref =
  list( decon3 = rarefy_even_depth(
      psList$decon3, sample.size=700, rngseed=1103, replace=F),
        decon4 = rarefy_even_depth(
          psList$decon4, sample.size=500, rngseed=1103, replace=F),
        decon5 = rarefy_even_depth(
          psList$decon5, sample.size=700, rngseed=1103, replace=F),
        tsinf = rarefy_even_depth(
          psList$tsinf, sample.size=3500, 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

## ...

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

## Up to first five removed samples are:

## d3e02rcd10retry1p2d3e04rcd10retry1p2d3e05rcd10retry1p2d3e06rcd10d3e07rcd10

## ...

## 1536OTUs 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

## ...

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

## Up to first five removed samples are:

## d4e01rcd10d4e02rcd10d4e05rcd10d4e07rcd10d4e08rcd10

## ...

## 1844OTUs 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
## ...
## 15 samples removedbecause they contained fewer reads than `sample.size`.
## Up to first five removed samples are:
## d5e10rcd5e17rcd5e22rcd5e23rcd5e24rc
## ...
## 16590TUs 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:
## TSe30rcd10retry1p2TSe32rcd10TSe42rcd10retry1p2
## ...
## 9450TUs were removed because they are no longer
## present in any sample after random subsampling
## ...
psList.raref.rab = make_rel_abund(psList.raref)
```

Stacked bar plots

Decon3

Rarefied to 700 reads. Phyla <15% abundant in all samples are labelled "other".

```
ps = psList.raref$decon3
sample_sums(ps)
```

```
##         d3e01rcd10         d3e03rcd10         d3e08rcd10         d3e10rcd10
##                700                700                700                700
##         d3e14rcd10         d3e19rcd10         d3e21rcd10         d3e22rcd10
##                700                700                700                700
##         d3e26rcd10 d3e27rcd10retry1p2         d3e29rcd10         d3e30rcd10
##                700                700                700                700
##         d3e32rcd10         d3e33rcd10         d3e39rcd10 d3e41rcd10retry1p2
##                700                700                700                700
## d3e45rcd10retry1p2 d3e46rcd10retry1p2
##                700                700
```

```
## replace NA in spatial.layer to "none"
ps <- ps %>%
    ps_mutate(
        spatial.layer = replace(spatial.layer,is.na(spatial.layer),"none")
    )

## glom
ps.glom = tax_glom(ps, "Class")
## tax.glom <15% all samples filter
make_rel_abund(ps.glom) %>% taxnames_lt_threshold(0.15)
```

```
##  [1] "ASV279"  "ASV1690" "ASV14"   "ASV700"  "ASV480"  "ASV310"  "ASV781"
##  [8] "ASV1953" "ASV213"  "ASV2117" "ASV363"  "ASV989"  "ASV1952" "ASV889"
## [15] "ASV493"  "ASV339"  "ASV94"   "ASV19"   "ASV853"  "ASV59"   "ASV126"
## [22] "ASV83"   "ASV257"  "ASV904"  "ASV18"   "ASV293"  "ASV925"
```

What are my sample variables?

```
sample_variables(ps.glom)
```

```
##  [1] "seq.id"            "s.id"              "s.name"
##  [4] "s.type"            "fg.coord"          "spatial.layer"
##  [7] "spatial.col"       "spatial.row"       "trashed.yn"
## [10] "inoc.dist"         "inoc.dist.cm"      "extr.batch"
## [13] "pcr.plate"         "pcr.batch"         "dil.factor"
## [16] "Sample_or_Control" "host.species"
```

Class Colors

```
d3.class0.01 = as.vector(
  tax_table(per_sample_taxafilt(psList.raref$decon3,threshold=0.01, glom="Class"))[,"Class"])
```

```
##
## >>> Glomming by Class ... ... ...
```

```
d4.class0.01 = as.vector(
  tax_table(per_sample_taxafilt(psList.raref$decon4,threshold=0.01, glom="Class"))[,"Class"])
```

```
##
## >>> Glomming by Class ... ... ...
```

```
d5.class0.01 = as.vector(
  tax_table(per_sample_taxafilt(psList.raref$decon5,threshold=0.01, glom="Class"))[,"Class"])
```

```
##
## >>> Glomming by Class ... ... ...
```

```
ts.class0.01 = as.vector(
  tax_table(per_sample_taxafilt(psList.raref$tsinf,threshold=0.01, glom="Class"))[,"Class"])
```

```
##
## >>> Glomming by Class ... ... ...
```

```
class.cols.martin =
  setNames(unname(paletteMartin), # paletteMartin is 15 colors per chance
          c(d3.class0.01, d4.class0.01, d5.class0.01, ts.class0.01) %>% unique() )

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"
```

```
# Black    SherpaBlue  PersianGreen       HotPink   CottonCandy PigmentIndigo   ScienceBlue    Heliotro
#     "#000000"     "#004949"     "#009292"     "#ff6db6"     "#ffb6db"     "#490092"     "#006ddb"
#        Malibu    FrenchPass      RedBerry         Brown    MangoTango     Harlequin    LaserLemon
#     "#6db6ff"     "#b6dbff"     "#920000"     "#924900"     "#db6d00"     "#24ff24"     "#ffff6d"
ps.m = make_rel_abund(ps) %>%
  per_sample_taxafilt(threshold=0.01, glom="Class") %>%
  ps_melt()
```

```
##
## >>> Glomming by Class ... ... ...
```

```
ps.m %>% group_by(Class) %>% summarise(mean.rab = mean(Abundance)*100) %>% arrange(desc(mean.rab))
```

```
## # A tibble: 12 x 2
##    Class               mean.rab
##    <chr>                  <dbl>
##  1 Alphaproteobacteria   32.3
##  2 Bacilli               18.0
##  3 other 0.01            15.2
##  4 Actinobacteria        12.7
##  5 Bacteroidia            8.27
##  6 Gammaproteobacteria    5.17
##  7 Chlamydiia             3.28
##  8 Verrucomicrobiia       3.20
##  9 Leptospirae            0.690
## 10 Myxococcia             0.540
## 11 Planctomycetes         0.516
## 12 Gemmatimonadia         0.230
```

```
#save.image()
```

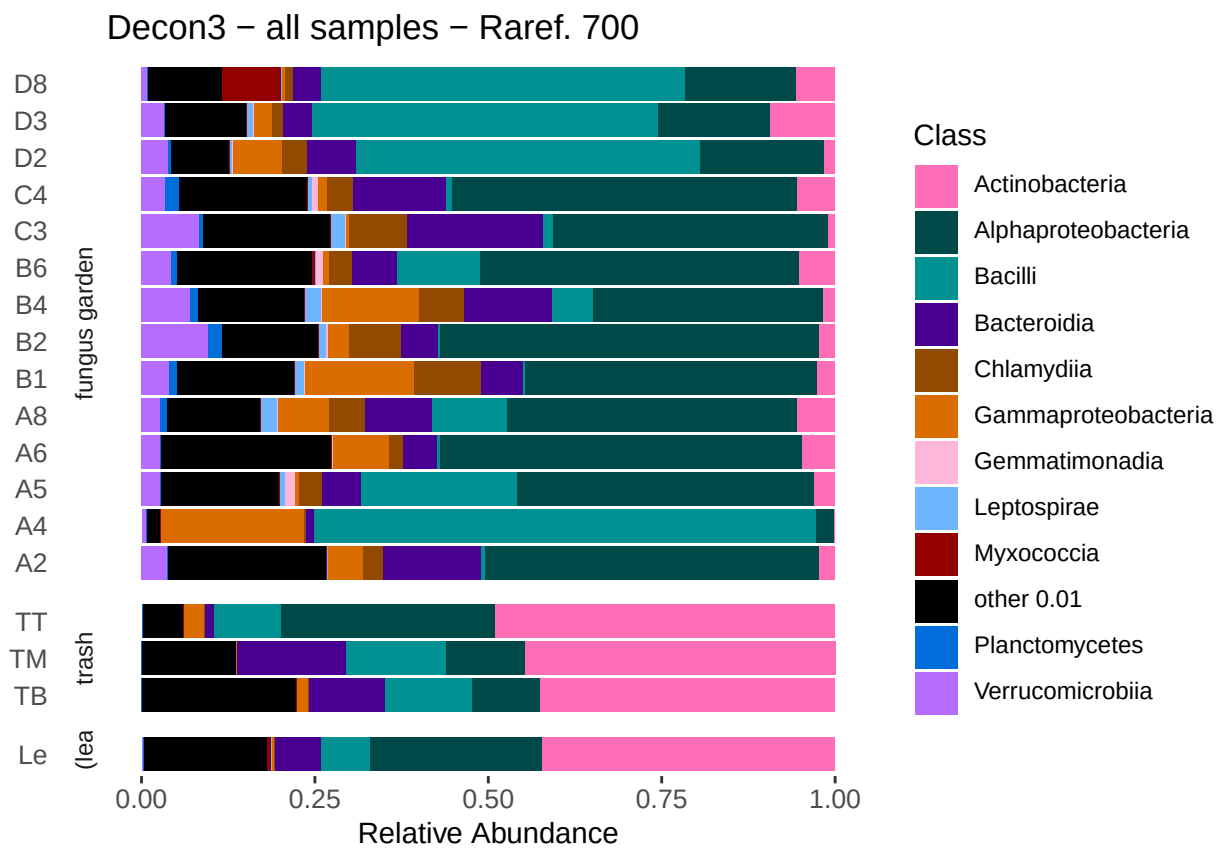
Plot!

```
## ---- stacked bar plot ---- ##
p <-
    make_rel_abund(ps) %>%
    per_sample_taxafilt(threshold=0.01, glom="Class") %>%
    # ps_filter(s.type == "fungus garden", .keep_all_taxa=T) %>%
    psmelt() %>%
    ggplot(
        aes(x=Abundance, y=s.name, fill=Class)
    ) +
    geom_bar(
      stat="identity"
    ) +
    theme(
      axis.text=element_text(size=10)
    ) +
    scale_fill_manual(
      # values = c(unname(paletteMartin[2:5]),"black")
```

```
      values = class.cols.martin
    ) +
    facet_grid(
      # rows=vars(s.type,spatial.layer),
      rows = vars(s.type),
      # scales="free_y",
      scales = "free",
      space = "free",
      switch="y"
    ) +
    labs(x="Relative Abundance",y=NULL) +
  ggtitle(
    "Decon3 - all samples - Raref. 700"
  ) +
    theme(
      panel.background = element_blank(),
      # axis.text.y = element_blank(),
      legend.text = element_markdown(),
      panel.grid  = element_blank(),
      axis.ticks.y = element_blank(),
      strip.background = element_blank(),
      strip.switch.pad.grid = unit('00.1',"cm")
  )
```

```
##
## >>> Glomming by Class ... ... ...
```

```
# manually reorder
# p$data$Phylum =
#   factor(
#     p$data$Phylum,
#     levels = c( "Pseudomonadota","Bacillota","Bacteroidota","Actinomycetota","other 0.15" )
#   )
# p$data$spatial.layer =
#   factor(
#     p$data$spatial.layer,
#     levels = c("top","mid1","mid2","bottom")
#   )
p$data$s.type =
  factor(
    p$data$s.type,
    levels = c("fungus garden", "trash", "food (leaves)")
  )
# print plot
p
```

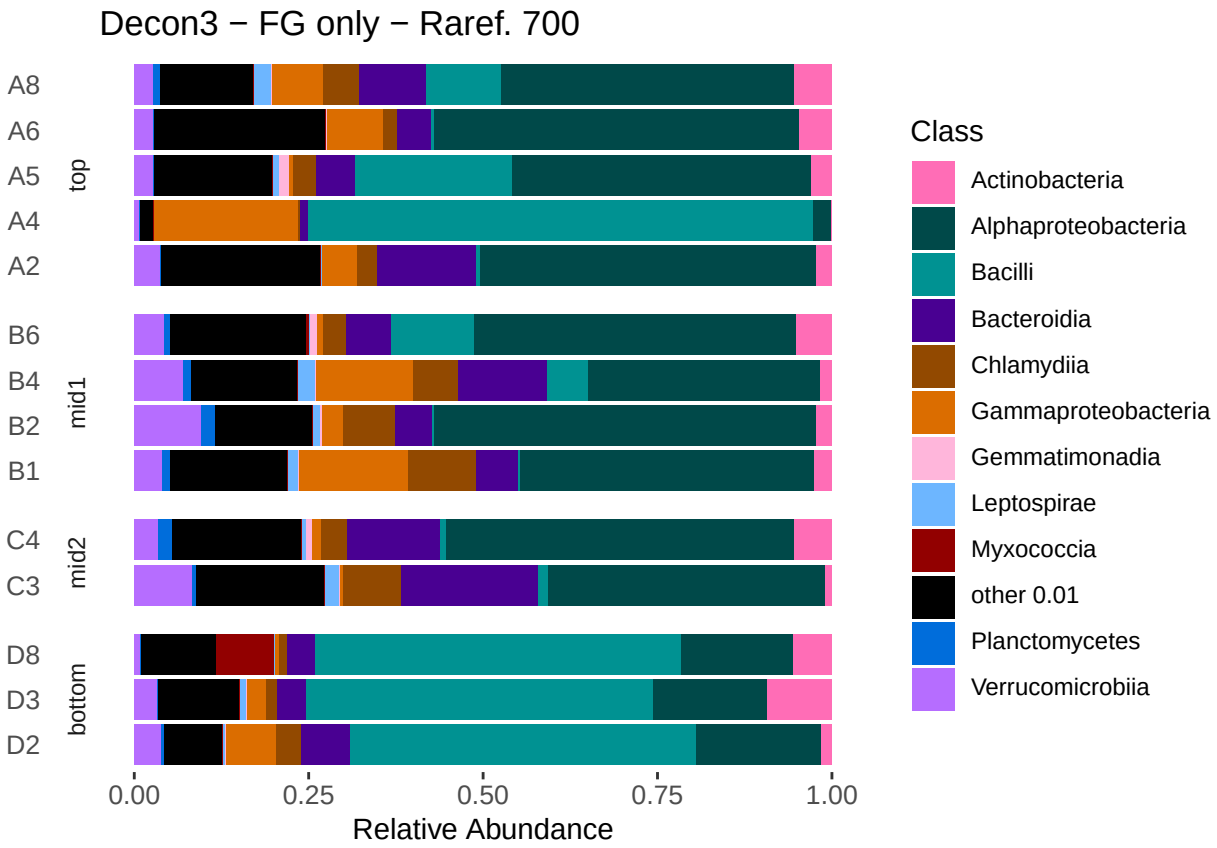



```
## ---- stacked bar plot ---- ##
p <-
    make_rel_abund(ps) %>%
    per_sample_taxafilt(threshold=0.01, glom="Class") %>%
    ps_filter(s.type == "fungus garden", .keep_all_taxa=T) %>%
    psmelt() %>%
    ggplot(
        aes(x=Abundance, y=s.name, fill=Class)
    ) +
    geom_bar(
      stat="identity"
    ) +
    theme(
      axis.text=element_text(size=10)
    ) +
    scale_fill_manual(
      # values = c(unname(paletteMartin[2:5]),"black")
      values = class.cols.martin
    ) +
    facet_grid(
      # rows=vars(s.type,spatial.layer),
      rows=vars(spatial.layer),
      # scales="free_y",
      scales = "free",
      space = "free",
      switch="y"
    ) +
```

```
    labs(x="Relative Abundance",y=NULL) +
  ggtitle("Decon3 - FG only - Raref. 700"
  ) +
    theme(
      panel.background = element_blank(),
      # axis.text.y = element_blank(),
      legend.text = element_markdown(),
      panel.grid  = element_blank(),
      axis.ticks.y = element_blank(),
      strip.background = element_blank(),
      strip.switch.pad.grid = unit('00.1',"cm")
  )
```

```
##
## >>> Glomming by Class ... ... ...
```

```
# manually reorder
# p$data$Phylum =
#   factor(
#     p$data$Phylum,
#     levels = c( "Pseudomonadota","Bacillota","Bacteroidota","Actinomycetota","other 0.15" )
#   )
p$data$spatial.layer =
  factor(
    p$data$spatial.layer,
    levels = c("top","mid1","mid2","bottom")
  )
# print plot
p
```



Decon4

Rarefied to 500 reads. Phyla <15% abundant in all samples labelled "other".

```
ps = psList.raref$decon4
# tax_table(ps)[,"Class"] %>% unique()
## glom by Phylum
ps.glom = tax_glom(ps, "Class")
## Phyla <15% all samples filter
make_rel_abund(ps.glom) %>% taxnames_lt_threshold(0.01)
```

```
##  [1] "ASV441" "ASV498" "ASV493" "ASV288" "ASV59"  "ASV126" "ASV94"  "ASV386"
##  [9] "ASV529" "ASV749"
```

What are my sample variables?

```
sample_variables(ps.glom)
```

```
##  [1] "seq.id"           "s.id"             "s.name"
##  [4] "s.type"           "fg.coord"         "spatial.layer"
##  [7] "spatial.col"      "spatial.row"      "trashed.yn"
## [10] "inoc.dist"        "inoc.dist.cm"     "extr.batch"
## [13] "pcr.plate"        "pcr.batch"        "dil.factor"
## [16] "Sample_or_Control" "host.species"
```

Mean rel abunds

```
ps.m = make_rel_abund(ps) %>%
  per_sample_taxafilt(threshold=0.01, glom="Class") %>%
  ps_melt()
```

```
##
## >>> Glomming by Class ... ... ...
```

```
ps.m %>% group_by(Class) %>% summarise(mean.rab = mean(Abundance)*100) %>% arrange(desc(mean.rab))
```

```
## # A tibble: 7 x 2
##   Class               mean.rab
##   <chr>                  <dbl>
## 1 Gammaproteobacteria   70.9
## 2 Actinobacteria         7.71
## 3 other 0.01             7.63
## 4 Clostridia             5.37
## 5 Bacilli                4.21
## 6 Alphaproteobacteria    3.77
## 7 Verrucomicrobiia       0.427
```

```
#save.image()
```

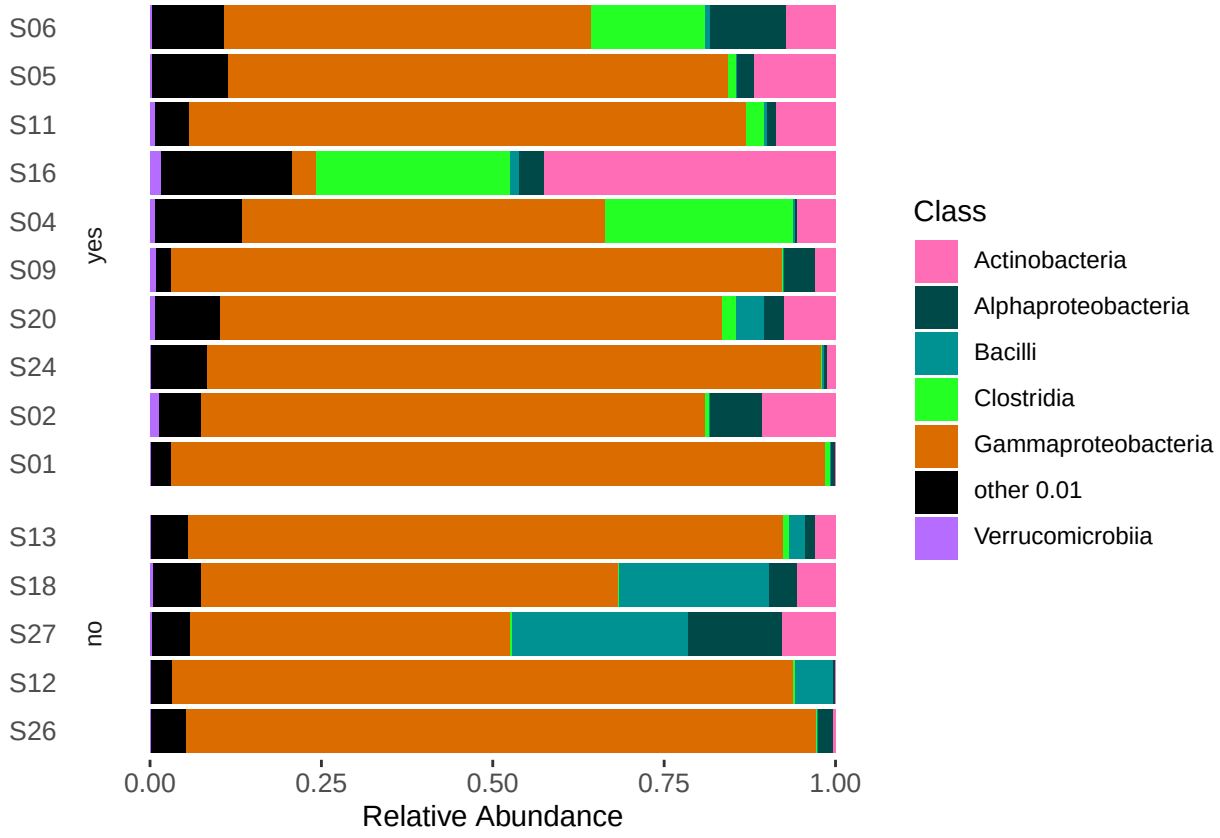
Plot!

```
## ---- stacked bar plot ---- ##
p <-
    make_rel_abund(ps) %>%
    per_sample_taxafilt(threshold=0.01, glom="Class") %>%
    # ps_filter(s.type == "fungus garden", .keep_all_taxa=T) %>%
    psmelt() %>%
  # mutate(s.name = fct_reorder(s.name, inoc.dist.cm)) %>%
    ggplot(
        aes(x=Abundance, y=reorder(s.name, -inoc.dist.cm), fill=Class)
    ) +
    geom_bar(
      stat="identity"
    ) +
    theme(
      axis.text=element_text(size=10)
    ) +
    scale_fill_manual(
      values = class.cols.martin
    ) +
    facet_grid(
      rows=vars(trashed.yn),
      scales="free",
      space = "free",
      switch="y"
    ) +
    labs(
      x="Relative Abundance",
      y=NULL
    ) +
    theme(
      panel.background = element_blank(),
      # axis.text.y = element_blank(),
      legend.text = element_markdown(),
      panel.grid  = element_blank(),
      axis.ticks.y = element_blank(),
      strip.background = element_blank(),
```

```
      strip.switch.pad.grid = unit('00.1',"cm")
  )
```

```
##
## >>> Glomming by Class ... ... ...
```

```
# manually reorder
# p$data$Phylum =
#   factor(
#     p$data$Phylum,
#     levels = c("Pseudomonadota","Bacillota","Actinomycetota","other 0.15" )
#   )
# p$data$Class =
#   factor(
#     p$data$Class,
#     levels = c("Actinobacteria", "Alphaproteobacteria", "Bacilli",
#                "Clostridia", "Gammaproteobacteria", "Verrucomicrobiia", "other 0.01"
#                )
#   )
p$data$trashed.yn =
  factor(
    p$data$trashed.yn,
    levels = c("yes","no")
  )

# print plot
p
```

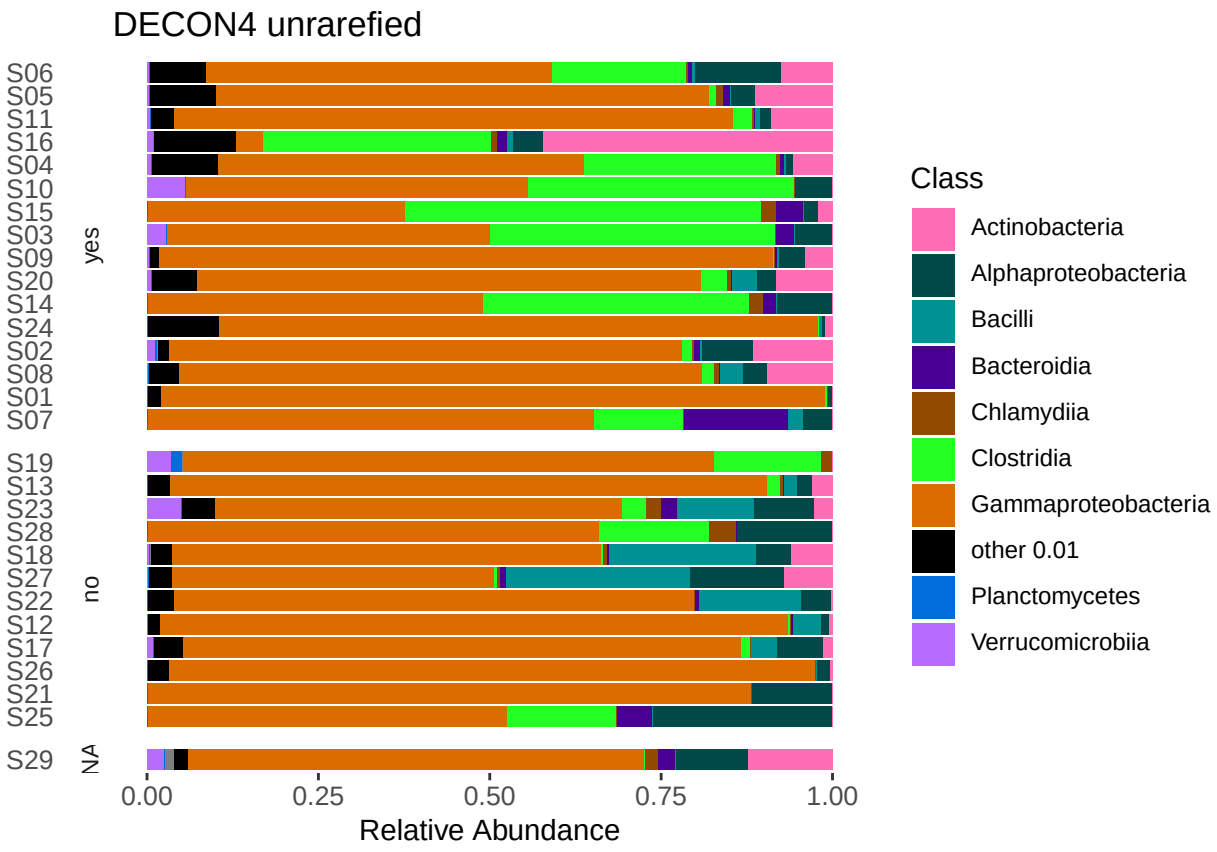


```
ps = psList$decon4

## ---- stacked bar plot ---- ##
p <-
    make_rel_abund(ps) %>%
    per_sample_taxafilt(threshold=0.01, glom="Class") %>%
    # ps_filter(s.type == "fungus garden", .keep_all_taxa=T) %>%
    psmelt() %>%
    ggplot(
        aes(x=Abundance, y=reorder(s.name, -inoc.dist.cm), fill=Class)
    ) +
    geom_bar(
      stat="identity"
    ) +
    theme(
      axis.text=element_text(size=10)
    ) +
    scale_fill_manual(
      # values = c(unname(paletteMartin[2:5]),"black")
      values = class.cols.martin
    ) +
    facet_grid(
      rows=vars(trashed.yn),
      scales="free_y",
      space = "free",
      switch="y"
    ) +
    labs(
      x="Relative Abundance",
      y=NULL
    ) +
  ggtitle(
    "DECON4 unrarefied"
  ) +
    theme(
      panel.background = element_blank(),
      # axis.text.y = element_blank(),
      legend.text = element_markdown(),
      panel.grid  = element_blank(),
      axis.ticks.y = element_blank(),
      strip.background = element_blank(),
      strip.switch.pad.grid = unit('00.1',"cm")
  )
```

```
##
## >>> Glomming by Class ... ... ...
```

```
# manually reorder
# p$data$Phylum =
#   factor(
#     p$data$Phylum,
#     levels = c("Pseudomonadota","Bacillota","Bacteroidota","Actinomycetota","other 0.15" )
#   )
p$data$trashed.yn =
```

```
  factor(
    p$data$trashed.yn,
    levels = c("yes","no")
  )
# print plot
p
```



Decon5

Rarefied to 700 reads. Phyla <15% abundant in all samples labelled as "other".

```
ps = psList.raref$decon5
## glom by Phylum
ps.glom = tax_glom(ps, "Class")
## Phyla <15% all samples filter
make_rel_abund(ps.glom) %>% taxnames_lt_threshold(0.15)
```

```
##  [1] "ASV59"   "ASV126"  "ASV938"  "ASV288"  "ASV1331" "ASV339"  "ASV94"
##  [8] "ASV1112" "ASV310"  "ASV1845" "ASV1460" "ASV1962" "ASV2044" "ASV213"
## [15] "ASV308"  "ASV124"  "ASV440"  "ASV493"  "ASV39"   "ASV878"  "ASV257"
## [22] "ASV749"  "ASV777"  "ASV539"  "ASV293"  "ASV636"  "ASV279"
```

```
sample_sums(ps)
```

```
##      d5e01rc      d5e02rc      d5e03rc      d5e04rc      d5e05rc      d5e06rc
##          700          700          700          700          700          700
##      d5e07rc      d5e08rc      d5e09rc      d5e11rc      d5e12rc      d5e13rc
##          700          700          700          700          700          700
```

```
##      d5e14rc      d5e15rc      d5e16rc      d5e18rc d5e19rcretry      d5e20rc
##          700          700          700          700          700          700
##      d5e21rc      d5e27rc      d5e28rc      d5e29rc      d5e31rc      d5e32rc
##          700          700          700          700          700          700
##      d5e36rc      d5e37rc      d5e38rc      d5e39rc      d5e41rc      d5e42rc
##          700          700          700          700          700          700
##      d5e44rc      d5e45rc      d5e46rc      d5e48rc      d5e49rc      d5e50rc
##          700          700          700          700          700          700
##      d5e51rc
##          700
```

What are my sample variables?

```
sample_variables(ps.glom)
```

```
##  [1] "seq.id"            "s.id"              "s.name"
##  [4] "s.type"            "fg.coord"          "spatial.layer"
##  [7] "spatial.col"       "spatial.row"       "trashed.yn"
## [10] "inoc.dist"         "inoc.dist.cm"      "extr.batch"
## [13] "pcr.plate"         "pcr.batch"         "dil.factor"
## [16] "Sample_or_Control" "host.species"
```

Mean relabund per class

```
ps.m = make_rel_abund(ps) %>%
  per_sample_taxafilt(threshold=0.01, glom="Class") %>%
  ps_melt()
```

```
##
## >>> Glomming by Class ... ... ...
```

```
ps.m %>% group_by(Class) %>% summarise(mean.rab = mean(Abundance)*100) %>% arrange(desc(mean.rab))
```

```
## # A tibble: 11 x 2
##    Class               mean.rab
##    <chr>                  <dbl>
##  1 Alphaproteobacteria   40.4
##  2 Bacteroidia           16.7
##  3 other 0.01            12.0
##  4 Gammaproteobacteria    9.09
##  5 Chlamydiia             7.97
##  6 Verrucomicrobiia       7.46
##  7 Actinobacteria         2.72
##  8 Leptospirae            1.35
##  9 Thermoleophilia        1.10
## 10 Planctomycetes         0.869
## 11 Babeliae               0.390
```

```
#save.image()1
```

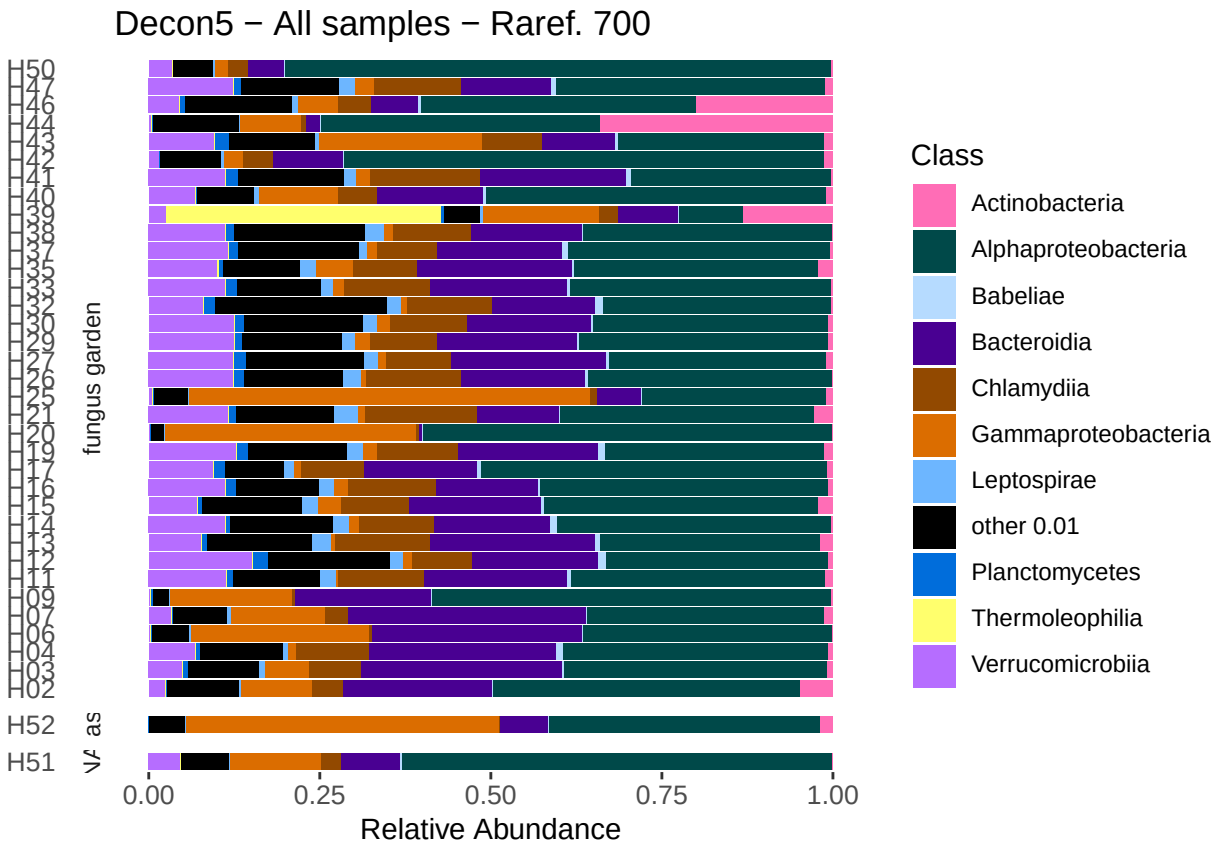
Plot!

```
## ---- stacked bar plot ---- ##
p <-
    make_rel_abund(ps) %>%
    per_sample_taxafilt(threshold=0.01, glom="Class") %>%
    # ps_filter(s.type == "fungus garden", .keep_all_taxa=T) %>%
    psmelt() %>%
```

```
    ggplot(
        aes(x=Abundance, y=s.name, fill=Class)
    ) +
    geom_bar(stat="identity"
    ) +
    theme(axis.text=element_text(size=10)
    ) +
    scale_fill_manual(
      # values = c(unname(paletteMartin[c(2,4:7)]),"black")
      values = class.cols.martin
    ) +
    facet_grid(
      rows=vars(s.type),
      scales="free",
      space = "free",
      switch="y"
    ) +
    labs(x="Relative Abundance",y=NULL
    ) +
  ggtitle(
    "Decon5 - All samples - Raref. 700"
  )+
    theme(
      panel.background = element_blank(),
      # axis.text.y = element_blank(),
      legend.text = element_markdown(),
      panel.grid  = element_blank(),
      axis.ticks.y = element_blank(),
      strip.background = element_blank(),
      strip.switch.pad.grid = unit('00.1',"cm")
  )
```

```
##
## >>> Glomming by Class ... ... ...
```

```
# manually reorder
# p$data$Phylum =
#   factor(
#     p$data$Phylum,
#     levels = c("Pseudomonadota","Bacteroidota",
#                "Actinomycetota","Chlamydiota","Verrucomicrobiota","other 0.15" )
#     )
p$data$spatial.layer =
  factor(
    p$data$spatial.layer, levels = c("top","bottom")
  )
p$data$s.type =
  factor(
    p$data$s.type,
    levels = c("fungus garden", "trash", "food (cornmeal)")
  )
# print plot
p
```

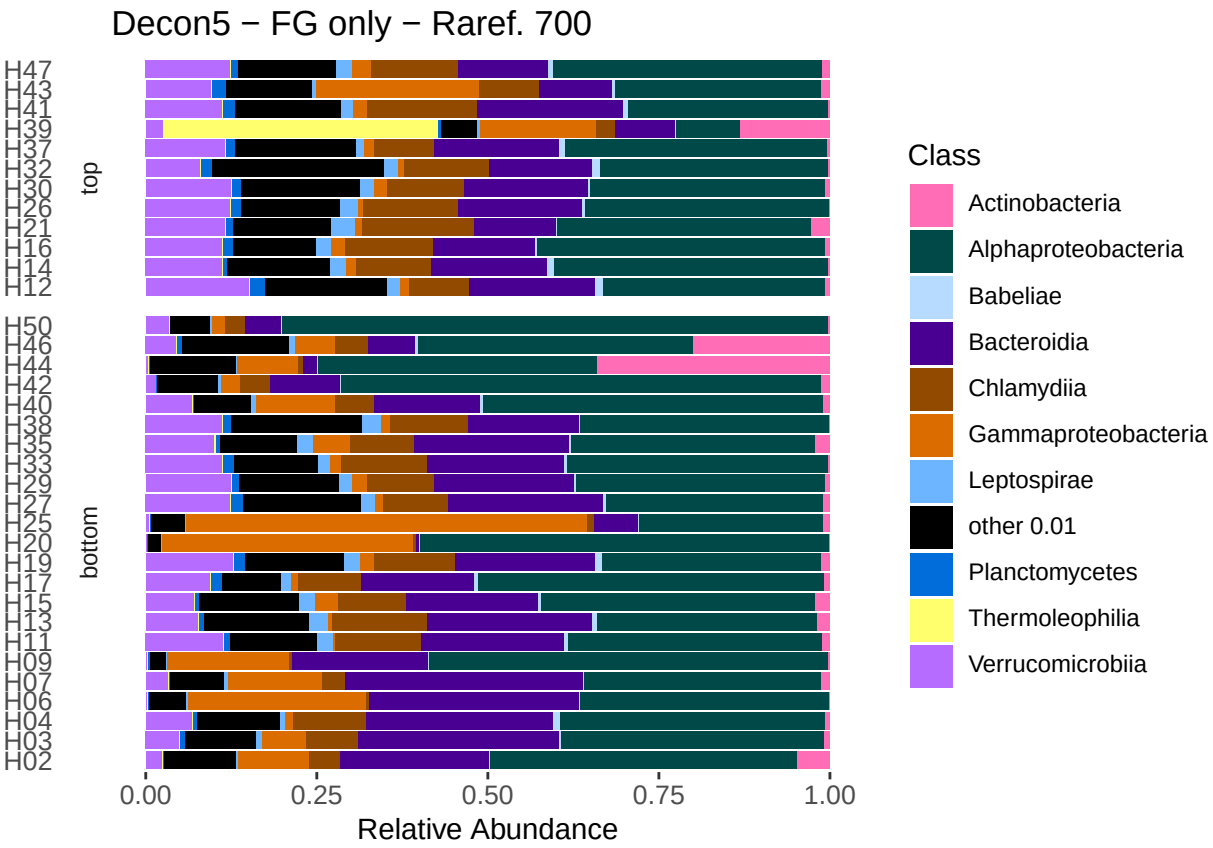



```
## ---- stacked bar plot ---- ##
p <-
    make_rel_abund(ps) %>%
    per_sample_taxafilt(threshold=0.01, glom="Class") %>%
    ps_filter(s.type == "fungus garden", .keep_all_taxa=T) %>%
    psmelt() %>%
    ggplot(
        aes(x=Abundance, y=s.name, fill=Class)
    ) +
    geom_bar(stat="identity"
    ) +
    theme(axis.text=element_text(size=10)
    ) +
    scale_fill_manual(
      # values = c(unname(paletteMartin[c(2,4:7)]),"black")
      values = class.cols.martin
    ) +
    facet_grid(
      rows=vars(spatial.layer),
      scales="free",
      space = "free",
      switch="y"
    ) +
    labs(x="Relative Abundance",y=NULL
    ) +
  ggtitle(
    "Decon5 - FG only - Raref. 700"
```

```
  )+
    theme(
      panel.background = element_blank(),
      # axis.text.y = element_blank(),
      legend.text = element_markdown(),
      panel.grid  = element_blank(),
      axis.ticks.y = element_blank(),
      strip.background = element_blank(),
      strip.switch.pad.grid = unit('00.1',"cm")
  )
```

```
##
## >>> Glomming by Class ... ... ...
```

```
# manually reorder
# p$data$Phylum =
#   factor(
#     p$data$Phylum,
#     levels = c("Pseudomonadota","Bacteroidota",
#                "Actinomycetota","Chlamydiota","Verrucomicrobiota","other 0.15" )
#     )
p$data$spatial.layer =
  factor(
    p$data$spatial.layer, levels = c("top","bottom")
  )
p$data$s.type =
  factor(
    p$data$s.type,
    levels = c("fungus garden", "trash", "food (cornmeal)")
  )
# print plot
p
```



TSinf

Rarefied to 3500 reads.

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

What are my sample variables?

```
sample_variables(ps)
```

```
##  [1] "seq.id"            "s.id"              "s.name"
##  [4] "s.type"            "treatment"         "ants.yn"
##  [7] "exp.rep"           "day.post.inf"      "expect.inf"
## [10] "extract.batch"     "pcr.plate"         "pcr.batch"
## [13] "dil.factor"        "Sample_or_Control" "host.species"
## [16] "inoc.dist"
```

```
save.image()
```

Which ASVs are going to be classified as "other"?

```
## Phyla <15% all samples filter
make_rel_abund(ps) %>% taxnames_lt_threshold(0.15)
```

```
##    [1] "ASV1300" "ASV1870" "ASV660"  "ASV561"  "ASV872"  "ASV1044" "ASV444"
##    [8] "ASV634"  "ASV419"  "ASV951"  "ASV827"  "ASV647"  "ASV1444" "ASV657"
##   [15] "ASV724"  "ASV131"  "ASV222"  "ASV1534" "ASV1817" "ASV974"  "ASV1543"
##   [22] "ASV1604" "ASV1488" "ASV123"  "ASV22"   "ASV965"  "ASV283"  "ASV1717"
##   [29] "ASV1508" "ASV624"  "ASV1468" "ASV1624" "ASV2211" "ASV520"  "ASV667"
##   [36] "ASV143"  "ASV2008" "ASV1120" "ASV1595" "ASV659"  "ASV452"  "ASV1236"
##   [43] "ASV346"  "ASV487"  "ASV1726" "ASV1019" "ASV1092" "ASV232"  "ASV776"
##   [50] "ASV403"  "ASV450"  "ASV68"   "ASV496"  "ASV275"  "ASV684"  "ASV888"
##   [57] "ASV651"  "ASV1349" "ASV1875" "ASV1554" "ASV1633" "ASV489"  "ASV375"
##   [64] "ASV944"  "ASV1583" "ASV1736" "ASV1283" "ASV1329" "ASV1160" "ASV541"
##   [71] "ASV1280" "ASV1158" "ASV271"  "ASV190"  "ASV734"  "ASV1141" "ASV627"
##   [78] "ASV918"  "ASV423"  "ASV73"   "ASV686"  "ASV189"  "ASV357"  "ASV1174"
##   [85] "ASV1070" "ASV900"  "ASV681"  "ASV28"   "ASV70"   "ASV1115" "ASV250"
##   [92] "ASV341"  "ASV264"  "ASV508"  "ASV609"  "ASV562"  "ASV145"  "ASV1328"
##   [99] "ASV55"   "ASV2077" "ASV2189" "ASV188"  "ASV225"  "ASV464"  "ASV2188"
##  [106] "ASV1292" "ASV108"  "ASV1811" "ASV161"  "ASV314"  "ASV1725" "ASV296"
##  [113] "ASV255"  "ASV971"  "ASV503"  "ASV1919" "ASV742"  "ASV1727" "ASV214"
##  [120] "ASV525"  "ASV164"  "ASV169"  "ASV1123" "ASV85"   "ASV1617" "ASV966"
##  [127] "ASV179"  "ASV470"  "ASV51"   "ASV1671" "ASV1931" "ASV672"  "ASV1469"
##  [134] "ASV1080" "ASV149"  "ASV584"  "ASV447"  "ASV184"  "ASV1142" "ASV1414"
##  [141] "ASV645"  "ASV653"  "ASV349"  "ASV132"  "ASV1773" "ASV1125" "ASV210"
##  [148] "ASV1430" "ASV238"  "ASV1625" "ASV1997" "ASV182"  "ASV1072" "ASV180"
##  [155] "ASV618"  "ASV622"  "ASV434"  "ASV157"  "ASV1109" "ASV191"  "ASV52"
```

```
##   [162] "ASV2187" "ASV608"  "ASV146"  "ASV71"   "ASV733"  "ASV309"  "ASV1905"
##   [169] "ASV2074" "ASV1799" "ASV579"  "ASV696"  "ASV1406" "ASV713"  "ASV1192"
##   [176] "ASV628"  "ASV1162" "ASV1124" "ASV1628" "ASV45"   "ASV1729" "ASV260"
##   [183] "ASV1504" "ASV286"  "ASV1257" "ASV1849" "ASV490"  "ASV707"  "ASV284"
##   [190] "ASV278"  "ASV833"  "ASV95"   "ASV708"  "ASV1672" "ASV505"  "ASV1490"
##   [197] "ASV1532" "ASV564"  "ASV254"  "ASV556"  "ASV150"  "ASV376"  "ASV118"
##   [204] "ASV2281" "ASV1772" "ASV794"  "ASV523"  "ASV758"  "ASV1441" "ASV921"
##   [211] "ASV249"  "ASV398"  "ASV439"  "ASV69"   "ASV2183" "ASV166"  "ASV279"
##   [218] "ASV1135" "ASV1933" "ASV522"  "ASV230"  "ASV414"  "ASV76"   "ASV261"
##   [225] "ASV135"  "ASV141"  "ASV127"  "ASV2221" "ASV1573" "ASV2081" "ASV1010"
##   [232] "ASV2261" "ASV2073" "ASV1215" "ASV1254" "ASV1995" "ASV2192" "ASV1613"
##   [239] "ASV1632" "ASV1105" "ASV1377" "ASV1298" "ASV880"  "ASV2093" "ASV1217"
##   [246] "ASV595"  "ASV1815" "ASV1433" "ASV1367" "ASV1623" "ASV1939" "ASV1861"
##   [253] "ASV739"  "ASV694"  "ASV1423" "ASV1118" "ASV442"  "ASV1866" "ASV825"
##   [260] "ASV1514" "ASV1582" "ASV695"  "ASV359"  "ASV824"  "ASV1378" "ASV908"
##   [267] "ASV1218" "ASV1363" "ASV322"  "ASV1319" "ASV717"  "ASV1176" "ASV570"
##   [274] "ASV621"  "ASV1249" "ASV1086" "ASV1664" "ASV1666" "ASV1299" "ASV1401"
##   [281] "ASV1778" "ASV1857" "ASV699"  "ASV674"  "ASV2057" "ASV1364" "ASV591"
##   [288] "ASV163"  "ASV344"  "ASV269"  "ASV569"  "ASV993"  "ASV1506" "ASV2185"
##   [295] "ASV1241" "ASV673"  "ASV137"  "ASV1320" "ASV319"  "ASV372"  "ASV364"
##   [302] "ASV81"   "ASV136"  "ASV590"  "ASV435"  "ASV23"   "ASV519"  "ASV1567"
##   [309] "ASV1028" "ASV1876" "ASV2065" "ASV1492" "ASV1770" "ASV1718" "ASV1816"
##   [316] "ASV1182" "ASV1563" "ASV1429" "ASV1379" "ASV1311" "ASV1621" "ASV576"
##   [323] "ASV641"  "ASV876"  "ASV750"  "ASV1529" "ASV1253" "ASV1146" "ASV2206"
##   [330] "ASV1313" "ASV1085" "ASV2063" "ASV2179" "ASV493"  "ASV498"  "ASV1670"
##   [337] "ASV1175" "ASV1025" "ASV565"  "ASV1116" "ASV1203" "ASV670"  "ASV1340"
##   [344] "ASV1296" "ASV972"  "ASV537"  "ASV969"  "ASV1002" "ASV960"  "ASV763"
##   [351] "ASV1400" "ASV1321" "ASV1307" "ASV1246" "ASV791"  "ASV1395" "ASV1341"
##   [358] "ASV1235" "ASV1614" "ASV690"  "ASV543"  "ASV1207" "ASV40"   "ASV1130"
##   [365] "ASV1936" "ASV36"   "ASV929"  "ASV1453" "ASV1479" "ASV1229" "ASV1372"
##   [372] "ASV941"  "ASV325"  "ASV1041" "ASV1027" "ASV1675" "ASV139"  "ASV201"
##   [379] "ASV2053" "ASV937"  "ASV909"  "ASV1245" "ASV1294" "ASV1355" "ASV844"
##   [386] "ASV2146" "ASV841"  "ASV959"  "ASV1084" "ASV2160" "ASV1996" "ASV1714"
##   [393] "ASV1712" "ASV253"  "ASV865"  "ASV1523" "ASV1535" "ASV1911" "ASV1494"
##   [400] "ASV858"  "ASV1850" "ASV1021" "ASV1581" "ASV1569" "ASV1003" "ASV1008"
##   [407] "ASV48"   "ASV1571" "ASV1710" "ASV113"  "ASV175"  "ASV1663" "ASV527"
##   [414] "ASV1117" "ASV1330" "ASV895"  "ASV1608" "ASV105"  "ASV1297" "ASV863"
##   [421] "ASV1172" "ASV1275" "ASV644"  "ASV1758" "ASV1800" "ASV936"  "ASV757"
##   [428] "ASV1612" "ASV39"   "ASV1538" "ASV775"  "ASV345"  "ASV823"  "ASV2191"
##   [435] "ASV438"  "ASV524"  "ASV573"  "ASV102"  "ASV1431" "ASV923"  "ASV826"
##   [442] "ASV1405" "ASV1032" "ASV902"  "ASV790"  "ASV2202" "ASV1659" "ASV2088"
##   [449] "ASV1869" "ASV2190" "ASV2082" "ASV1190" "ASV111"  "ASV1202" "ASV2167"
##   [456] "ASV2007" "ASV2204" "ASV369"  "ASV723"  "ASV1938" "ASV2097" "ASV1055"
##   [463] "ASV441"  "ASV350"  "ASV935"  "ASV828"  "ASV1371" "ASV1662" "ASV1813"
##   [470] "ASV1306" "ASV1801" "ASV347"  "ASV1611" "ASV187"  "ASV1462" "ASV371"
##   [477] "ASV1011" "ASV1998" "ASV1530" "ASV697"  "ASV1201" "ASV665"  "ASV114"
##   [484] "ASV305"  "ASV1071" "ASV1616" "ASV2096" "ASV1362" "ASV1537" "ASV120"
##   [491] "ASV138"  "ASV970"  "ASV1937" "ASV1860" "ASV1424" "ASV1660" "ASV1415"
##   [498] "ASV1738" "ASV1910" "ASV2166" "ASV1871" "ASV272"  "ASV1159" "ASV1858"
##   [505] "ASV35"   "ASV1541" "ASV631"  "ASV1121" "ASV1252" "ASV1144" "ASV205"
##   [512] "ASV1234" "ASV552"  "ASV404"  "ASV906"  "ASV178"  "ASV1134" "ASV1491"
##   [519] "ASV1314" "ASV121"  "ASV171"  "ASV1798" "ASV532"  "ASV560"  "ASV336"
##   [526] "ASV1089" "ASV1067" "ASV509"  "ASV385"  "ASV1539" "ASV612"  "ASV1180"
##   [533] "ASV531"  "ASV581"  "ASV32"   "ASV594"  "ASV106"  "ASV600"  "ASV1360"
```

```
##  [540] "ASV1796" "ASV930"  "ASV1119" "ASV521"  "ASV593"  "ASV142"  "ASV358"
##  [547] "ASV500"  "ASV518"  "ASV460"  "ASV1221" "ASV463"  "ASV1083" "ASV868"
##  [554] "ASV884"  "ASV1233" "ASV1398" "ASV195"  "ASV1157" "ASV764"  "ASV1163"
##  [561] "ASV901"  "ASV421"  "ASV1214" "ASV846"  "ASV1375" "ASV557"  "ASV1580"
##  [568] "ASV515"  "ASV1091" "ASV1610" "ASV1322" "ASV553"  "ASV1333" "ASV1194"
##  [575] "ASV716"  "ASV952"  "ASV92"   "ASV2002" "ASV2169" "ASV1090" "ASV1495"
##  [582] "ASV642"  "ASV234"  "ASV1600" "ASV1959" "ASV1531" "ASV568"  "ASV144"
##  [589] "ASV1057" "ASV1629" "ASV687"  "ASV469"  "ASV664"  "ASV399"  "ASV567"
##  [596] "ASV1361" "ASV736"  "ASV693"  "ASV2006" "ASV2076" "ASV213"  "ASV2175"
##  [603] "ASV2182" "ASV689"  "ASV834"  "ASV2181" "ASV308"  "ASV1006" "ASV1502"
##  [610] "ASV803"  "ASV363"  "ASV2068" "ASV461"  "ASV386"  "ASV516"  "ASV480"
##  [617] "ASV1679" "ASV542"  "ASV997"  "ASV1108" "ASV1197" "ASV1527" "ASV1809"
##  [624] "ASV473"  "ASV585"  "ASV1056" "ASV1368" "ASV1394" "ASV1136" "ASV1147"
##  [631] "ASV212"  "ASV922"  "ASV416"  "ASV1760" "ASV427"  "ASV1722" "ASV847"
##  [638] "ASV1661" "ASV440"  "ASV1397" "ASV1767" "ASV74"   "ASV1422" "ASV1711"
##  [645] "ASV43"   "ASV1279" "ASV1865" "ASV1177" "ASV497"  "ASV1276" "ASV1477"
##  [652] "ASV1501" "ASV801"  "ASV599"  "ASV1195" "ASV1181" "ASV1219" "ASV494"
##  [659] "ASV931"  "ASV981"  "ASV499"  "ASV671"  "ASV714"  "ASV1102" "ASV957"
##  [666] "ASV815"  "ASV1768" "ASV718"  "ASV859"  "ASV1178" "ASV1496" "ASV1220"
##  [673] "ASV1464" "ASV1402" "ASV939"  "ASV495"  "ASV821"  "ASV950"  "ASV688"
##  [680] "ASV887"  "ASV384"  "ASV656"  "ASV295"  "ASV1615" "ASV1047" "ASV1719"
##  [687] "ASV1009" "ASV967"  "ASV1196" "ASV1852" "ASV1012" "ASV1243" "ASV1323"
##  [694] "ASV1013" "ASV940"  "ASV1516" "ASV751"  "ASV994"  "ASV165"  "ASV335"
##  [701] "ASV1515" "ASV598"  "ASV712"  "ASV692"  "ASV725"  "ASV1043" "ASV432"
##  [708] "ASV510"  "ASV475"  "ASV1334" "ASV619"  "ASV148"  "ASV47"   "ASV666"
##  [715] "ASV862"  "ASV891"  "ASV332"  "ASV722"  "ASV995"  "ASV961"  "ASV388"
##  [722] "ASV247"  "ASV830"  "ASV1989" "ASV396"  "ASV504"  "ASV856"  "ASV1352"
##  [729] "ASV1553" "ASV546"  "ASV1914" "ASV2180" "ASV539"  "ASV1399" "ASV1949"
##  [736] "ASV2069" "ASV1224" "ASV842"  "ASV1542" "ASV257"  "ASV548"  "ASV1380"
##  [743] "ASV1669" "ASV1823" "ASV1273" "ASV422"  "ASV373"  "ASV477"  "ASV1310"
##  [750] "ASV1436" "ASV2051" "ASV1804" "ASV2210" "ASV1986" "ASV807"  "ASV1921"
##  [757] "ASV2163" "ASV1992" "ASV151"  "ASV331"  "ASV2066" "ASV18"   "ASV1812"
##  [764] "ASV1005" "ASV323"  "ASV67"   "ASV293"  "ASV75"   "ASV352"  "ASV1499"
##  [771] "ASV1277" "ASV1096" "ASV459"  "ASV64"   "ASV875"  "ASV726"  "ASV1231"
##  [778] "ASV506"  "ASV125"  "ASV259"  "ASV66"   "ASV1122" "ASV158"  "ASV559"
##  [785] "ASV239"  "ASV89"   "ASV395"  "ASV46"   "ASV428"  "ASV649"  "ASV774"
##  [792] "ASV54"   "ASV265"  "ASV11"   "ASV122"  "ASV42"   "ASV1448" "ASV1631"
##  [799] "ASV467"  "ASV351"  "ASV252"  "ASV379"  "ASV563"  "ASV243"  "ASV1052"
##  [806] "ASV100"  "ASV277"  "ASV99"   "ASV691"  "ASV1054" "ASV588"  "ASV431"
##  [813] "ASV709"  "ASV703"  "ASV1232" "ASV1035" "ASV980"  "ASV864"  "ASV620"
##  [820] "ASV643"  "ASV852"  "ASV626"  "ASV478"  "ASV313"  "ASV297"  "ASV107"
##  [827] "ASV2209" "ASV1525" "ASV1920" "ASV405"  "ASV745"  "ASV954"  "ASV33"
##  [834] "ASV1154" "ASV31"   "ASV1295" "ASV1173" "ASV390"  "ASV436"  "ASV87"
##  [841] "ASV387"  "ASV223"  "ASV640"  "ASV744"  "ASV307"  "ASV753"  "ASV1079"
##  [848] "ASV1140" "ASV1440" "ASV1874" "ASV2078" "ASV658"  "ASV2059" "ASV424"
##  [855] "ASV589"  "ASV1899" "ASV536"  "ASV996"  "ASV1274" "ASV281"  "ASV1486"
##  [862] "ASV1984" "ASV1585" "ASV410"  "ASV755"  "ASV1877" "ASV1619" "ASV370"
##  [869] "ASV104"  "ASV204"  "ASV211"  "ASV155"  "ASV867"  "ASV273"  "ASV1104"
##  [876] "ASV117"  "ASV534"  "ASV583"  "ASV380"  "ASV646"  "ASV2176" "ASV623"
##  [883] "ASV767"  "ASV1864" "ASV727"  "ASV1179" "ASV2075" "ASV874"  "ASV168"
##  [890] "ASV1985" "ASV84"   "ASV1466" "ASV1278" "ASV632"  "ASV448"  "ASV663"
##  [897] "ASV454"  "ASV288"  "ASV2164" "ASV1198" "ASV1677" "ASV920"  "ASV1061"
##  [904] "ASV1438" "ASV730"  "ASV2004" "ASV1200" "ASV1607" "ASV1945" "ASV572"
##  [911] "ASV285"  "ASV1731" "ASV1242" "ASV533"  "ASV268"  "ASV2085" "ASV938"
```

```
##  [918] "ASV1577" "ASV1764" "ASV1315" "ASV1872" "ASV1039" "ASV1317" "ASV896"
##  [925] "ASV1318" "ASV1771" "ASV942"  "ASV853"  "ASV517"  "ASV1734" "ASV1088"
##  [932] "ASV1500" "ASV840"  "ASV1668" "ASV1087" "ASV812"  "ASV1667" "ASV1560"
##  [939] "ASV985"  "ASV2005" "ASV1806" "ASV1193" "ASV1497" "ASV1369" "ASV1721"
##  [946] "ASV1336" "ASV990"  "ASV873"  "ASV1308" "ASV1403" "ASV1961" "ASV1941"
##  [953] "ASV855"  "ASV1016" "ASV1584" "ASV1634" "ASV1707" "ASV1720" "ASV784"
##  [960] "ASV1526" "ASV19"   "ASV2079" "ASV1434" "ASV1465" "ASV1863" "ASV86"
##  [967] "ASV1037" "ASV1461" "ASV1480" "ASV2011" "ASV1370" "ASV2178" "ASV177"
##  [974] "ASV729"  "ASV2184" "ASV1716" "ASV1060" "ASV1425" "ASV1930" "ASV606"
##  [981] "ASV2080" "ASV710"  "ASV1446" "ASV1873" "ASV94"   "ASV140"  "ASV1618"
##  [988] "ASV850"  "ASV280"  "ASV1100" "ASV2207" "ASV1763" "ASV1940" "ASV1818"
##  [995] "ASV1820" "ASV779"  "ASV991"  "ASV1819" "ASV1432" "ASV1732" "ASV1342"
## [1002] "ASV1216" "ASV1622" "ASV1335" "ASV1390" "ASV2086" "ASV1932" "ASV2003"
## [1009] "ASV1924" "ASV1821" "ASV194"  "ASV229"  "ASV1902" "ASV1620" "ASV1810"
## [1016] "ASV2071" "ASV2212" "ASV1635" "ASV2092" "ASV2012" "ASV1404" "ASV2278"
## [1023] "ASV1880" "ASV1944" "ASV1724" "ASV339"  "ASV397"  "ASV2072" "ASV1026"
## [1030] "ASV1606" "ASV315"  "ASV485"  "ASV310"  "ASV529"  "ASV1925" "ASV1255"
## [1037] "ASV1678" "ASV1676" "ASV1376" "ASV1437" "ASV2001" "ASV811"  "ASV2155"
## [1044] "ASV636"  "ASV2064" "ASV1145" "ASV1650" "ASV2049" "ASV59"   "ASV156"
## [1051] "ASV126"  "ASV2162" "ASV2276" "ASV301"  "ASV227"  "ASV1189" "ASV892"
## [1058] "ASV2170" "ASV1665" "ASV1766" "ASV2070" "ASV1024" "ASV843"  "ASV53"
## [1065] "ASV1359" "ASV1493" "ASV1570" "ASV1452" "ASV1281" "ASV1999" "ASV1967"
## [1072] "ASV1733" "ASV746"  "ASV785"  "ASV719"  "ASV1169" "ASV903"  "ASV486"
## [1079] "ASV698"  "ASV2052" "ASV160"  "ASV2260" "ASV1451" "ASV1709" "ASV27"
## [1086] "ASV1247" "ASV992"  "ASV1103" "ASV1213" "ASV367"  "ASV72"   "ASV1928"
## [1093] "ASV16"   "ASV1301" "ASV1230" "ASV408"  "ASV1439" "ASV1435" "ASV1878"
## [1100] "ASV38"   "ASV465"  "ASV919"  "ASV128"  "ASV88"   "ASV2269" "ASV2256"
## [1107] "ASV451"  "ASV2259"
```

Mean relabund per class

```
ps.m = make_rel_abund(ps) %>%
  per_sample_taxafilt(threshold=0.01, glom="Class") %>%
  ps_melt()
```

```
##
## >>> Glomming by Class ... ... ...
```

```
ps.m %>% group_by(Class) %>% summarise(mean.rab = mean(Abundance)*100) %>% arrange(desc(mean.rab))
```

```
## # A tibble: 9 x 2
##   Class               mean.rab
##   <chr>                  <dbl>
## 1 Gammaproteobacteria  61.3
## 2 Clostridia           25.5
## 3 other 0.01            7.76
## 4 Alphaproteobacteria   3.23
## 5 Bacteroidia           0.740
## 6 Chlamydiia            0.681
## 7 Verrucomicrobiia      0.569
## 8 Bacilli               0.152
## 9 Actinobacteria        0.0286
```

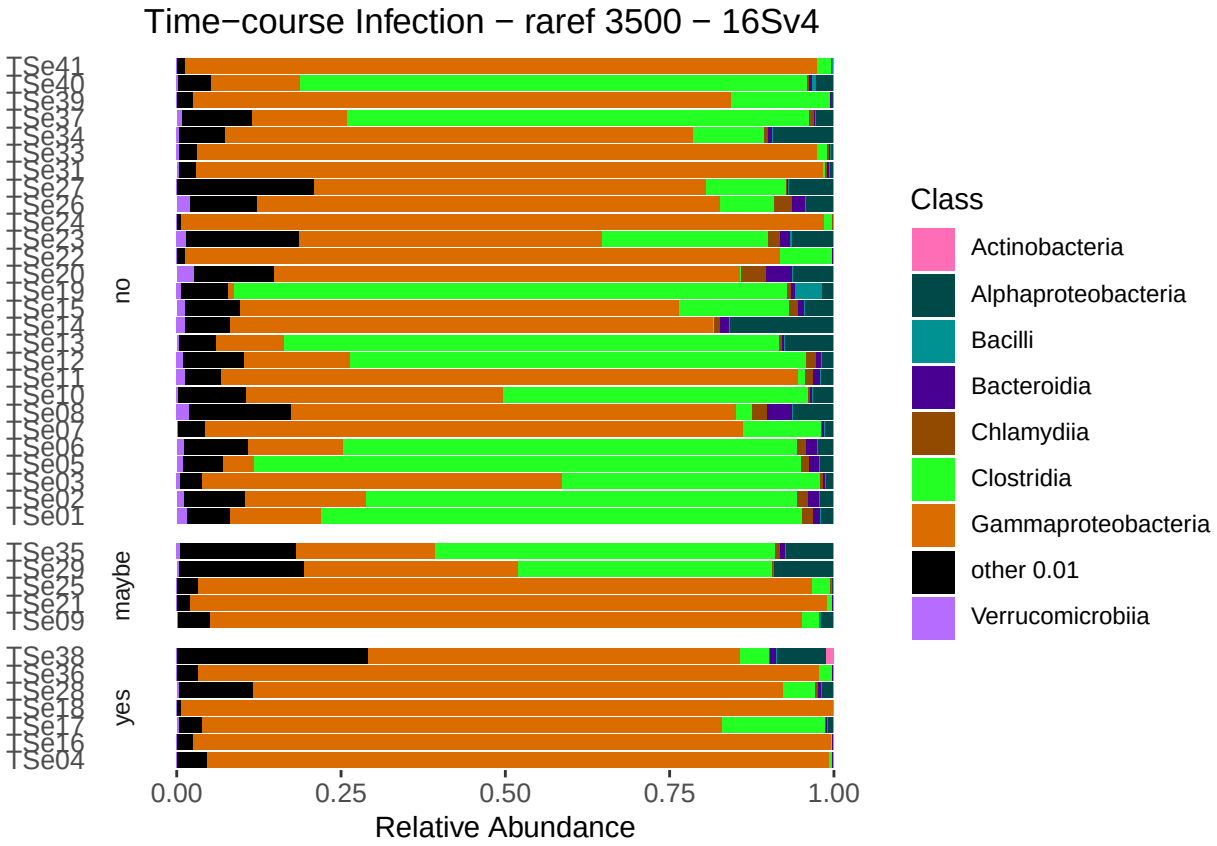
```
#save.image()
```

Plot!

```
## ---- stacked bar plot ---- ##
p <-
    make_rel_abund(ps) %>%
    per_sample_taxafilt(threshold=0.01, glom="Class") %>%
    # ps_filter(s.type == "fungus garden", .keep_all_taxa=T) %>%
    psmelt() %>%
    ggplot(
        aes(x = Abundance, y = s.id, fill = Class)
    ) +
    geom_bar(stat = "identity"
    ) +
    theme(axis.text = element_text(size = 10)
    ) +
    scale_fill_manual(
      # values = c(unname(paletteMartin[2:3]),"black")
      values = class.cols.martin
    ) +
    facet_grid(
      # rows = vars(treatment,ants.yn),
      rows = vars(expect.inf),
      scales = "free",
      space = "free",
      switch = "y"
    ) +
    labs(x = "Relative Abundance", y = NULL
    ) +
  ggtitle(
    "Time-course Infection - raref 3500 - 16Sv4"
  )+
    theme(
      panel.background = element_blank(),
      # axis.text.y = element_blank(),
      legend.text = element_markdown(),
      panel.grid  = element_blank(),
      axis.ticks.y = element_blank(),
      strip.background = element_blank(),
      strip.switch.pad.grid = unit('00.1',"cm")
  )
```

```
##
## >>> Glomming by Class ... ... ...
```

```
# manually reorder
p$data$expect.inf =
  factor(
    p$data$expect.inf,
    levels = c("no", "maybe", "yes")
  )
p
```



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"
```

```
## ---- Data Wrangling ----------------------------------------------------------------------- ##
psList.raref.rab = make_rel_abund(psList.raref)
psList.raref.rab.gglom = lapply(psList.raref.rab, tax_glom, "Genus")

# set taxa that are < threshold abundant in all sample to "other"
psList.raref.rab.1pctOth.gglom =
  lapply( psList.raref.rab,               # data list
          per_sample_taxafilt,            # function
          threshold=0.01, glom="Genus" ) # other function parameters
```

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

```
##
## >>> Glomming by Genus ... ... ...
psList.melt =
  lapply(psList.raref.rab.1pctOth.gglom,
         psmelt )
```

Decon3

Data wrangling

```
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
      )
  )
```

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)"),
                 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...

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

```
# 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
```

```
## d3e46 "#ffff6d" "#6db6ff"
## d3e27 "#ffff6d" "#6db6ff"
## d3e19 "#ffff6d" "#6db6ff"
## d3e26 "#924900" "#924900"
## d3e45 "#ffff6d" "#ffb6db"
## d3e33 "#ffff6d" "#b6dbff"
## d3e03 "#ffff6d" "#ff6db6"
## d3e21 "#ffff6d" "#ff6db6"
## d3e22 "#ffff6d" "#ff6db6"
## d3e39 "#ffff6d" "#ff6db6"
## d3e08 "#924900" "#920000"
## d3e41 "#ffff6d" "#ffb6db"
## d3e32 "#ffff6d" "#b6dbff"
## d3e30 "#009292" "#004949"
## d3e01 "#ffff6d" "#ffb6db"
## d3e10 "#924900" "#db6d00"
## d3e14 "#ffff6d" "#ffb6db"
## d3e29 "#ffff6d" "#ff6db6"
```

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

Create massive color legend...

```
## ---- 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 ------------------------------------------------------------------------- ##

# 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
  # )
```

Decon4

Data wrangling...

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

Color wrangling...

Get color codes from desired palette:

```
# paletteMartin
# # inoc.dist full range
# sample_data(psList$decon4)$inoc.dist %>% sort() %>% unique()
# Blue2DarkOrange18Steps
# # [1] "#006666" "#009999" "#00CCCC" "#00FFFF" "#33FFFF" "#65FFFF" "#99FFFF" "#B2FFFF"
# #  [9] "#CBFFFF" "#E5FFFF" "#FFE5CB" "#FFCA99" "#FFAD65" "#FF8E33" "#FF6E00" "#CC5500"
# # [17] "#993D00" "#662700"
#
# full.inoc.dist.cols =
#   data.frame(
#     inoc.dist = sample_data(psList$decon4)$inoc.dist %>% sort() %>% unique(),
#     col = c("black",Blue2DarkOrange18Steps[1:17])
#   )

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)]
}

mypal <- colorRampPalette( c( "darkgreen","white") )( 50 )
x = sample_data(psList$decon4)$inoc.dist.cm %>% sort() %>% unique()
# x[-1] bc setting -1 to black
map2color(x[-1],mypal)
```

```
##  [1] "#006400" "#247A24" "#348334" "#489048" "#4E934E" "#539653" "#68A368"
##  [8] "#77AC77" "#7CAF7C" "#7CAF7C" "#87B687" "#96BF96" "#9CC29C" "#A1C6A1"
## [15] "#A1C6A1" "#ABCCAB" "#B0CFB0" "#B6D2B6" "#BBD5BB" "#C5DCC5" "#CADFCA"
## [22] "#CADFCA" "#D0E2D0" "#D0E2D0" "#DFECDF" "#E4EFE4" "#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"

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

```
## ---- 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(trashed.yn =
         data.frame(
           trashed.yn = c("yes","no"),
           col = c("#924900","#ffff6d")), # random cols for trashed.yn
       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
```

```
## $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 #247A24
## 4          2.77 #348334
## 5          4.09 #489048
## 6          4.30 #4E934E
## 7          4.64 #539653
## 8          5.54 #68A368
## 9          6.45 #77AC77
## 10         6.68 #7CAF7C
## 11         6.76 #7CAF7C
## 12         7.33 #87B687
## 13         8.18 #96BF96
## 14         8.31 #9CC29C
## 15         8.60 #A1C6A1
## 16         8.77 #A1C6A1
## 17         9.28 #ABCCAB
## 18         9.49 #B0CFB0
## 19         9.70 #B6D2B6
## 20        10.06 #BBD5BB
## 21        10.75 #C5DCC5
## 22        10.83 #CADFCA
## 23        10.89 #CADFCA
## 24        11.08 #D0E2D0
## 25        11.30 #D0E2D0
## 26        11.95 #DFECDF
## 27        12.23 #E4EFE4
## 28        12.81 #EFF5EF
## 29        13.85 #FFFFFF
```

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

```
## Joining with `by = join_by(trashed.yn)`
## 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$extr.batch
#       ) %>% select(s.id, extr.batch = col),
#     join_by(s.id)
#   ) # [s.id, trashed.yn, inoc.dist, extr.batch]
#
# tmp.j =
#   full_join(
#     tmp.j,
#     left_join(
#       ps.m %>% select(s.id, pcr.batch) %>% group_by(s.id) %>% distinct(),
#       metadat.cols$pcr.batch
#       ) %>% select(s.id, pcr.batch = col),
#     join_by(s.id)
#   ) # [s.id, trashed.yn, inoc.dist, 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
```

```
##       trashed.yn inoc.dist.cm
## d4e10 "#924900"  "#96BF96"
## d4e04 "#924900"  "#C5DCC5"
## d4e22 "#924900"  "#348334"
## d4e15 "#924900"  "#A1C6A1"
## d4e03 "#924900"  "#247A24"
```

```
## d4e09 "#924900"  "#7CAF7C"
## d4e16 "#ffff6d"  "#ABCCAB"
## d4e19 "#924900"  "#489048"
## d4e29 "#924900"  "#4E934E"
## d4e18 "#924900"  "#7CAF7C"
## d4e06 "#ffff6d"  "#E4EFE4"
## d4e25 "#924900"  "#006400"
## d4e14 "#ffff6d"  "#CADFCA"
## d4e13 "#ffff6d"  "#BBD5BB"
## d4e28 "#ffff6d"  "#D0E2D0"
```

```
## ---- 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,] "trashed.yn"   "white"
##  [2,] "yes"          "#924900"
##  [3,] "no"           "#ffff6d"
##  [4,] ""             "white"
##  [5,] "inoc.dist.cm" "white"
##  [6,] "-1"           "black"
##  [7,] "0"            "#006400"
##  [8,] "2.15"         "#247A24"
##  [9,] "2.77"         "#348334"
## [10,] "4.09"         "#489048"
## [11,] "4.3"          "#4E934E"
## [12,] "4.64"         "#539653"
## [13,] "5.54"         "#68A368"
## [14,] "6.45"         "#77AC77"
## [15,] "6.68"         "#7CAF7C"
## [16,] "6.76"         "#7CAF7C"
## [17,] "7.33"         "#87B687"
## [18,] "8.18"         "#96BF96"
## [19,] "8.31"         "#9CC29C"
## [20,] "8.6"          "#A1C6A1"
## [21,] "8.77"         "#A1C6A1"
## [22,] "9.28"         "#ABCCAB"
```

```
## [23,] "9.49"         "#B0CFB0"
## [24,] "9.7"          "#B6D2B6"
## [25,] "10.06"        "#BBD5BB"
## [26,] "10.75"        "#C5DCC5"
## [27,] "10.83"        "#CADFCA"
## [28,] "10.89"        "#CADFCA"
## [29,] "11.08"        "#D0E2D0"
## [30,] "11.3"         "#D0E2D0"
## [31,] "11.95"        "#DFECDF"
## [32,] "12.23"        "#E4EFE4"
## [33,] "12.81"        "#EFF5EF"
## [34,] "13.85"        "#FFFFFF"
## [35,] ""             "white"
```

Plotting . . .

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

```
## ---- 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

Data wrangling...

```
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, "top")
#   ) #%>% select(s.name, spatial.layer) %>% filter(s.name %in% tb.samples)
```

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"
```

```
## ---- 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
```

```
## ---- 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
## d5e05 "#db6d00" "#6db6ff"
## d5e11 "#db6d00" "#ff6db6"
## d5e49 "#db6d00" "#6db6ff"
## d5e06 "#db6d00" "#6db6ff"
## d5e03 "#924900" "black"
## d5e21 "#db6d00" "#6db6ff"
## d5e19 "#db6d00" "#6db6ff"
## d5e08 "#db6d00" "#ff6db6"
## d5e29 "#ffff6d" "black"
## d5e28 "#db6d00" "#6db6ff"
## d5e51 "#db6d00" "#6db6ff"
## d5e44 "#db6d00" "#6db6ff"
## d5e02 "#db6d00" "#6db6ff"
## d5e07 "#db6d00" "#6db6ff"
## d5e38 "#db6d00" "#ff6db6"
## d5e01 "#db6d00" "#ff6db6"
## d5e31 "#db6d00" "#ff6db6"
## d5e18 "#db6d00" "#6db6ff"
## d5e42 "#db6d00" "#ff6db6"
## d5e13 "#db6d00" "#6db6ff"
## d5e04 "#db6d00" "#ff6db6"
## d5e14 "#db6d00" "#6db6ff"
## d5e48 "#db6d00" "#6db6ff"
## d5e32 "#db6d00" "#ff6db6"
## d5e09 "#db6d00" "#6db6ff"
## d5e12 "#db6d00" "#6db6ff"
## d5e45 "#db6d00" "#6db6ff"
## d5e27 "#db6d00" "#ff6db6"
## d5e37 "#db6d00" "#6db6ff"
## d5e46 "#db6d00" "#ff6db6"
## d5e39 "#db6d00" "#ff6db6"
## d5e16 "#db6d00" "#ff6db6"
## d5e50 "#db6d00" "#6db6ff"
## d5e15 "#db6d00" "#6db6ff"
## d5e20 "#db6d00" "#6db6ff"
## d5e41 "#db6d00" "#6db6ff"
```

```
## d5e36 "#db6d00" "#6db6ff"
```

```
## ---- 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.

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

TSinf

Data wrangling...

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

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

```
  mutate(
    expect.inf = replace(expect.inf,
                         s.id == "TSe38",
                         "yes")
    ) %>%
  mutate(
    expect.inf = replace(expect.inf,
                         s.id == "TSe35",
                         "maybe")
  )
```

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

```
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)`
```

```
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)`
```

```
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)`
```

```
# 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
## TSe33 "#db6d00" "#004949" "#24ff24"    "#920000" "#b6dbff"
## TSe18 "#db6d00" "#004949" "#009292"    "black"   "#004949"
## TSe41 "#db6d00" "#6db6ff" "#ffff6d"    "#920000" "#b6dbff"
## TSe07 "#db6d00" "#6db6ff" "#24ff24"    "#920000" "#b6dbff"
## TSe28 "#db6d00" "#004949" "#24ff24"    "black"   "#004949"
## TSe17 "#db6d00" "#004949" "#009292"    "black"   "#004949"
## TSe36 "#db6d00" "#004949" "#24ff24"    "black"   "#004949"
## TSe25 "#924900" "#6db6ff" "#009292"    "#920000" "#009292"
## TSe20 "#db6d00" "grey"    "#009292"    "#920000" "#b6dbff"
## TSe26 "#db6d00" "#6db6ff" "#009292"    "#920000" "#b6dbff"
## TSe08 "#db6d00" "grey"    "#009292"    "#920000" "#b6dbff"
## TSe11 "#db6d00" "#6db6ff" "#24ff24"    "#920000" "#b6dbff"
## TSe34 "#db6d00" "grey"    "#ffff6d"    "#920000" "#b6dbff"
## TSe38 "#db6d00" "grey"    "#009292"    "black"   "#004949"
## TSe27 "#db6d00" "grey"    "#24ff24"    "black"   "#b6dbff"
## TSe22 "#db6d00" "#6db6ff" "#ffff6d"    "black"   "#b6dbff"
## TSe05 "#db6d00" "#004949" "#009292"    "#920000" "#b6dbff"
## TSe14 "#db6d00" "grey"    "#24ff24"    "#920000" "#b6dbff"
## TSe16 "#924900" "#004949" "#009292"    "#920000" "#004949"
## TSe24 "#db6d00" "#6db6ff" "#009292"    "black"   "#b6dbff"
## TSe31 "#db6d00" "#004949" "#24ff24"    "#920000" "#b6dbff"
## TSe39 "#db6d00" "#6db6ff" "#009292"    "black"   "#b6dbff"
## TSe09 "#924900" "grey"    "#009292"    "#920000" "#009292"
## TSe23 "#db6d00" "#004949" "#ffff6d"    "black"   "#b6dbff"
## TSe12 "#db6d00" "grey"    "#24ff24"    "#920000" "#b6dbff"
## TSe02 "#db6d00" "#004949" "#ffff6d"    "#920000" "#b6dbff"
## TSe01 "#db6d00" "#6db6ff" "#ffff6d"    "#920000" "#b6dbff"
## TSe19 "#db6d00" "grey"    "#ffff6d"    "#920000" "#b6dbff"
## TSe04 "#924900" "#004949" "#009292"    "#920000" "#004949"
## TSe06 "#db6d00" "grey"    "#009292"    "black"   "#b6dbff"
## TSe40 "#db6d00" "#6db6ff" "#ffff6d"    "black"   "#b6dbff"
## TSe10 "#db6d00" "#6db6ff" "#24ff24"    "black"   "#b6dbff"
## TSe21 "#924900" "#6db6ff" "#009292"    "#920000" "#009292"
## TSe15 "#db6d00" "#004949" "#ffff6d"    "black"   "#b6dbff"
## TSe13 "#db6d00" "grey"    "#ffff6d"    "black"   "#b6dbff"
```

```
## TSe03 "#db6d00" "#6db6ff" "#24ff24"    "black"   "#b6dbff"
## TSe29 "#924900" "grey"    "#009292"    "#920000" "#009292"
## TSe35 "#db6d00" "grey"    "#24ff24"    "black"   "#009292"
## TSe37 "#db6d00" "grey"    "#ffff6d"    "black"   "#b6dbff"
```

```
## ---- 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()
# getwd()
# ## ---- 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$decon3
```

What are my sample variables?

```
sample_variables(ps)
```

```
##  [1] "seq.id"           "s.id"             "s.name"
##  [4] "s.type"           "fg.coord"         "spatial.layer"
##  [7] "spatial.col"      "spatial.row"      "trashed.yn"
```

```
## [10] "inoc.dist"         "inoc.dist.cm"      "extr.batch"
## [13] "pcr.plate"         "pcr.batch"         "dil.factor"
## [16] "Sample_or_Control" "host.species"
```

Violin plot

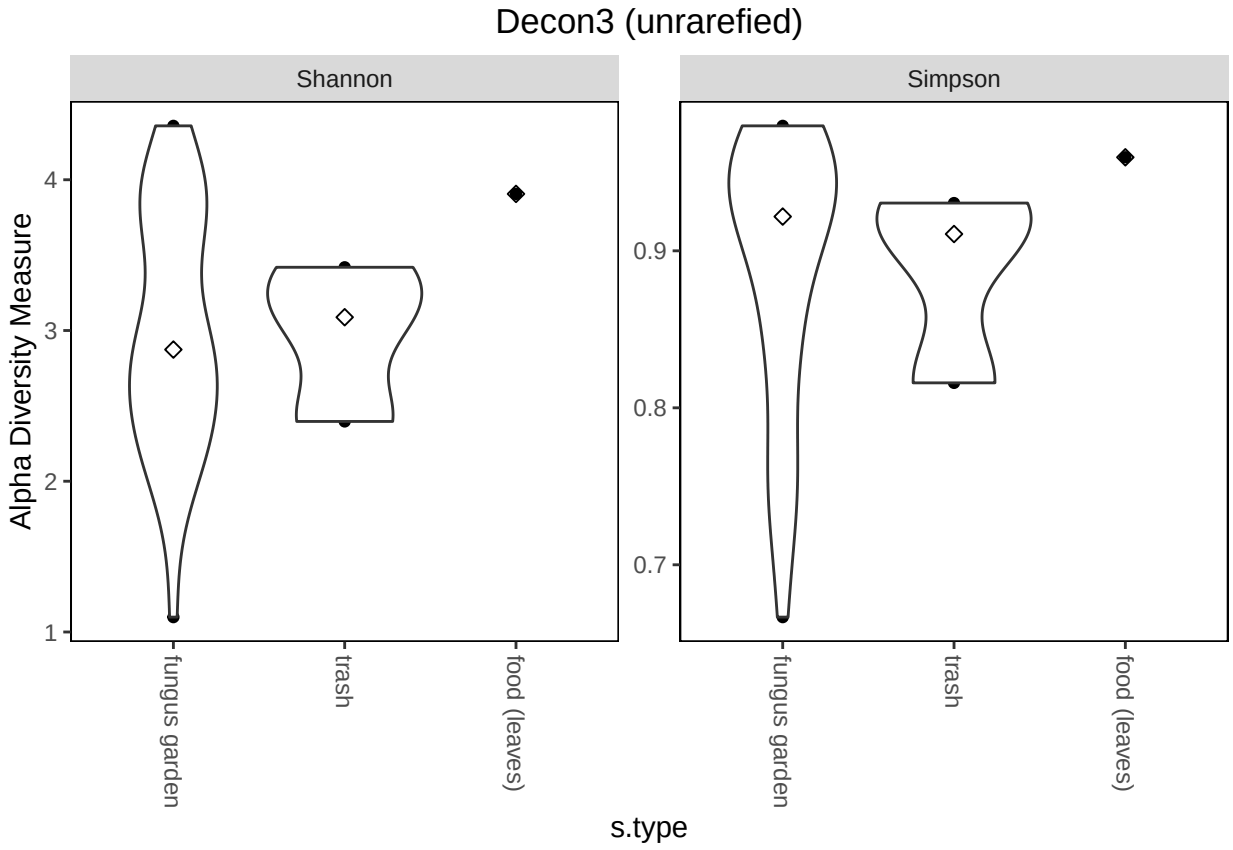
```
p <-
  ps %>%
  plot_richness(., x = "s.type",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_violin(
  ) +
  ggtitle("Decon3 (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

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

```
## 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.
```



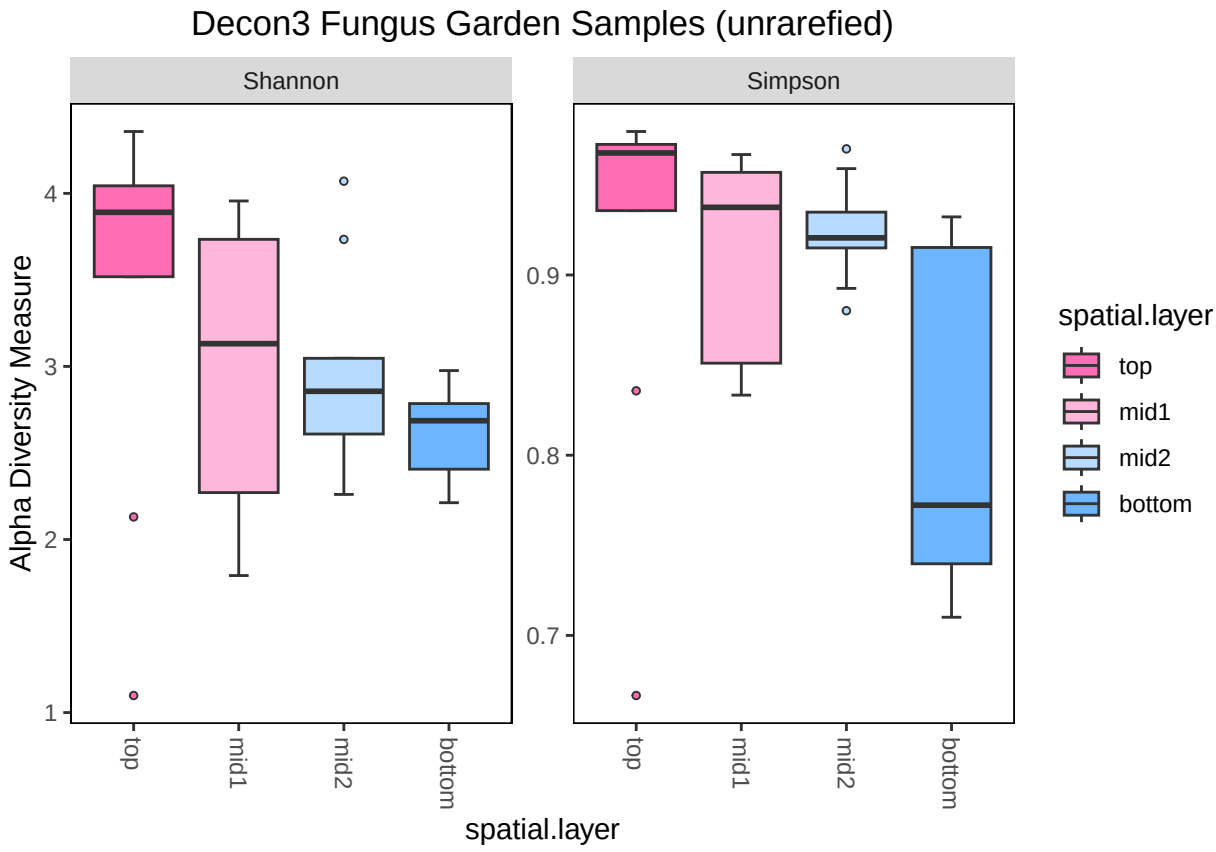

```
paletteMartin
```

```
##         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"
```

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

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

p <-
  ps %>%
  subset_samples(s.type == "fungus garden") %>%
  plot_richness(., x = "spatial.layer",
                measures = c("Shannon", "Simpson"),

  ) +
  # geom_violin(
  # ) +
  geom_boxplot(
    aes(fill = spatial.layer),
```

```
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25

  ) +
  # geom_point(
  #   shape = 21,
  #   color = "black",
  #   alpha = 0.5
  # ) +
  scale_fill_manual(
    values = unlist(spatial.layer.col.ls)
  ) +
  # scale_color_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)

  ) #+
  # stat_summary(
  #   fun.y= median,
  #   geom="point",
  #   shape=23,
  #   size=2
  # )
p$data$spatial.layer = factor(p$data$spatial.layer, levels=c("top","mid1","mid2","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
ps.adiv.df = ps.adiv.df %>% filter(s.type != "food (leaves)") # removing bc only 1 samples
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     1  0.0023 0.00230  0.0037 0.9517
## Residuals 37 22.8974 0.61885
##
## $sim
## Analysis of Variance Table
##
## Response: Simpson
##           Df   Sum Sq   Mean Sq F value Pr(>F)
## s.type     1 0.000033 0.0000335  0.0045 0.9466
```

```
## Residuals 37 0.272462 0.0073638
```

```
#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  3.184 1.06133  1.7718 0.1724
## Residuals     32 19.169 0.59903
##
## $sim
## Analysis of Variance Table
##
## Response: Simpson
##               Df   Sum Sq   Mean Sq F value   Pr(>F)
## spatial.layer  3 0.080365 0.0267884  4.6438 0.008336 **
## Residuals     32 0.184598 0.0057687
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

TukeyHSD

```
TukeyHSD(aov(Simpson ~ spatial.layer, ps.adiv.df.fg))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Simpson ~ spatial.layer, data = ps.adiv.df.fg)
##
## $spatial.layer
##                      diff          lwr        upr     p adj
## mid1-bottom  0.098724629  0.001718515 0.19573074 0.0448499
## mid2-bottom  0.115516744  0.018510629 0.21252286 0.0145715
## top-bottom   0.110376968  0.013370853 0.20738308 0.0207848
## mid2-mid1    0.016792114 -0.080214001 0.11379823 0.9653173
## top-mid1     0.011652339 -0.085353776 0.10865845 0.9878670
## top-mid2    -0.005139775 -0.102145890 0.09186634 0.9989219
```

```
# TukeyHSD(aov(InvSimpson ~ spatial.layer, ps.adiv.df.fg))
```

d3-Outlier

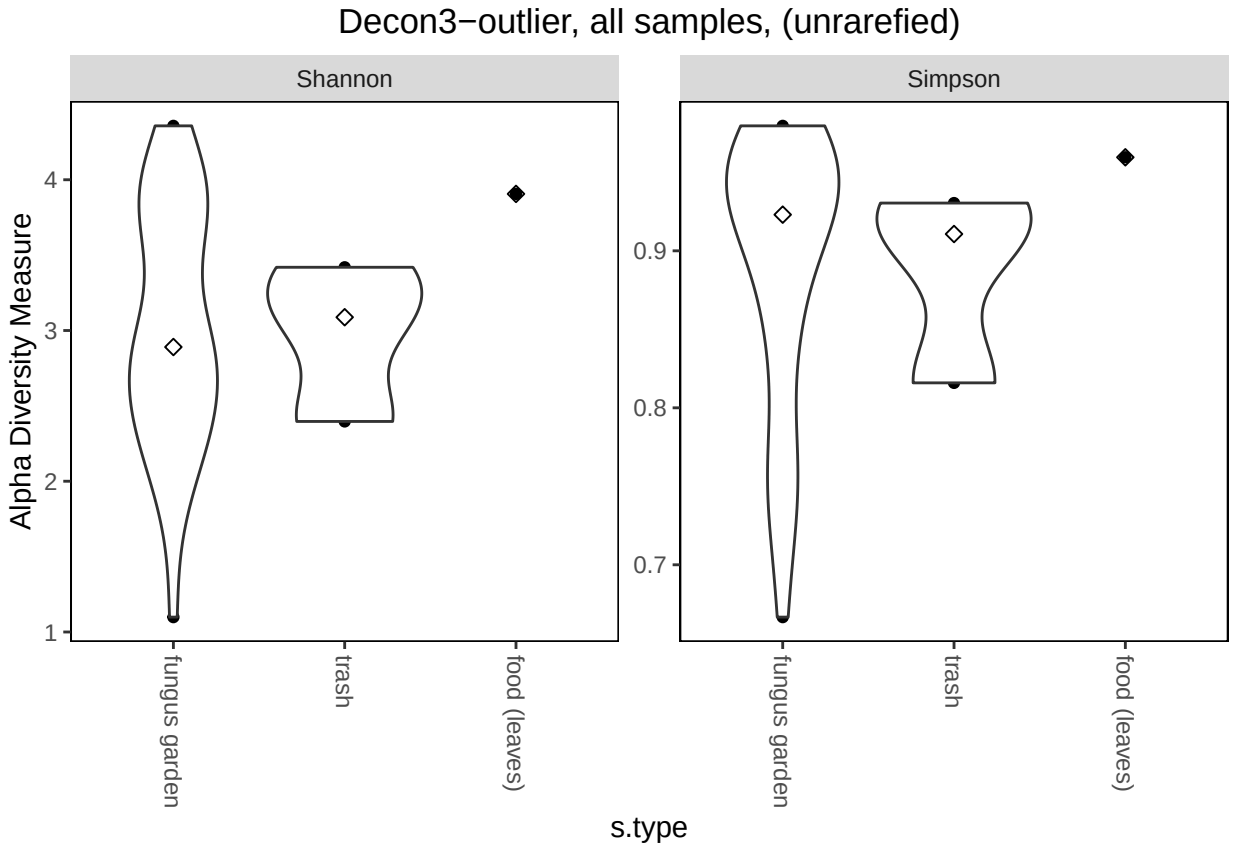
Removing outlier top, fungus garden sample. Determined to be outlier with Bray Curtis NMDS

```
ps = subset_samples(psList$decon3, s.id != "d3e21")
```

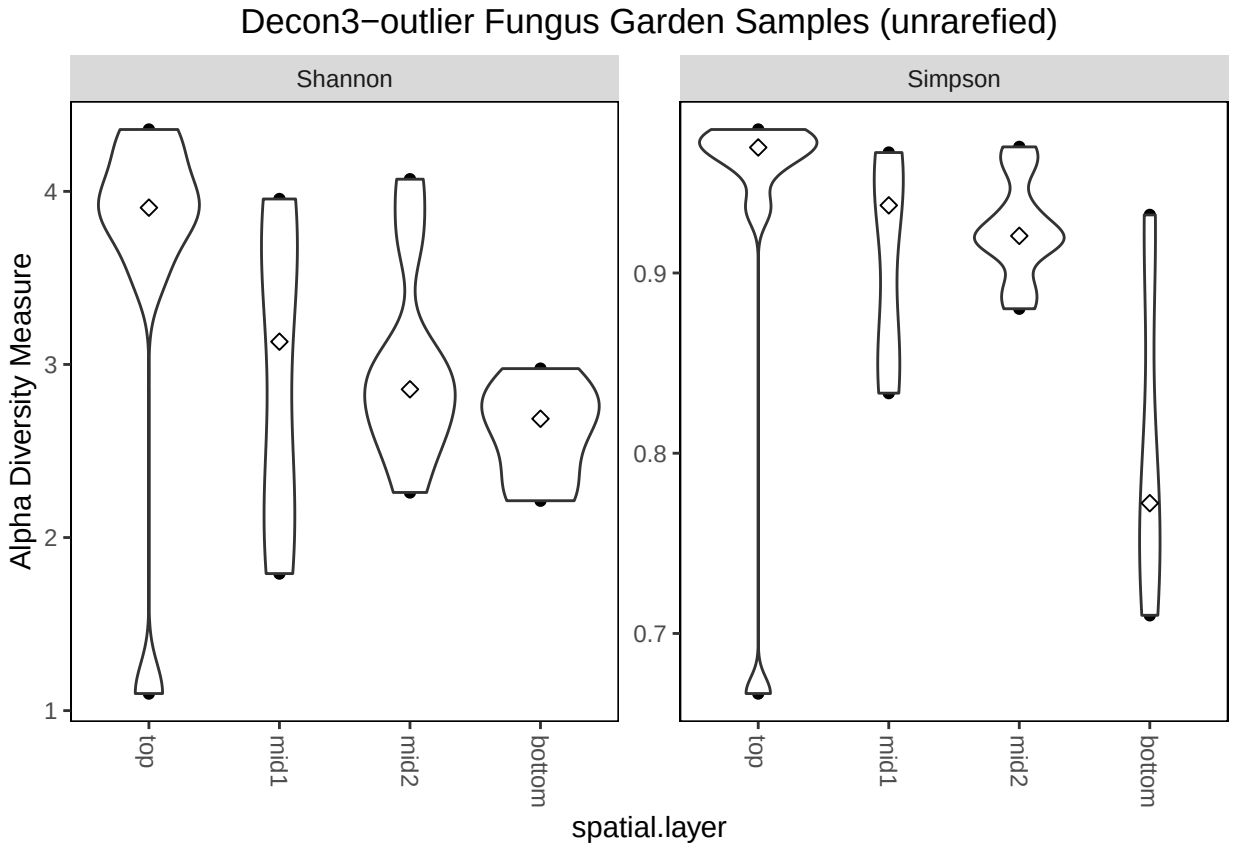
Violin Plot

```
p <-
  ps %>%
  plot_richness(., x = "s.type",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_violin(
  ) +
  ggtitle("Decon3-outlier, all samples, (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
```

```
## 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 = "spatial.layer",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_violin(
  ) +
  ggtitle("Decon3-outlier Fungus Garden Samples (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$data$spatial.layer = factor(p$data$spatial.layer, levels=c("top","mid1","mid2","bottom"))
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
ps.adiv.df = ps.adiv.df %>% filter(s.type != "food (leaves)") # removing bc only 1 samples
s.type.aov = list( shan = anova(aov(Shannon ~ s.type, ps.adiv.df)),
                   simp = 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     1  0.0079 0.00793  0.0129 0.9102
## Residuals 36 22.1254 0.61459
##
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##           Df   Sum Sq   Mean Sq F value Pr(>F)
## s.type     1 0.000069 0.0000691  0.0092  0.924
```

```
## Residuals 36 0.269526 0.0074868
```

```
#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)),
                 simp = 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  4.3543  1.4514  2.6119 0.06893 .
## Residuals     31 17.2266  0.5557
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##               Df   Sum Sq   Mean Sq F value   Pr(>F)
## spatial.layer  3 0.085113 0.0283712  4.9714 0.006236 **
## Residuals     31 0.176915 0.0057069
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

TukeyHSD

```
TukeyHSD(aov(Shannon ~ spatial.layer, ps.adiv.df.fg))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Shannon ~ spatial.layer, data = ps.adiv.df.fg)
##
## $spatial.layer
##                    diff         lwr       upr     p adj
## mid1-bottom 0.33906903 -0.61467940 1.2928175 0.7701204
## mid2-bottom 0.37864754 -0.57510089 1.3323960 0.7055564
## top-bottom  1.00029925  0.01719787 1.9834006 0.0449288
## mid2-mid1   0.03957851 -0.91416992 0.9933269 0.9994773
## top-mid1    0.66123022 -0.32187115 1.6443316 0.2809719
## top-mid2    0.62165172 -0.36144966 1.6047531 0.3327169
```

```
TukeyHSD(aov(Simpson ~ spatial.layer, ps.adiv.df.fg))
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Simpson ~ spatial.layer, data = ps.adiv.df.fg)
##
## $spatial.layer
##                     diff          lwr       upr     p adj
## mid1-bottom 0.098724629  0.002071466 0.1953778 0.0438524
```

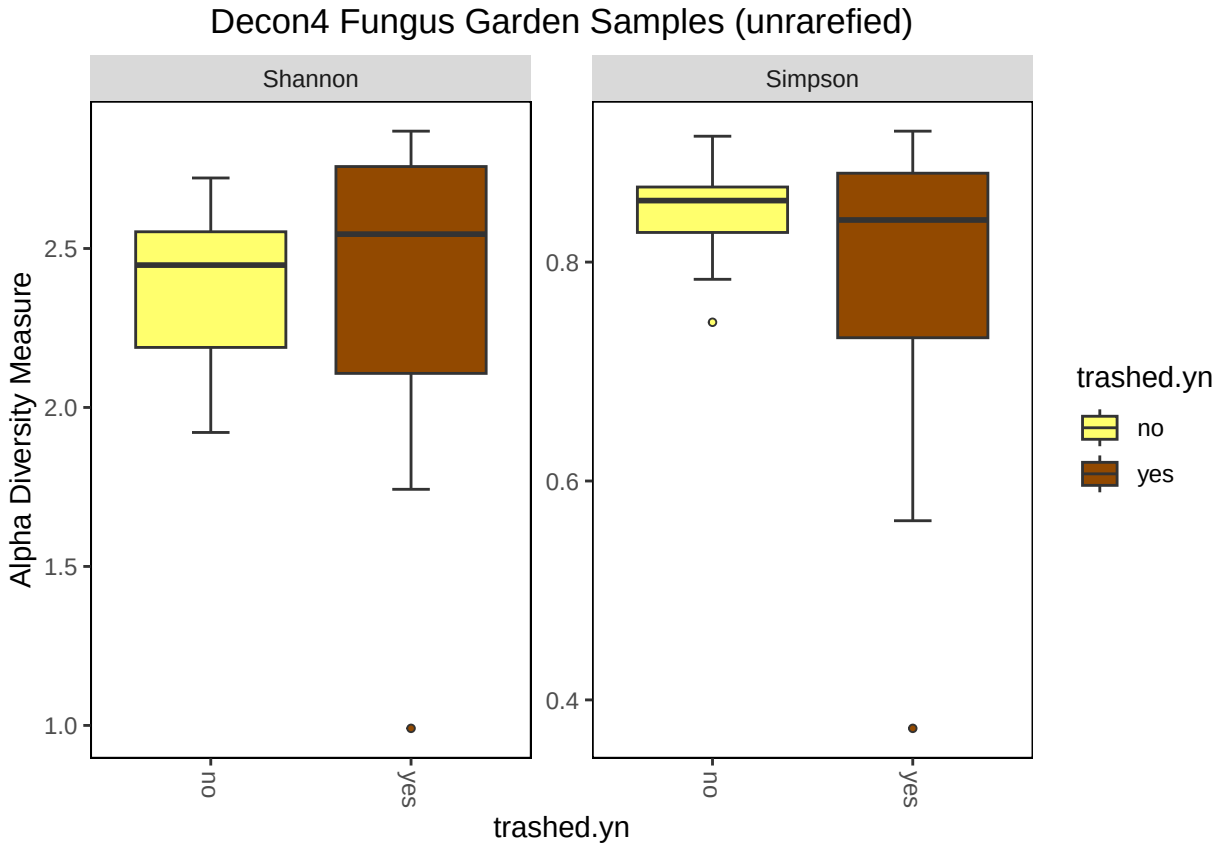
```
## mid2-bottom 0.115516744  0.018863581 0.2121699 0.0142306
## top-bottom  0.120707156  0.021079356 0.2203350 0.0127312
## mid2-mid1   0.016792114 -0.079861049 0.1134453 0.9647691
## top-mid1    0.021982526 -0.077645274 0.1216103 0.9316368
## top-mid2    0.005190412 -0.094437388 0.1048182 0.9989688
```

Decon4

```
ps = psList$decon4
trashed.yn.col.ls = list( "yes" ="#924900", "no" = "#ffff6d" )
# full.inoc.dist.cols
```

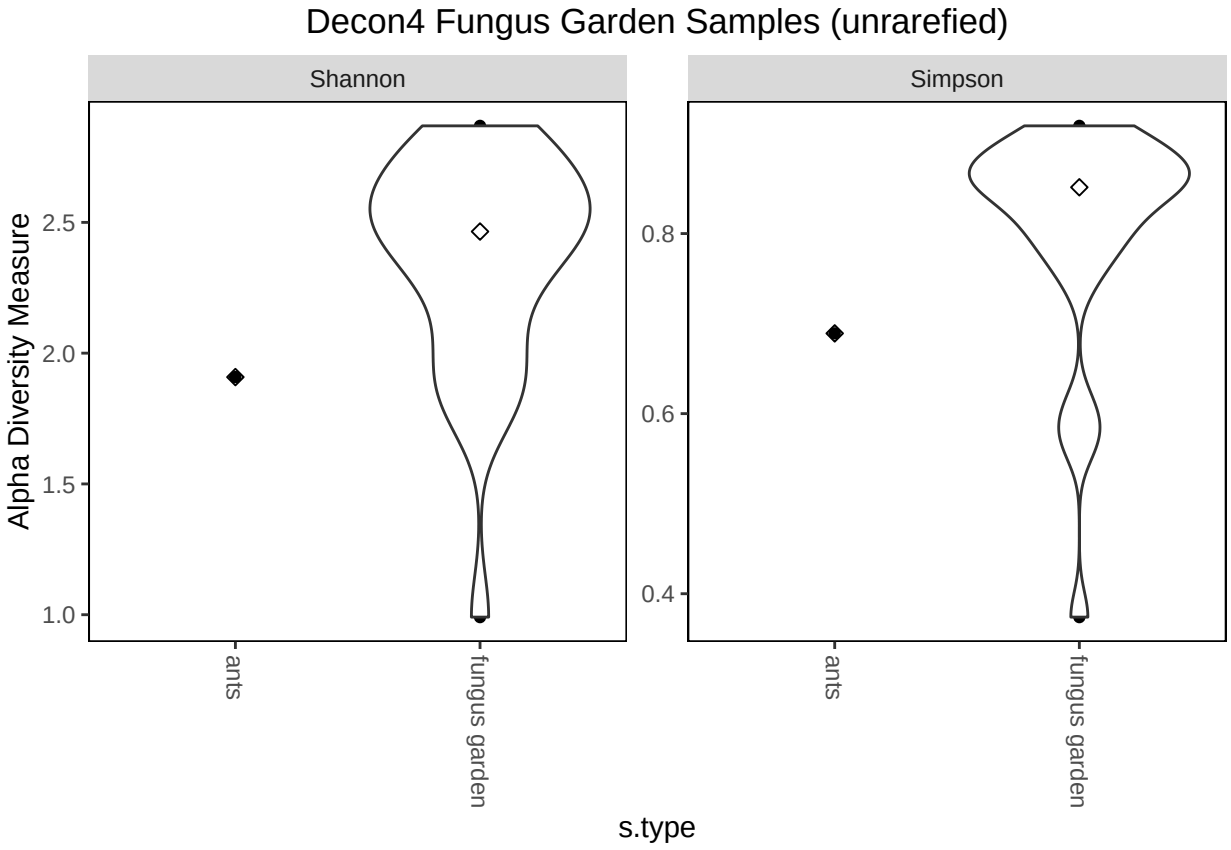
Violin plot

```
# Alpha Diversity violin plot
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 Fungus Garden Samples (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
```




```
p <-
  ps %>%
  plot_richness(., x = "s.type",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_violin(
  ) +
  ggtitle("Decon4 Fungus Garden Samples (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
```

```
## 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.
```



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$trashed.yn = sample_data(ps)$trashed.yn
ps.adiv.df$inoc.dist.cm = sample_data(ps)$inoc.dist.cm

### type doesnt make sense bc all FG except 1 ant samples
#Run anova on both Shannon and Simpson for s.type
# s.type.aov = list(shan = anova(aov(Shannon ~ s.type, ps.adiv.df)),
#                   simp = anova(aov(Simpson ~ s.type, ps.adiv.df)))
# s.type.aov
#Run anova on both Shannon and Simpson for spatial.layer
ps.adiv.df.fg = ps.adiv.df %>% filter(s.type == "fungus garden")
trash.yn.aov = list(shan = anova(aov(Shannon ~ trashed.yn, ps.adiv.df.fg)),
                    simp = anova(aov(Simpson ~ trashed.yn, ps.adiv.df.fg)))
trash.yn.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##             Df Sum Sq  Mean Sq F value Pr(>F)
## trashed.yn   1 0.0012 0.001192  0.0064 0.9366
## Residuals   26 4.8071 0.184888
##
```

```
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##            Df  Sum Sq  Mean Sq F value Pr(>F)
## trashed.yn  1 0.03360 0.033603  2.1825 0.1516
## Residuals  26 0.40032 0.015397
```

```
# Run anova on both Shannon and Simpson for inoc.dist.cm
inoc.dist.cm.aov = list(shan = anova(aov(Shannon ~ inoc.dist.cm, ps.adiv.df.fg)),
                        simp = 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 0.3737 0.37366  2.1908 0.1509
## Residuals    26 4.4346 0.17056
##
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##              Df  Sum Sq   Mean Sq F value Pr(>F)
## inoc.dist.cm  1 0.00021 0.0002135  0.0128 0.9108
## Residuals    26 0.43371 0.0166810
```

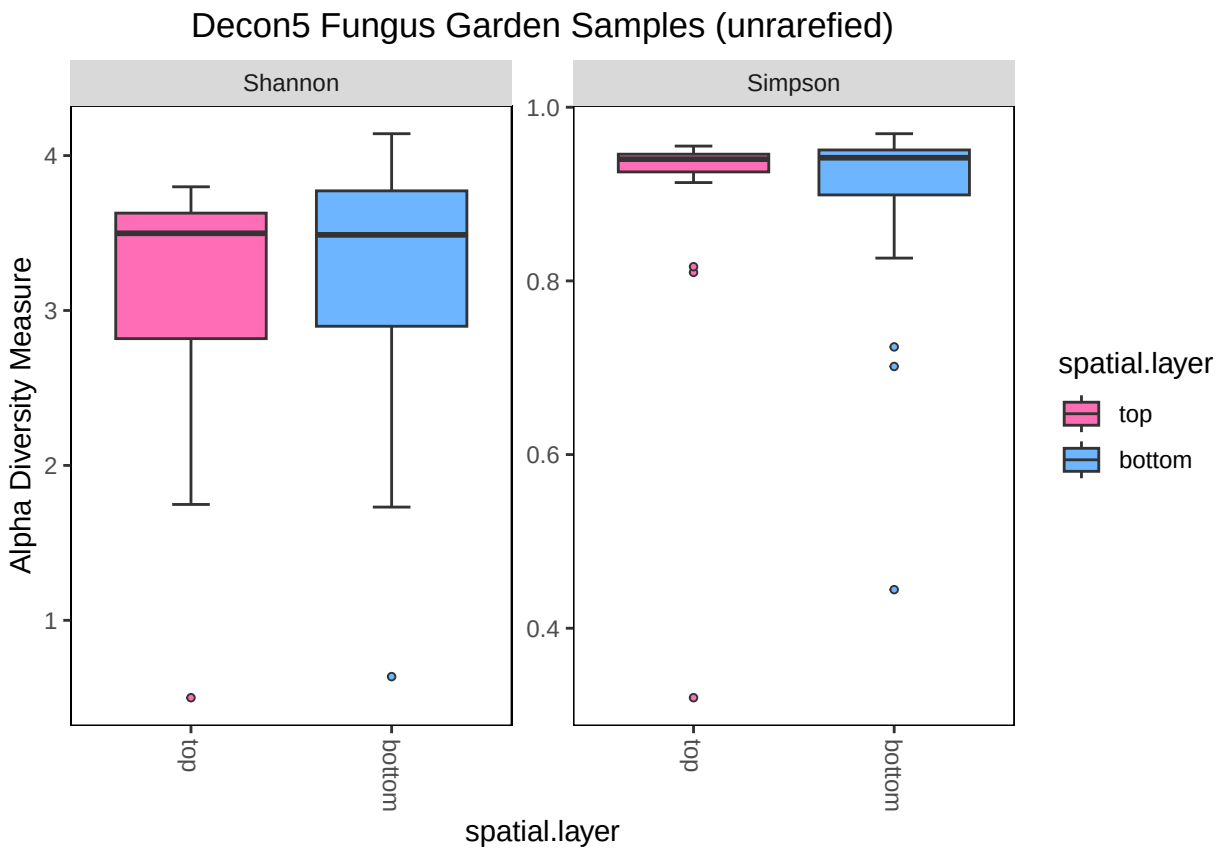
Decon5

```
ps = psList$decon5
spatial.layer.col.ls =
  list("top"="#ff6db6", "mid1"="#ffb6db","mid2"="#b6dbff","bottom"="#6db6ff")
```

Violin plot

```
# Alpha Diversity violin plot
## spatial.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)
```

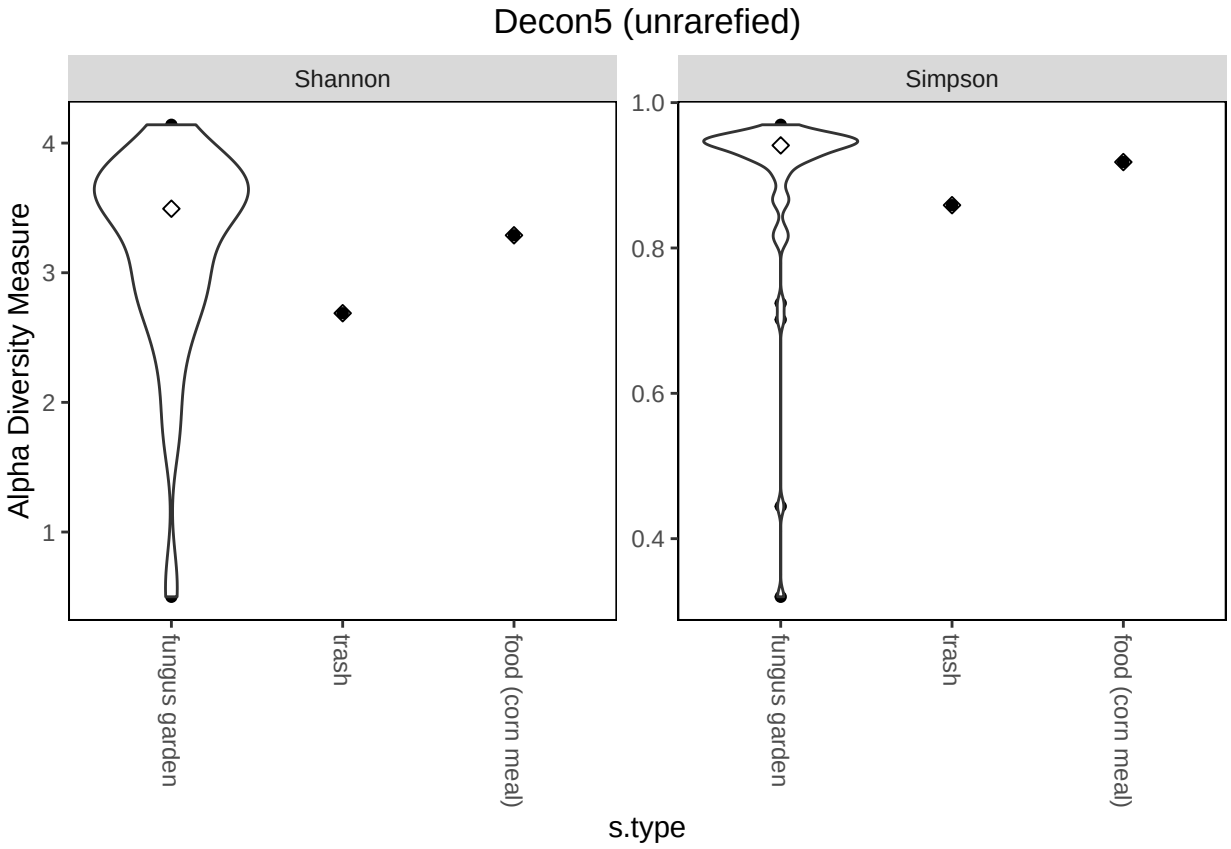
```
  ) +
  ggtitle("Decon5 Fungus Garden Samples (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

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




```
## s.type
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 (corn meal)"))
p
```

```
## 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.
```



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$layer = sample_data(ps)$spatial.layer
```

```r
#Run anova on both Shannon and Simpson for s.type
s.type.aov = list(shan = anova(aov(Shannon ~ s.type, ps.adiv.df)),
                  simp = 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  0.2439 0.12197  0.1972 0.8217
## Residuals 49 30.3035 0.61844
##
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##           Df  Sum Sq   Mean Sq F value Pr(>F)
## s.type     2 0.00192 0.0009593  0.0658 0.9364
## Residuals 49 0.71435 0.0145786
```

```r
#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 ~ layer, ps.adiv.df.fg)),
                 simp = anova(aov(Simpson ~ layer, ps.adiv.df.fg)))
layer.aov
```

```
## $shan
## Analysis of Variance Table
##
## Response: Shannon
##           Df  Sum Sq Mean Sq F value Pr(>F)
## layer      1  0.0981 0.09815   0.156 0.6946
## Residuals 48 30.2053 0.62928
##
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##           Df  Sum Sq   Mean Sq F value Pr(>F)
## layer      1 0.00015 0.0001469  0.0099 0.9213
## Residuals 48 0.71421 0.0148793
```

TSinf

```r
ps = psList$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")
  )
```

What are my sample variables?

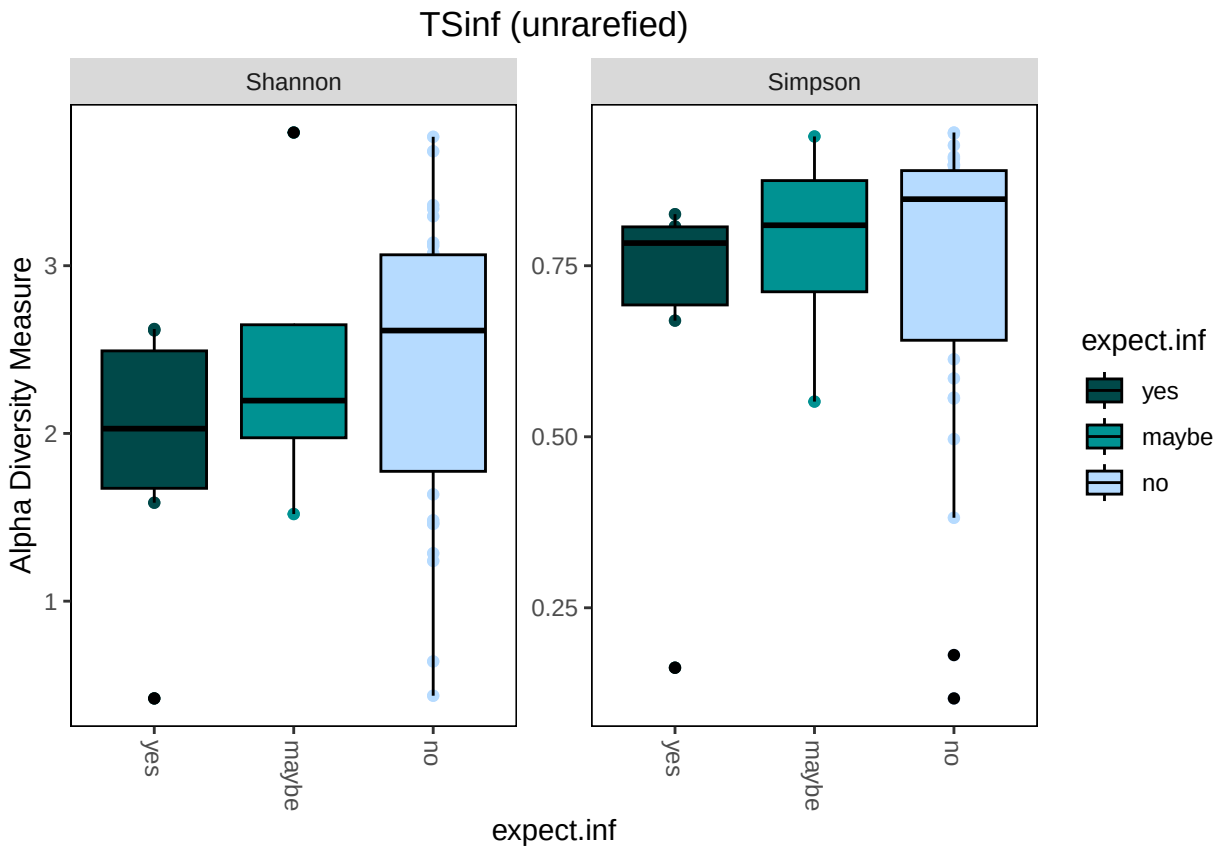
```
sample_variables(ps)
```

```
##  [1] "seq.id"            "s.id"              "s.name"
##  [4] "s.type"            "treatment"         "ants.yn"
##  [7] "exp.rep"           "day.post.inf"      "expect.inf"
## [10] "extract.batch"     "pcr.plate"         "pcr.batch"
## [13] "dil.factor"        "Sample_or_Control" "host.species"
## [16] "inoc.dist"
```

Violin Plot

```
exp.inf.col.ls = list("yes" = "#004949",
                      "maybe" = "#009292",
                      "no"= "#b6dbff")
## expect.inf
p <-
  ps %>%
  plot_richness(., x = "expect.inf",
                measures = c("Shannon", "Simpson"),
                color = "expect.inf"
  ) +
  # geom_violin( aes(fill = expect.inf)
  # ) +
  geom_point(
    alpha = 0.7
  ) +
  geom_boxplot(aes(fill=expect.inf), color="black"
  ) +
  ggtitle("TSinf (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

  ) +
```

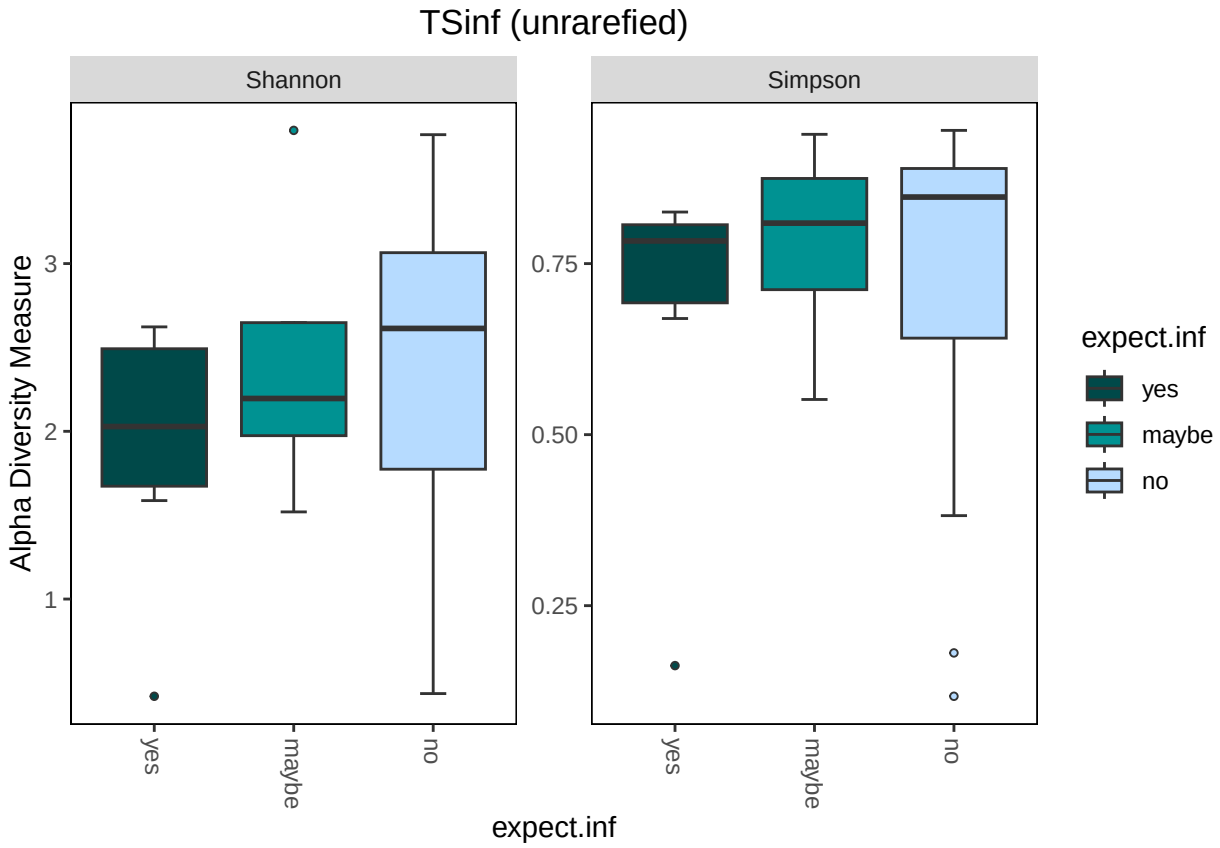
```
  scale_fill_manual(values = unlist(exp.inf.col.ls)) +
  scale_color_manual(values = unlist(exp.inf.col.ls))
# reorder
p$data$expect.inf = factor(p$data$expect.inf, levels=c("yes","maybe","no"))
# plot
p
```



Boxplot alpha diversity #### expect.inf

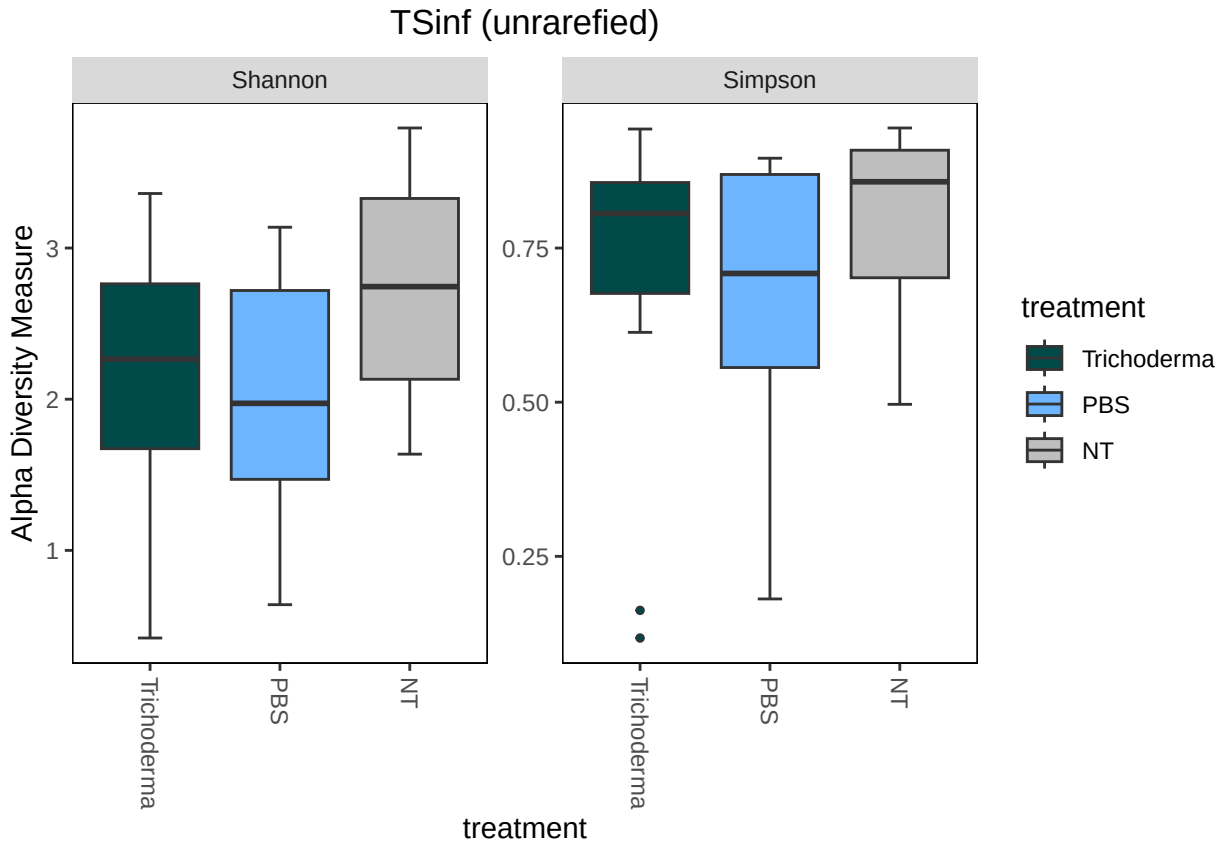
```
p <-
  ps %>%
  plot_richness(., x = "expect.inf",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_boxplot(
    aes(fill = expect.inf),
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25

  ) +
  scale_fill_manual(
    values = unlist(exp.inf.col.ls)
  ) +
  ggtitle("TSinf (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
```




```
treatment.col.ls = list("Trichoderma" = "#004949",
                        "PBS" = "#6db6ff",
                        "NT" ="grey")
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 (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
```

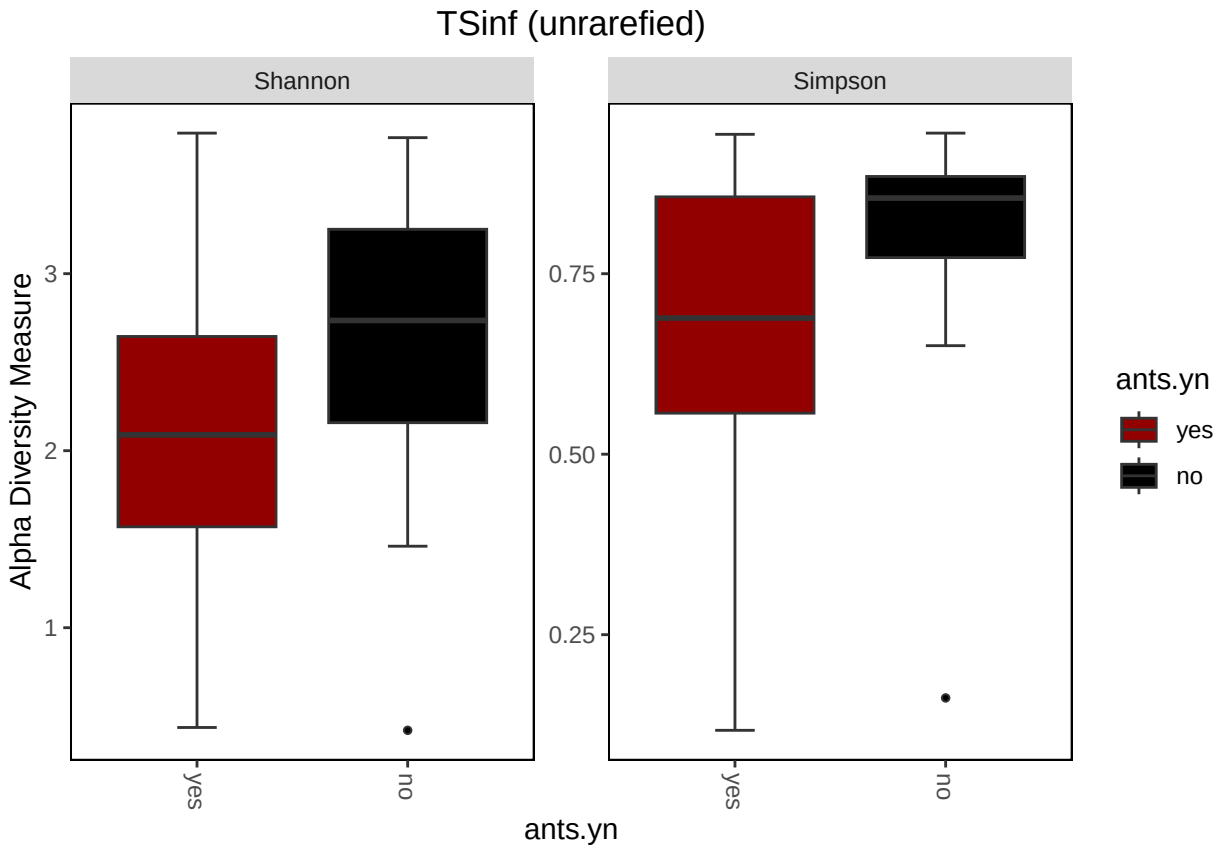


treatment

+/- ants

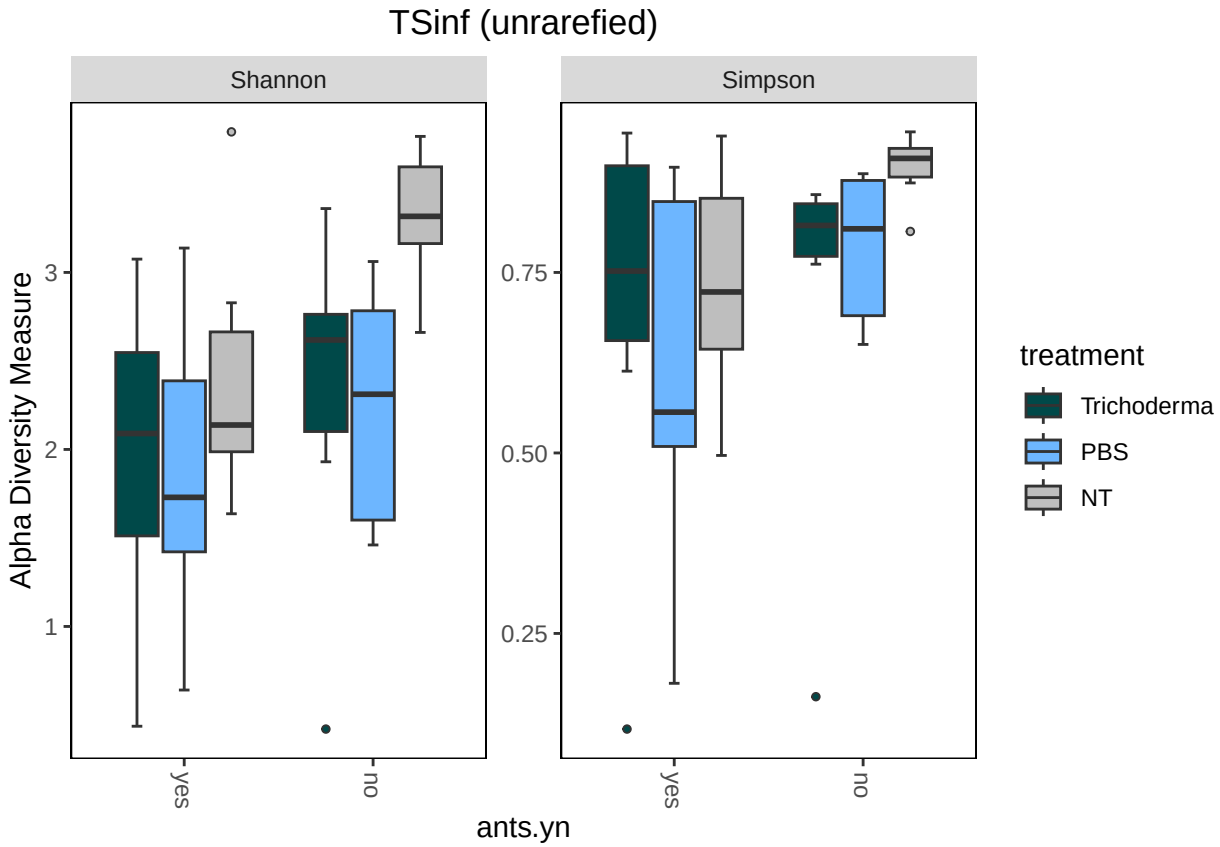
```
ants.yn.col.ls = list("yes" = "#920000", "no" = "black")
p <-
  ps %>%
  plot_richness(., x = "ants.yn",
                measures = c("Shannon", "Simpson"),
  ) +
  geom_boxplot(
    aes(fill = ants.yn),
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25

  ) +
```

```
  scale_fill_manual(
    values = unlist(ants.yn.col.ls)
  ) +
  ggtitle("TSinf (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.yn.col.ls
# treatment.col.ls

p <-
  ps %>%
  plot_richness(., x = "ants.yn",
                measures = c("Shannon", "Simpson")
  ) +
  geom_boxplot(
    aes(fill = treatment),
```

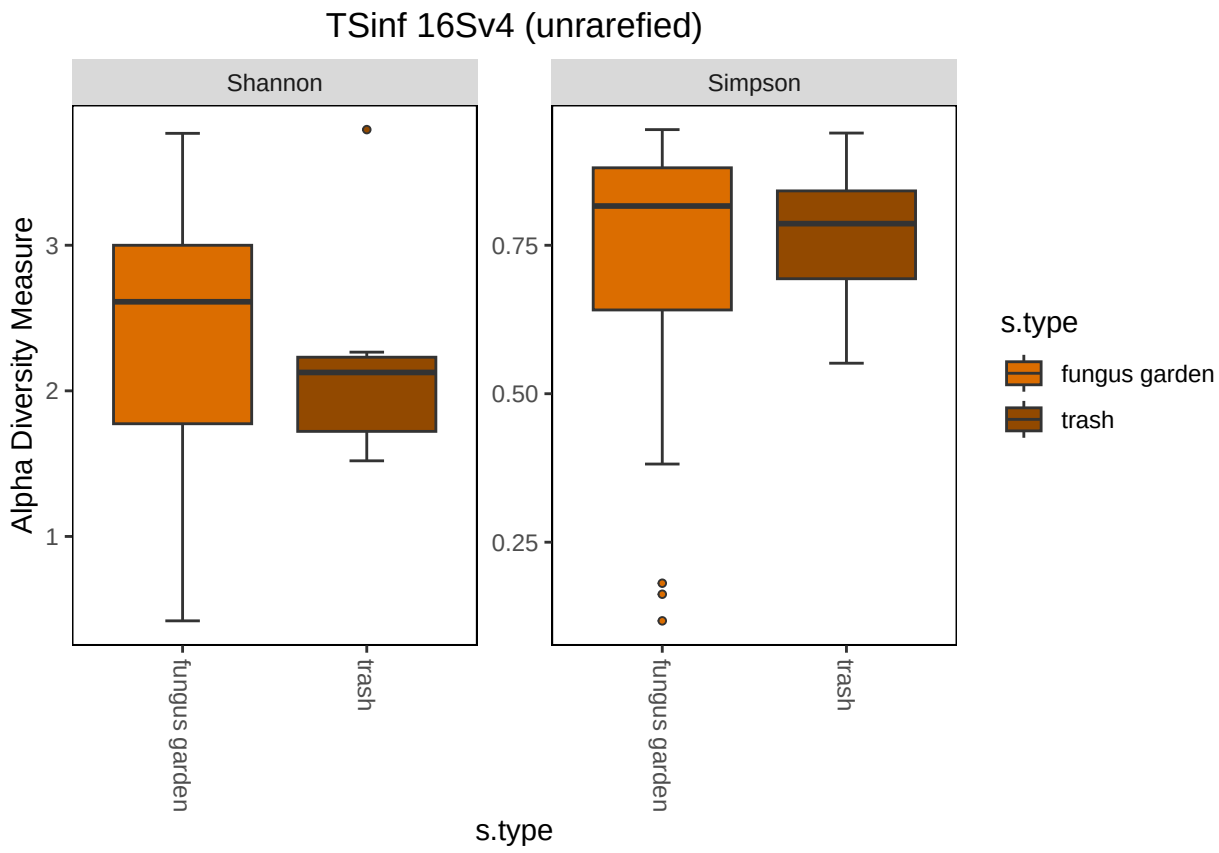
```
    outlier.shape = 21,
    outlier.size = 1,
    staplewidth = 0.25

  ) +
  scale_fill_manual(
    values = unlist(treatment.col.ls)
  ) +
  ggtitle("TSinf (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

```
s.type.col.ls = list("fungus garden" = "#db6d00",
                     "trash" = "#924900" )
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 16Sv4 (unrarefied)"
  ) +
  theme(
    panel.background = element_blank(),
    panel.border = element_rect(fill = NA),
    plot.title = element_text(hjust = 0.5)

  )
p$data$s.type = factor(p$data$s.type, levels=c("fungus garden","trash"))
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$treatment = sample_data(ps)$treatment
ps.adiv.df$ants.yn = sample_data(ps)$ants.yn
ps.adiv.df$day.post.inoc = sample_data(ps)$day.post.inoc
ps.adiv.df$expect.inf = sample_data(ps)$expect.inf

#Run anova on both Shannon and Simpson for s.type
s.type.aov = list(shan = anova(aov(Shannon ~ s.type, ps.adiv.df)),
                  simp = 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     1  0.0375 0.03753  0.0494 0.8252
## Residuals 40 30.3758 0.75939
##
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##           Df  Sum Sq  Mean Sq F value Pr(>F)
## s.type     1 0.00837 0.008365  0.1814 0.6725
## Residuals 40 1.84508 0.046127
```

```
#Run anova on both Shannon and Simpson for expect.inf
expect.inf.aov = list(shan = anova(aov(Shannon ~ expect.inf, ps.adiv.df)),
                      simp = 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  1.2806 0.64028  0.8571 0.4322
## Residuals  39 29.1327 0.74699
##
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##            Df  Sum Sq  Mean Sq F value Pr(>F)
## expect.inf  2 0.02987 0.014935  0.3194 0.7285
## Residuals  39 1.82358 0.046758
```

```
#Run anova on both Shannon and Simpson for treatment
treatment.aov = list(shan = anova(aov(Shannon ~ treatment, ps.adiv.df)),
                     simp = 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  4.6363 2.31817  3.5073 0.03974 *
## Residuals 39 25.7770 0.66095
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##           Df  Sum Sq  Mean Sq F value Pr(>F)
## treatment  2 0.11137 0.055684  1.2466 0.2987
## Residuals 39 1.74208 0.044669
```

```
#Run anova on both Shannon and Simpson for ants.yn
ants.yn.aov = list(shan = anova(aov(Shannon ~ ants.yn, ps.adiv.df)),
                   simp = 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  2.9407 2.94072  4.2817 0.04503 *
## Residuals 40 27.4726 0.68681
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##           Df  Sum Sq  Mean Sq F value  Pr(>F)
## ants.yn    1 0.14081 0.140808  3.2887 0.07727 .
## Residuals 40 1.71264 0.042816
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#Run anova on both Shannon and Simpson for treatment*ants
trtxants.aov = list( shan = anova(aov(Shannon ~ treatment * ants.yn, ps.adiv.df)),
                     simp = 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  4.6363 2.31817  3.7926 0.03201 *
```

```
## ants.yn            1  2.9407 2.94072  4.8112 0.03482 *
## treatment:ants.yn  2  0.8320 0.41601  0.6806 0.51270
## Residuals         36 22.0042 0.61123
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## $simp
## Analysis of Variance Table
##
## Response: Simpson
##                   Df  Sum Sq  Mean Sq F value  Pr(>F)
## treatment          2 0.11137 0.055684  1.3047 0.28377
## ants.yn            1 0.14081 0.140808  3.2992 0.07765 .
## treatment:ants.yn  2 0.06483 0.032416  0.7595 0.47524
## Residuals         36 1.53644 0.042679
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

TukeyHSD

```
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          -0.74597308 -1.4946022 0.002656075 0.0509853
## Trichoderma-NT  -0.65475566 -1.4033848 0.093873488 0.0966788
## Trichoderma-PBS  0.09121741 -0.6574117 0.839846565 0.9526435
```

```
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 -0.5346996 -1.056958 -0.01244164 0.0450284
```

```
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 -0.117003 -0.2474002 0.01339422 0.0772667
```

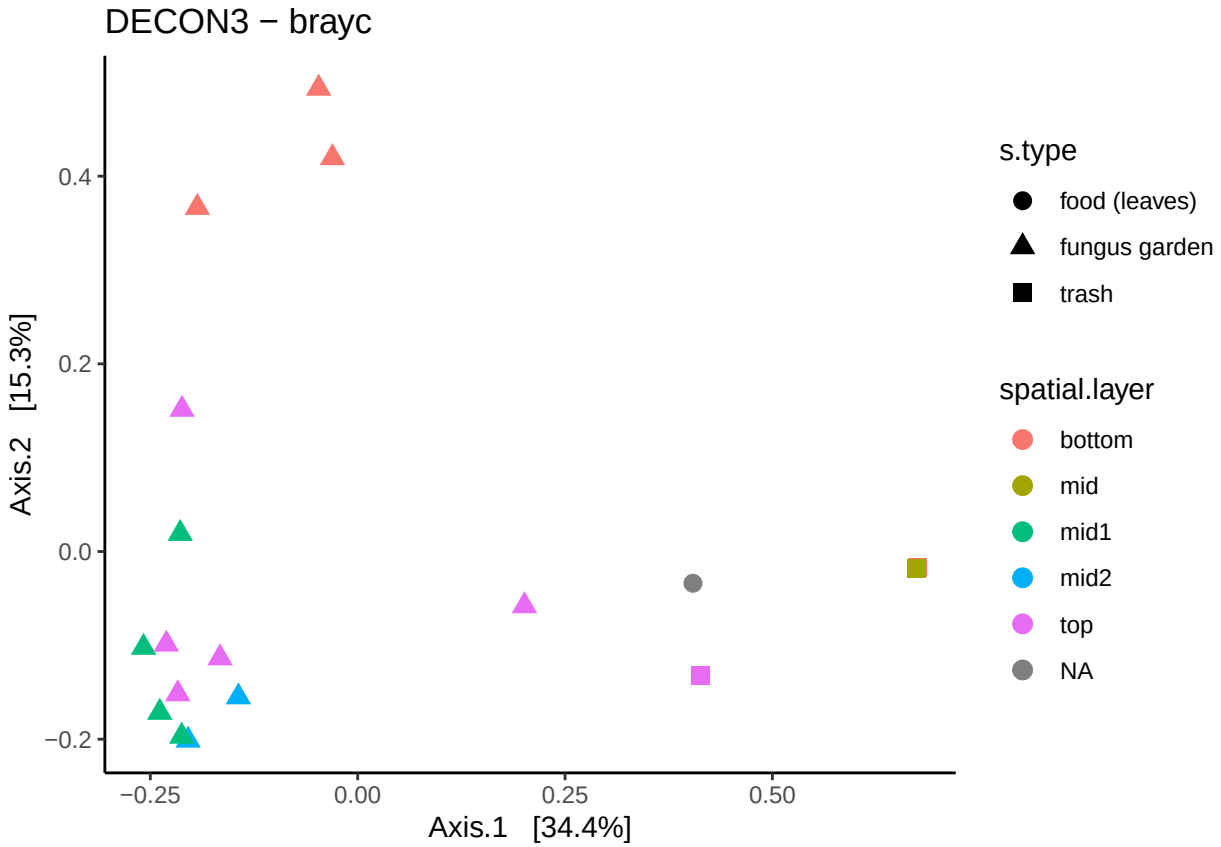
Beta Diversity

Decon3

```
ps = psList.raref.rab$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

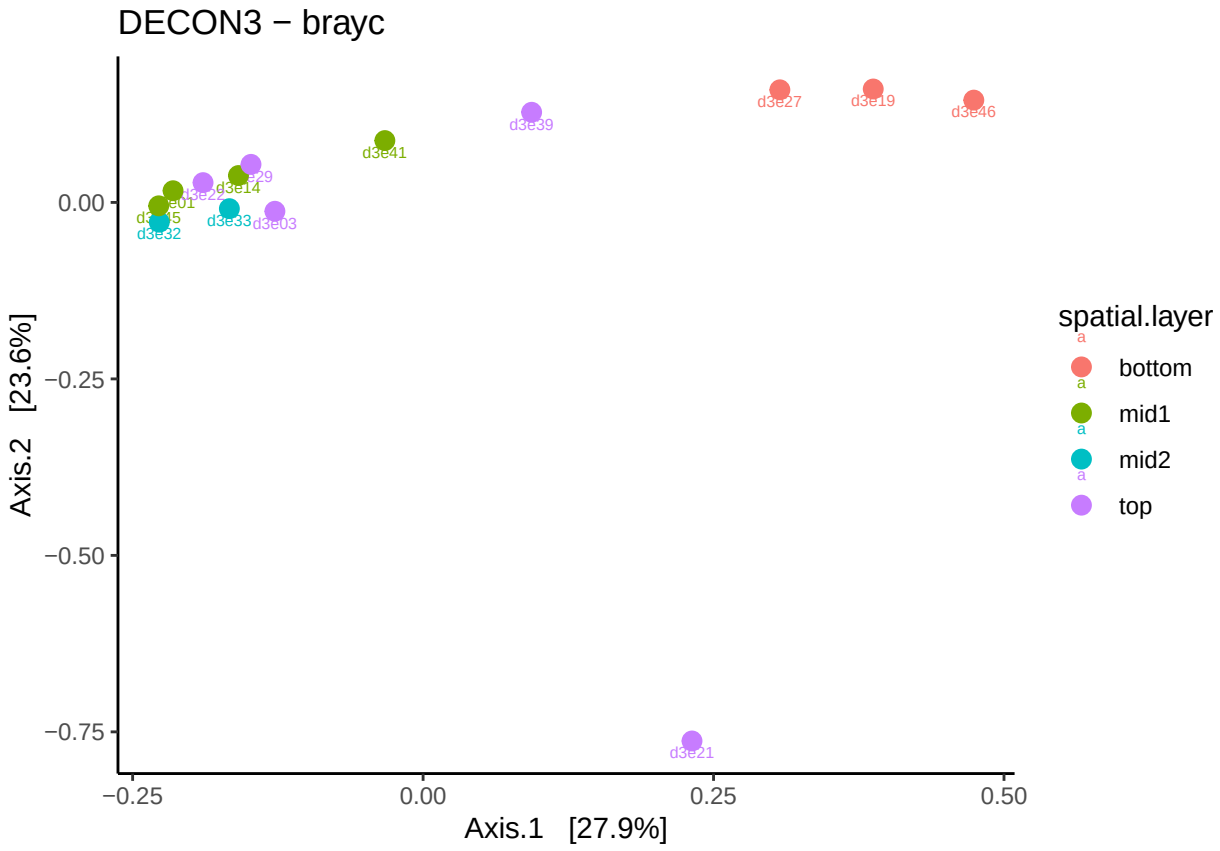
```
# Bray Curtis NMDS
ps.bcord = ordinate(ps, method= "PCoA", distance = "bray")

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



PCoA

```
## fungus garden samples only
# Bray Curtis NMDS
```

```
ps.fg.bcord = ordinate(ps.fg, method= "PCoA", distance = "bray")
plot_ordination( ps.fg,
                 ps.fg.bcord,
                 color = "spatial.layer",
                 label = "s.id") +
  geom_point(size=3) +
  ggtitle("DECON3 - brayc") +
  theme_classic()
```



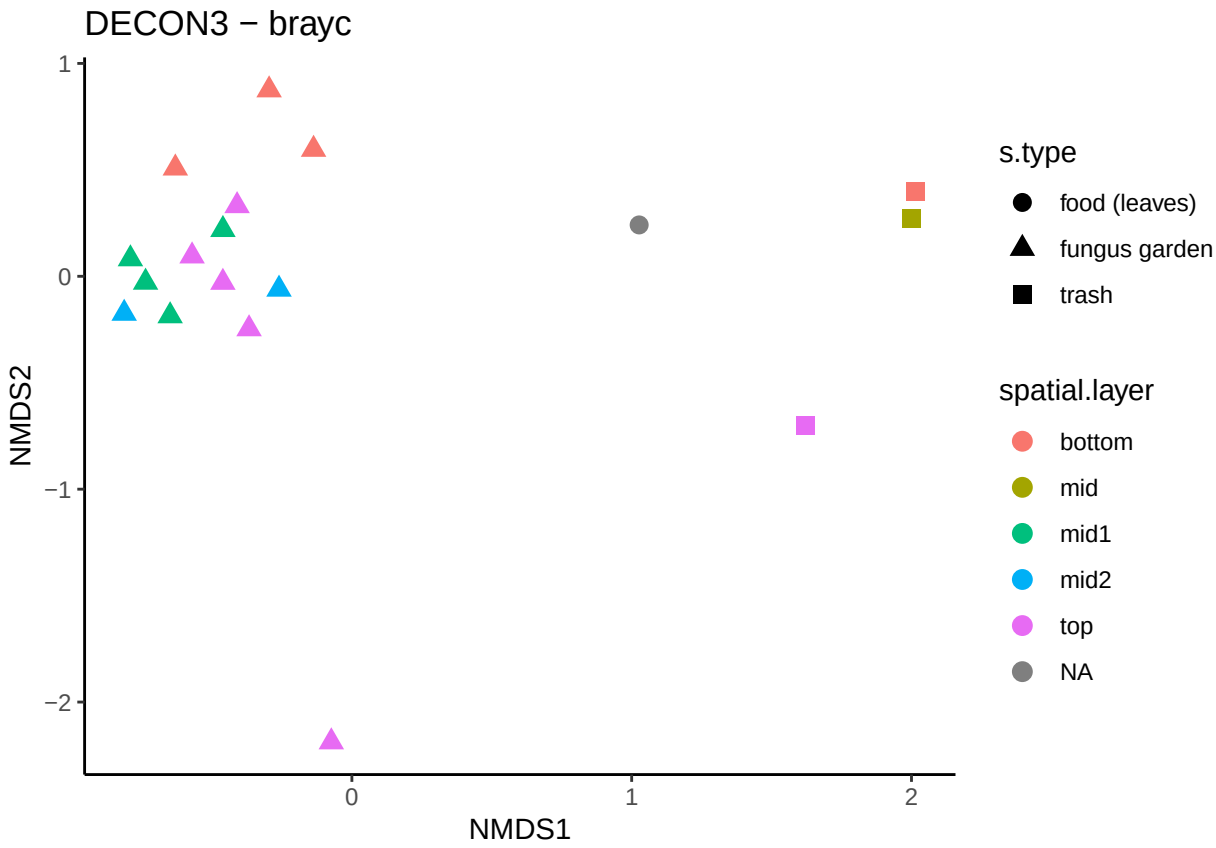

```
# Bray Curtis NMDS
ps.bcord = ordinate(ps, method= "NMDS", distance = "bray")
```

NMDS

```
## Run 0 stress 0.141649
## Run 1 stress 0.09416483
## ... New best solution
## ... Procrustes: rmse 0.1073215  max resid 0.3025957
## Run 2 stress 0.1090831
## Run 3 stress 0.09416503
## ... Procrustes: rmse 0.0002371385  max resid 0.0007387569
## ... Similar to previous best
## Run 4 stress 0.09416493
## ... Procrustes: rmse 0.0001848909  max resid 0.0005768381
## ... Similar to previous best
## Run 5 stress 0.367608
```

```
## Run 6 stress 0.1115804
## Run 7 stress 0.1055738
## Run 8 stress 0.1050734
## Run 9 stress 0.09416487
## ... Procrustes: rmse 8.873332e-05  max resid 0.0002731305
## ... Similar to previous best
## Run 10 stress 0.1050734
## Run 11 stress 0.1055735
## Run 12 stress 0.09416486
## ... Procrustes: rmse 8.235494e-05  max resid 0.0002494118
## ... Similar to previous best
## Run 13 stress 0.1055735
## Run 14 stress 0.1034275
## Run 15 stress 0.1115808
## Run 16 stress 0.09416486
## ... Procrustes: rmse 0.0001064639  max resid 0.0003317263
## ... Similar to previous best
## Run 17 stress 0.1055736
## Run 18 stress 0.09416485
## ... Procrustes: rmse 6.680275e-05  max resid 0.000206804
## ... Similar to previous best
## Run 19 stress 0.09475299
## Run 20 stress 0.1055735
## *** Best solution repeated 6 times
```

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



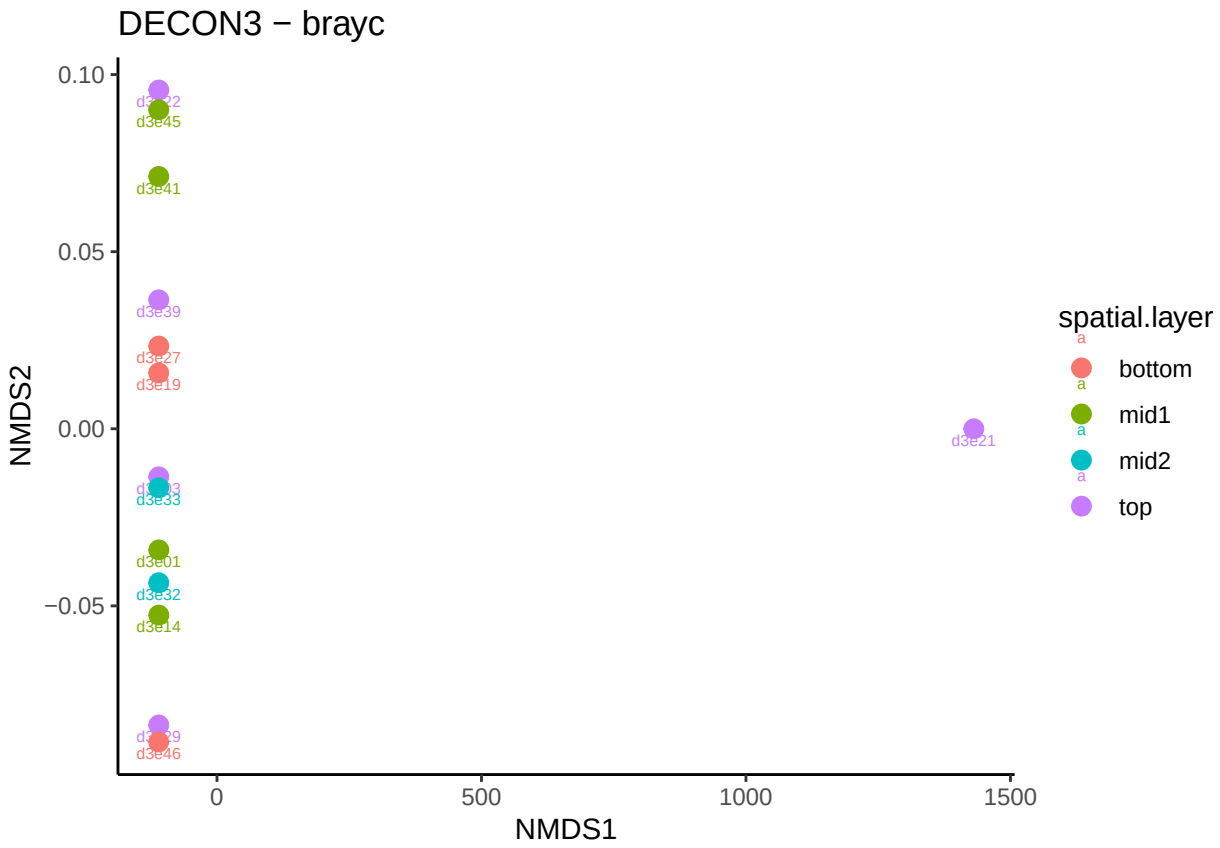

```
## fungus garden samples only
# Bray Curtis NMDS
ps.fg.bcord = ordinate(ps.fg, method= "NMDS", distance = "bray")
```

```
## Run 0 stress 9.128773e-05
## Run 1 stress 8.148177e-05
## ... New best solution
## ... Procrustes: rmse 0.000162893  max resid 0.0003292883
## ... Similar to previous best
## Run 2 stress 8.508513e-05
## ... Procrustes: rmse 8.051046e-05  max resid 0.0001908932
## ... Similar to previous best
## Run 3 stress 9.43356e-05
## ... Procrustes: rmse 9.271389e-05  max resid 0.0002404569
## ... Similar to previous best
## Run 4 stress 9.695444e-05
## ... Procrustes: rmse 0.0001725193  max resid 0.0003432615
## ... Similar to previous best
## Run 5 stress 9.813914e-05
## ... Procrustes: rmse 0.000131213  max resid 0.0002725705
## ... Similar to previous best
## Run 6 stress 9.938034e-05
## ... Procrustes: rmse 0.0001537593  max resid 0.0002825074
## ... Similar to previous best
## Run 7 stress 9.940427e-05
## ... Procrustes: rmse 0.0001542865  max resid 0.0002834464
## ... Similar to previous best
```

```
## Run 8 stress 9.727874e-05
## ... Procrustes: rmse 0.0001499763  max resid 0.0002776994
## ... Similar to previous best
## Run 9 stress 9.303722e-05
## ... Procrustes: rmse 0.0001200294  max resid 0.0002014531
## ... Similar to previous best
## Run 10 stress 7.733892e-05
## ... New best solution
## ... Procrustes: rmse 0.0001056701  max resid 0.0001930631
## ... Similar to previous best
## Run 11 stress 9.954421e-05
## ... Procrustes: rmse 0.0001278427  max resid 0.0001982807
## ... Similar to previous best
## Run 12 stress 9.25862e-05
## ... Procrustes: rmse 0.0001059626  max resid 0.0002647597
## ... Similar to previous best
## Run 13 stress 9.905111e-05
## ... Procrustes: rmse 0.0001077253  max resid 0.0001773519
## ... Similar to previous best
## Run 14 stress 9.38794e-05
## ... Procrustes: rmse 0.0001035237  max resid 0.0002393022
## ... Similar to previous best
## Run 15 stress 9.767595e-05
## ... Procrustes: rmse 0.0001202989  max resid 0.0002073931
## ... Similar to previous best
## Run 16 stress 9.419376e-05
## ... Procrustes: rmse 0.0001195877  max resid 0.0001902317
## ... Similar to previous best
## Run 17 stress 6.976945e-05
## ... New best solution
## ... Procrustes: rmse 0.0001018539  max resid 0.0001458934
## ... Similar to previous best
## Run 18 stress 9.465473e-05
## ... Procrustes: rmse 0.0001705402  max resid 0.0002790919
## ... Similar to previous best
## Run 19 stress 8.944021e-05
## ... Procrustes: rmse 0.0001588078  max resid 0.0002605427
## ... Similar to previous best
## Run 20 stress 8.229743e-05
## ... Procrustes: rmse 0.0001706974  max resid 0.0002630231
## ... Similar to previous best
## *** Best solution repeated 4 times
```

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

```
plot_ordination( ps.fg,
                 ps.fg.bcord,
                 color = "spatial.layer",
                 label = "s.id") +
  geom_point(size=3) +
  ggtitle("DECON3 - brayc") +
  theme_classic()
```




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

Permanova

```
## 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   1.4458 0.3018 6.4839  0.002 **
## Residual 15   3.3448 0.6982
## Total    16   4.7906 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
)
```

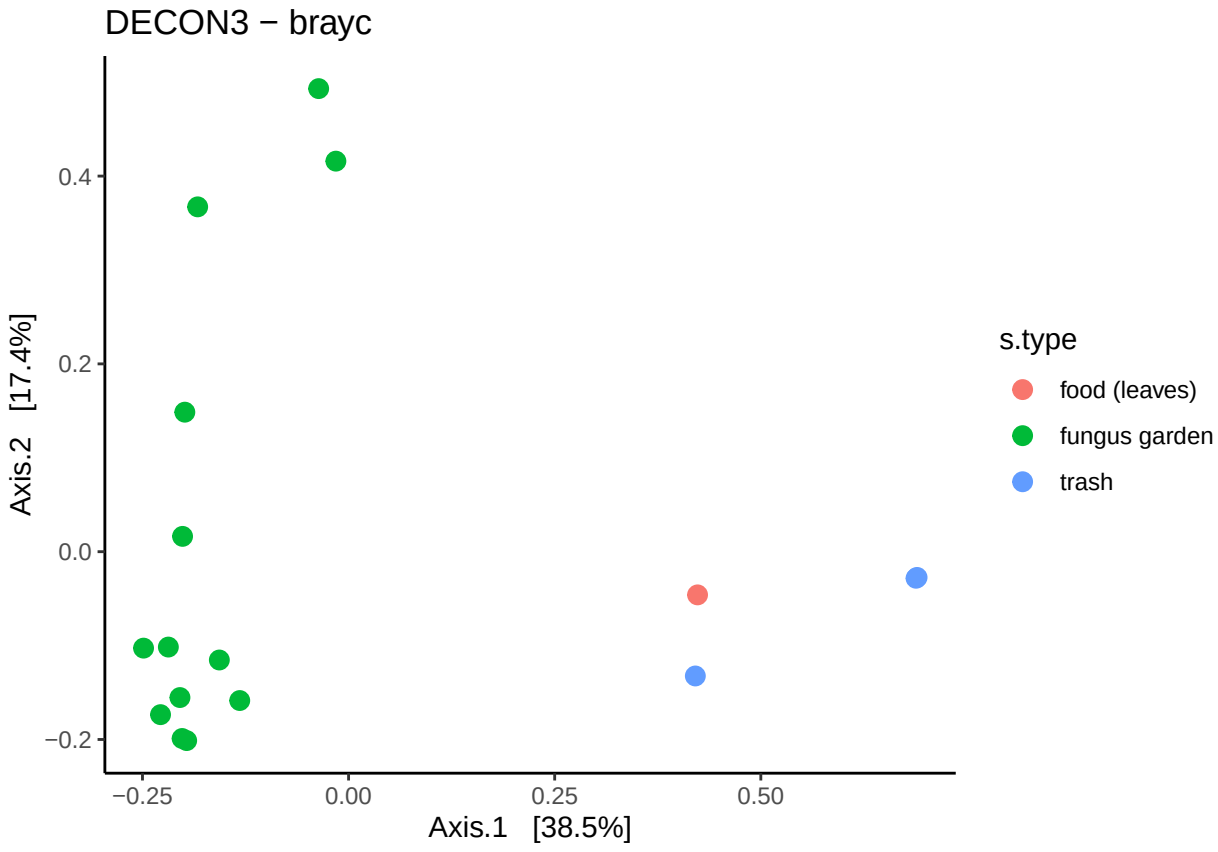
```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
```

```
##
## adonis2(formula = ps.fg.bcdist ~ spatial.layer, data = ps.fg.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     3   1.1160 0.38462 2.0834  0.003 **
## Residual 10   1.7855 0.61538
## Total    13   2.9015 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

remove outlier d3e21

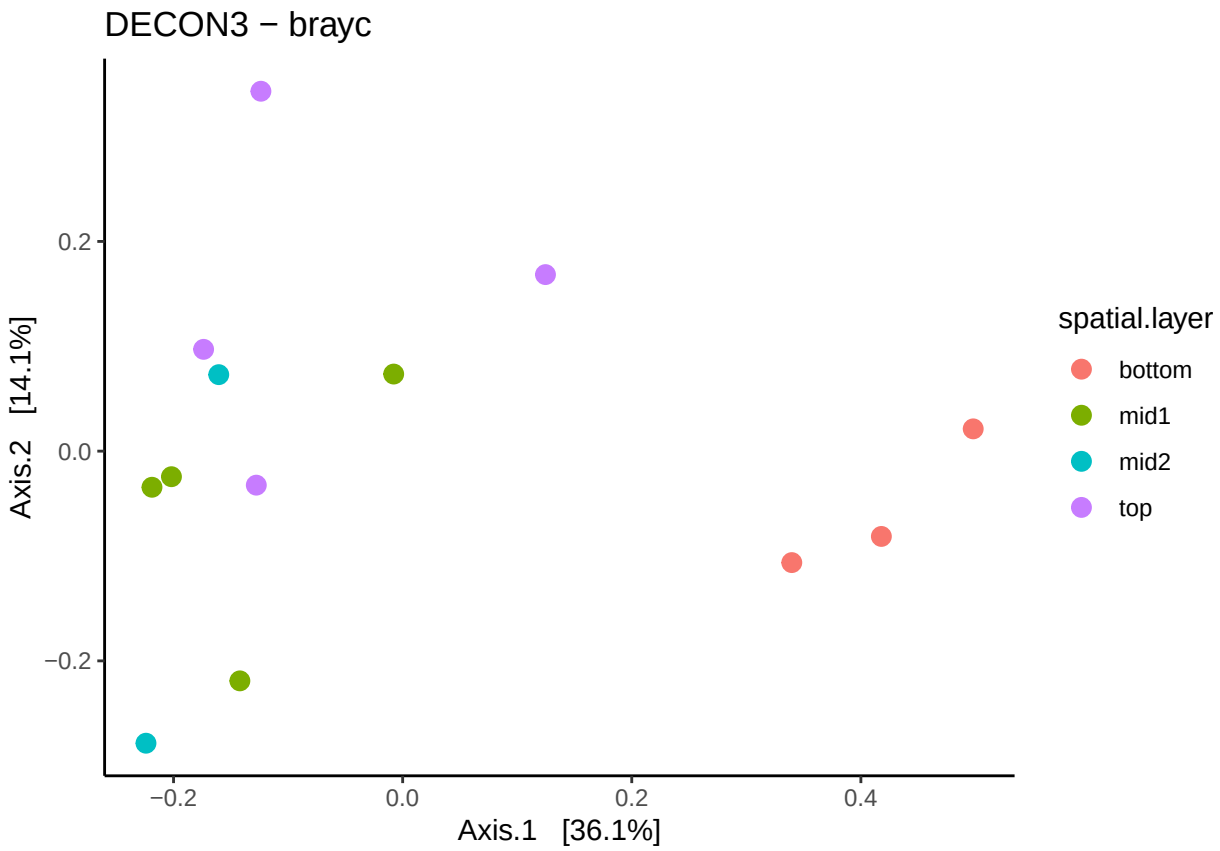
```
ps = subset_samples(ps, s.id != "d3e21")
ps.fg = subset_samples(ps.fg, s.id != "d3e21")
ps.samdat.df = data.frame(sample_data(ps))
ps.fg.samdat.df = data.frame(sample_data(ps.fg))
```

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

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



PCoA

```
## fungus garden samples only
# Bray Curtis PCoA
ps.fg.bcord = ordinate(ps.fg, method= "PCoA", distance = "bray")
plot_ordination( ps.fg,
                 ps.fg.bcord,
                 # label = "s.id",
                 color = "spatial.layer") +
  geom_point(size=3) +
  ggtitle("DECON3 - brayc") +
  theme_classic()
```




```
# Permanova
ps.bcdist = distance(subset_samples(ps, 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   1.5066 0.3617 7.9333  0.002 **
## Residual 14   2.6588 0.6383
## Total    15   4.1654 1.0000
## ---
## 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   1.0676 0.48191 2.7905  0.001 ***
## Residual  9   1.1478 0.51809
## Total    12   2.2154 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

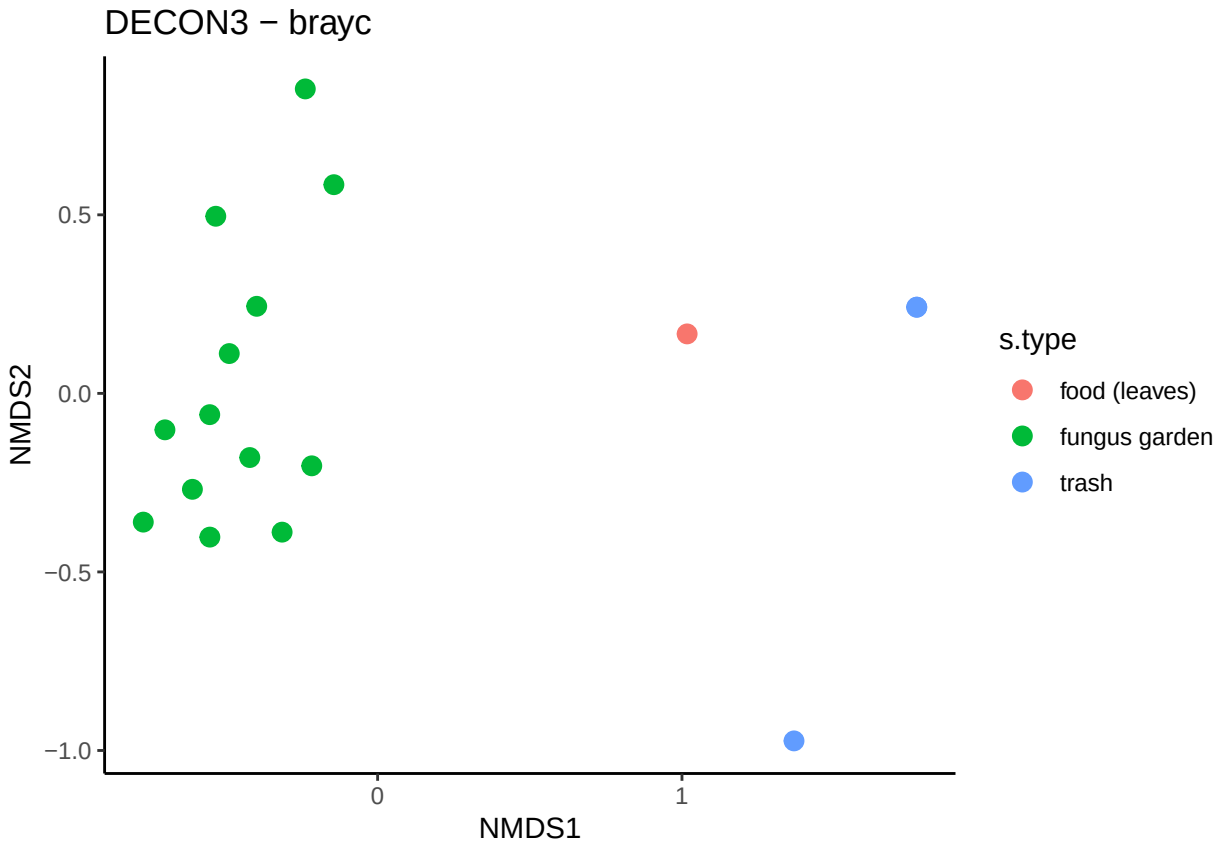
```r
# Bray Curtis NMDS
ps.bcord = ordinate(ps, method= "NMDS", distance = "bray")
```

NMDS

```
## Run 0 stress 0.08445539
## Run 1 stress 0.08437987
## ... New best solution
## ... Procrustes: rmse 0.00737288  max resid 0.02236416
## Run 2 stress 0.08678956
## Run 3 stress 0.09671085
## Run 4 stress 0.08437994
## ... Procrustes: rmse 0.0003978858  max resid 0.001258269
## ... Similar to previous best
## Run 5 stress 0.08678951
## Run 6 stress 0.08437994
## ... Procrustes: rmse 5.502832e-05  max resid 0.0001756565
## ... Similar to previous best
## Run 7 stress 0.09618321
## Run 8 stress 0.08437995
## ... Procrustes: rmse 6.6691e-05  max resid 0.0002134759
## ... Similar to previous best
## Run 9 stress 0.08437984
## ... New best solution
## ... Procrustes: rmse 5.093576e-05  max resid 0.0001561361
## ... Similar to previous best
## Run 10 stress 0.08437992
## ... Procrustes: rmse 7.019207e-05  max resid 0.000194758
## ... Similar to previous best
## Run 11 stress 0.08448634
## ... Procrustes: rmse 0.01882907  max resid 0.0589476
## Run 12 stress 0.09618321
## Run 13 stress 0.08448638
## ... Procrustes: rmse 0.01881443  max resid 0.05889311
## Run 14 stress 0.09618321
## Run 15 stress 0.09671085
```

```
## Run 16 stress 0.09618321
## Run 17 stress 0.1022811
## Run 18 stress 0.08437986
## ... Procrustes: rmse 0.0002331455  max resid 0.0007316272
## ... Similar to previous best
## Run 19 stress 0.0843799
## ... Procrustes: rmse 8.034251e-05  max resid 0.0002507088
## ... Similar to previous best
## Run 20 stress 0.08448633
## ... Procrustes: rmse 0.01881748  max resid 0.05889993
## *** Best solution repeated 4 times
```

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

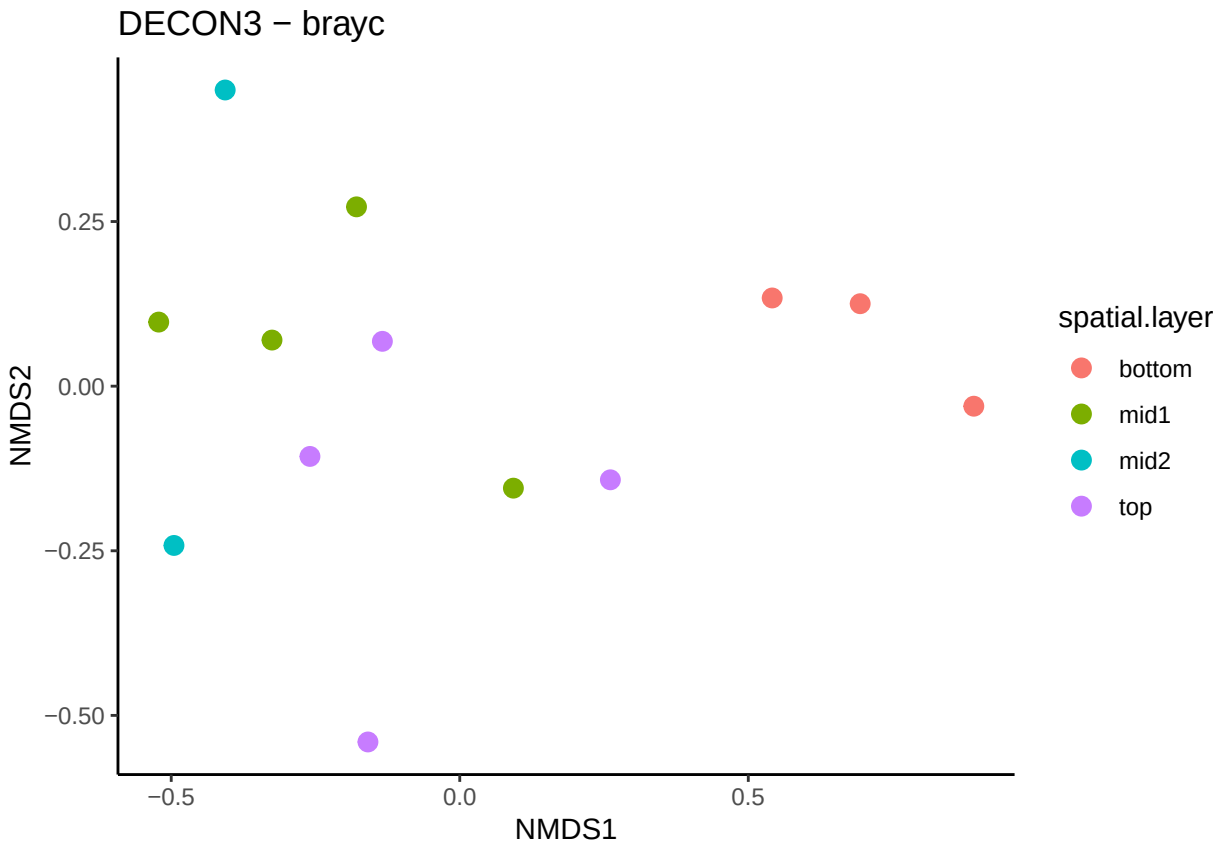



```
## fungus garden samples only
# Bray Curtis NMDS
ps.fg.bcord = ordinate(ps.fg, method= "NMDS", distance = "bray")
```

```
## Run 0 stress 0.111308
## Run 1 stress 0.1392468
## Run 2 stress 0.111308
## ... Procrustes: rmse 1.145264e-05  max resid 1.989058e-05
## ... Similar to previous best
```

```
## Run 3 stress 0.139946
## Run 4 stress 0.1346964
## Run 5 stress 0.1113081
## ... Procrustes: rmse 0.0001047996  max resid 0.000185432
## ... Similar to previous best
## Run 6 stress 0.139512
## Run 7 stress 0.1346964
## Run 8 stress 0.111308
## ... Procrustes: rmse 4.946095e-05  max resid 0.0001067946
## ... Similar to previous best
## Run 9 stress 0.1346964
## Run 10 stress 0.111308
## ... New best solution
## ... Procrustes: rmse 2.622849e-06  max resid 4.731419e-06
## ... Similar to previous best
## Run 11 stress 0.111308
## ... Procrustes: rmse 1.508789e-05  max resid 3.84947e-05
## ... Similar to previous best
## Run 12 stress 0.111308
## ... Procrustes: rmse 1.814493e-05  max resid 3.245858e-05
## ... Similar to previous best
## Run 13 stress 0.1113081
## ... Procrustes: rmse 0.0001379043  max resid 0.0002451214
## ... Similar to previous best
## Run 14 stress 0.1346964
## Run 15 stress 0.111308
## ... Procrustes: rmse 3.116586e-05  max resid 6.403466e-05
## ... Similar to previous best
## Run 16 stress 0.111308
## ... Procrustes: rmse 3.8393e-05  max resid 6.65518e-05
## ... Similar to previous best
## Run 17 stress 0.139946
## Run 18 stress 0.1392466
## Run 19 stress 0.111308
## ... Procrustes: rmse 3.314618e-05  max resid 6.447462e-05
## ... Similar to previous best
## Run 20 stress 0.1395119
## *** Best solution repeated 7 times
```

```
plot_ordination( ps.fg,
                 ps.fg.bcord,
                 # label = "s.id",
                 color = "spatial.layer") +
  geom_point(size=3) +
  ggtitle("DECON3 - brayc") +
  theme_classic()
```




```
# 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     2   1.8718 0.41315 4.9281  0.001 ***
## Residual 14   2.6588 0.58685
## Total    16   4.5306 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

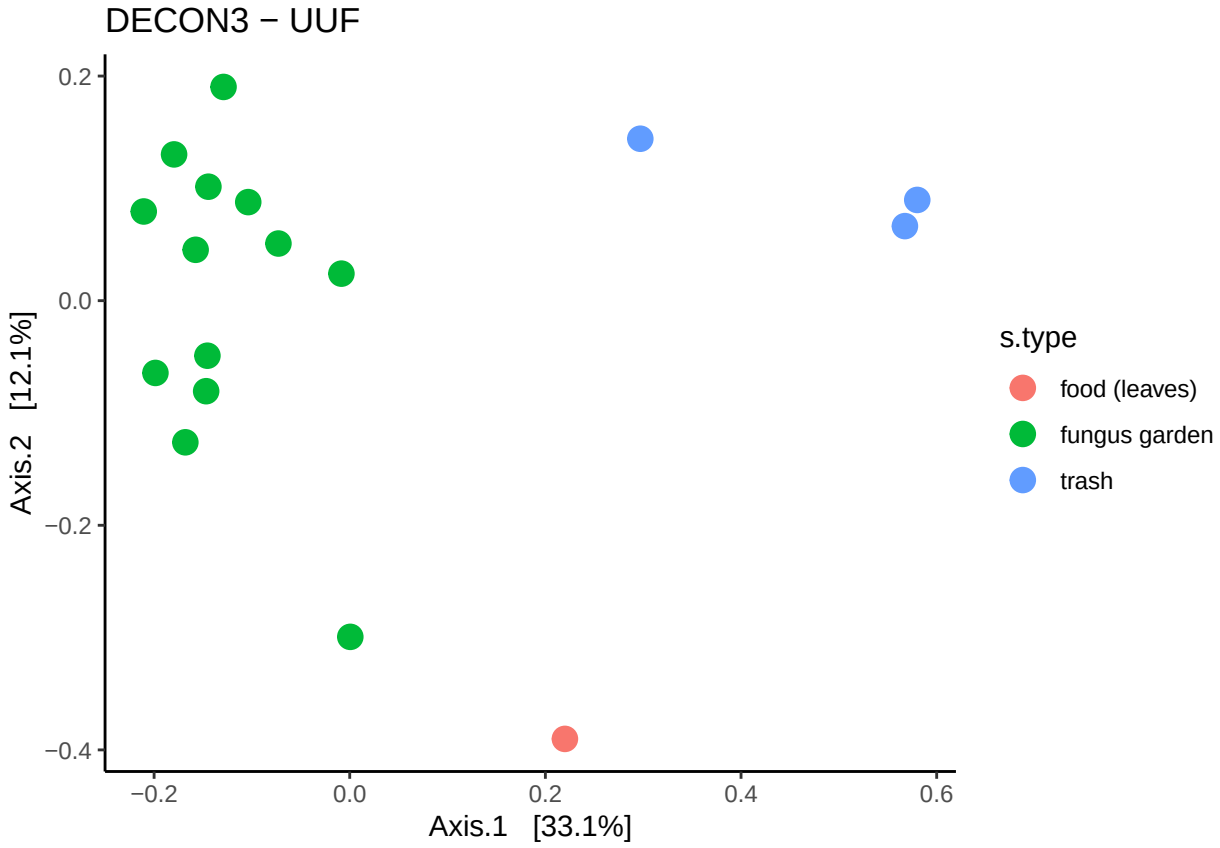
```
# Permanova - FG only
ps.fg.bcdist = distance(ps.fg, method ="bray")
adonis2( ps.fg.bcdist ~ spatial.layer,
         data = ps.fg.samdat.df
)
```

```
## 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   1.0676 0.48191 2.7905  0.002 **
## Residual  9   1.1478 0.51809
## Total    12   2.2154 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

UUF – Unweighted UNIFRAC

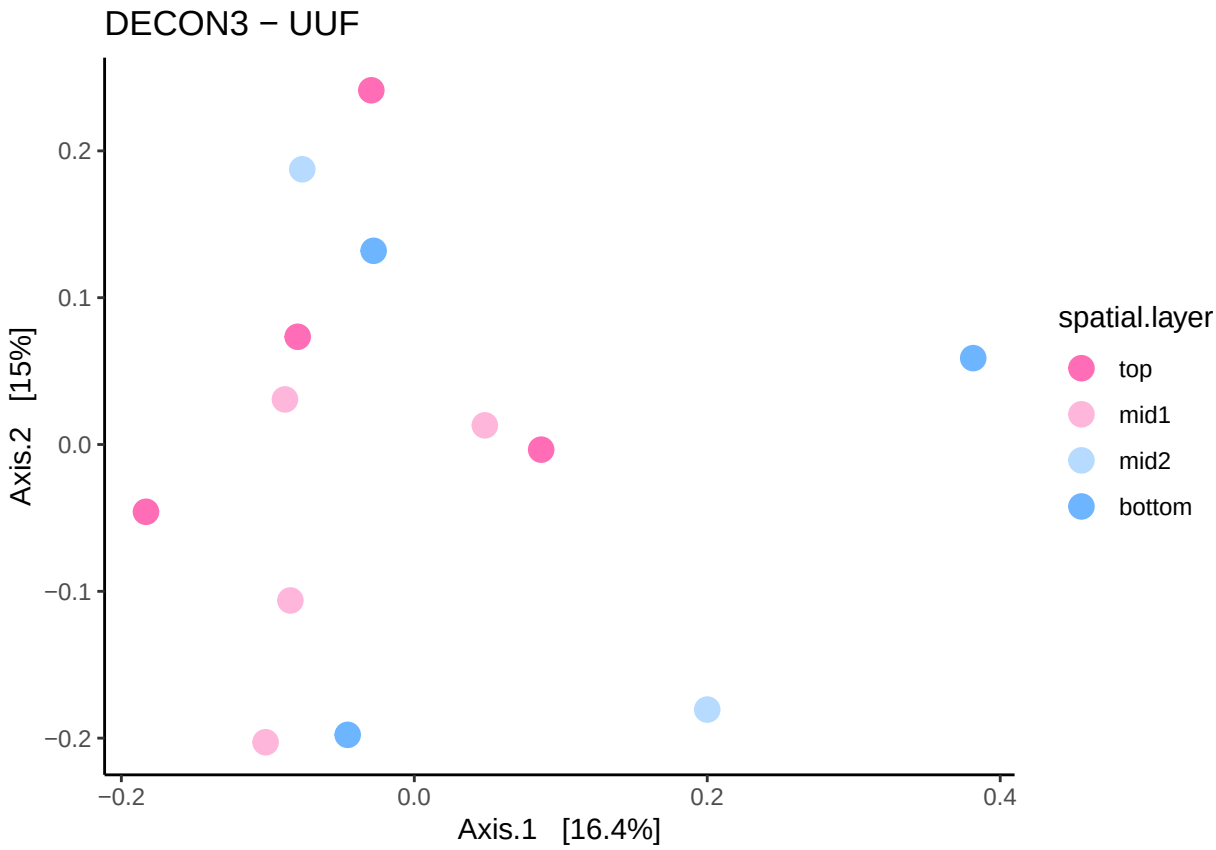
```
#Unweighted unifrac - s.type
ps.uuf.ord <- ordinate(ps, method = "PCoA", distance = "unifrac")
plot_ordination(ps, ps.uuf.ord, color = "s.type") +
  geom_point(size=4) +
  # scale_color_manual(
  #   values = unname(spatial.layer.col.ls)
  # ) +
  ggtitle("DECON3 - UUF") +
  theme_classic()
```



PCoA Plot

```
#Unweighted unifrac - spatial.layer
ps.fg.uuf.ord = ordinate(ps.fg, method = "PCoA", distance = "unifrac")
p <-
  plot_ordination(ps.fg, ps.fg.uuf.ord, color = "spatial.layer") +
  geom_point(size=4) +
  scale_color_manual(
    values = unlist(spatial.layer.col.ls)
  ) +
  ggtitle("DECON3 - UUF") +
```

```
  theme_classic()
p$data$spatial.layer = factor(p$data$spatial.layer, levels = c("top", "mid1", "mid2", "bottom"))
p
```




```
# PERMANOVA - s.type
ps.uuf.dist <- distance(subset_samples(ps, s.type != "food (leaves)"), method = "unifrac")
adonis2(ps.uuf.dist ~s.type, ps.samdat.df %>% filter(s.type != "food (leaves)"))
```

Permanova

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.uuf.dist ~ s.type, data = ps.samdat.df %>% filter(s.type != "food (leaves)"))
##          Df SumOfSqs    R2      F Pr(>F)
## Model     1  0.94172 0.322 6.6491  0.004 **
## Residual 14  1.98284 0.678
## Total    15  2.92456 1.000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

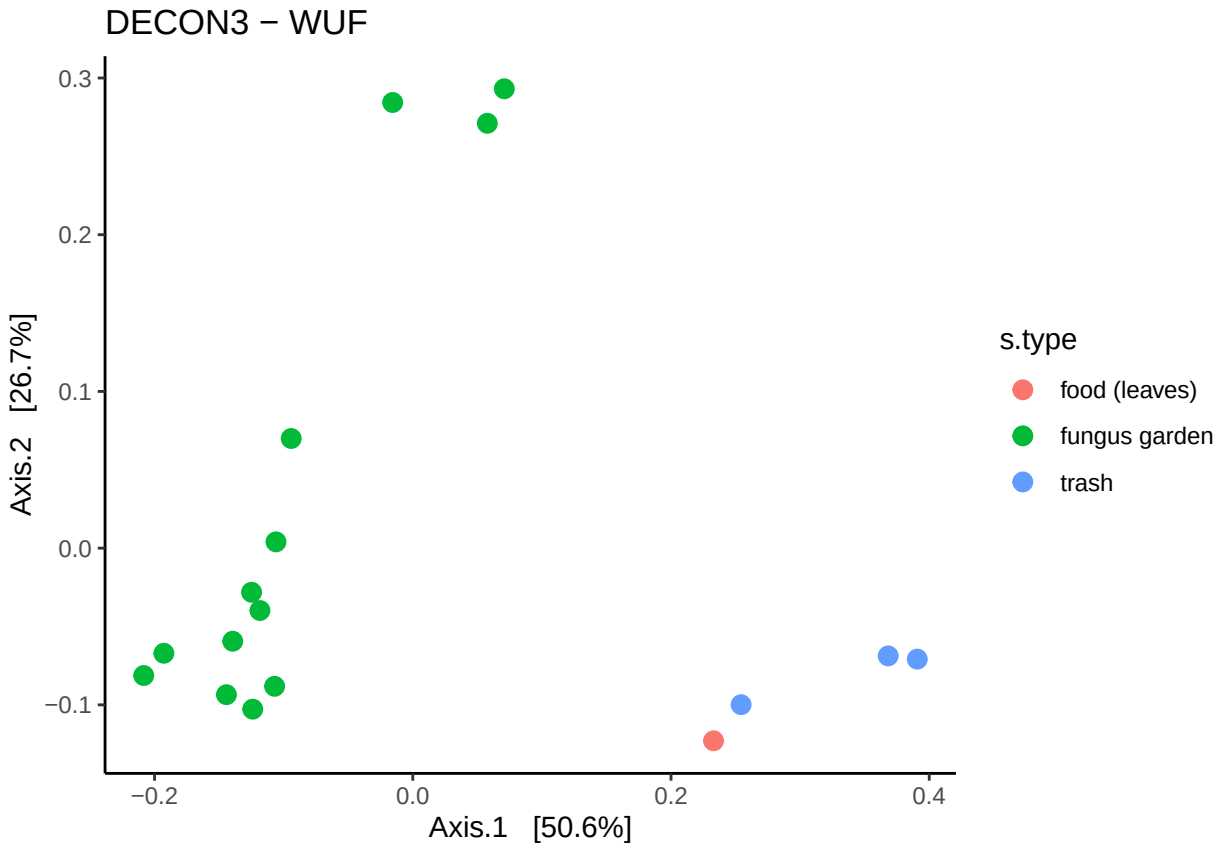
```
# PERMANOVA - spatial.layer
ps.fg.uuf.dist <- distance(ps.fg, method = "unifrac")
adonis2(ps.fg.uuf.dist ~spatial.layer, ps.fg.samdat.df)
```

```
## Permutation test for adonis under reduced model
```

```
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.fg.uuf.dist ~ spatial.layer, data = ps.fg.samdat.df)
##          Df SumOfSqs     R2      F Pr(>F)
## Model     3   0.4246 0.2573 1.0393  0.396
## Residual  9   1.2256 0.7427
## Total    12   1.6502 1.0000
```

WUF - Weighted UNIFRAC

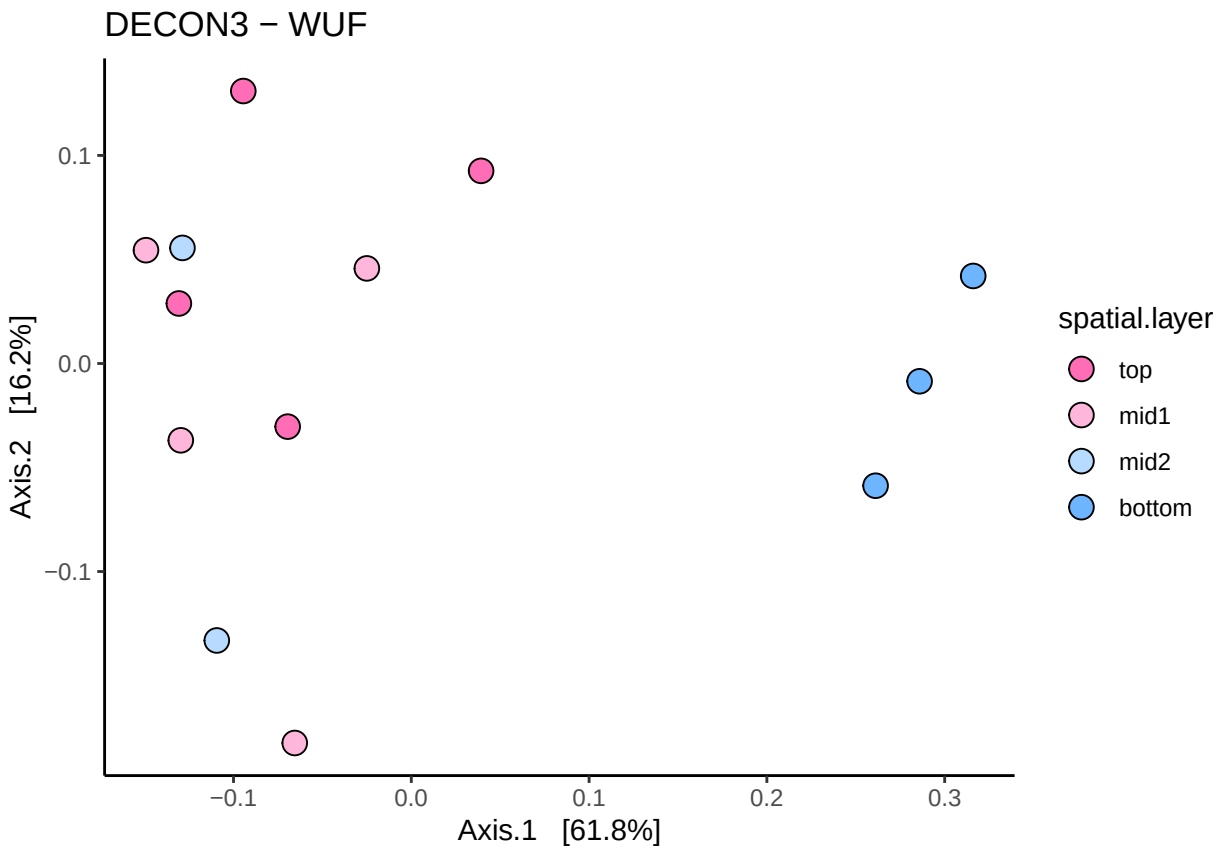
```
#weighted unifrac - s.type
ps.wuf.ord <- ordinate(ps, method = "PCoA", distance = "wunifrac")
plot_ordination(ps, ps.wuf.ord, color = "s.type") +
  geom_point(size=3
  ) +
  ggtitle("DECON3 - WUF") +
  theme_classic()
```



PCoA Plot

```
#weighted unifrac - spatial.layer
ps.fg.wuf.ord = ordinate(ps.fg, method = "PCoA", distance = "wunifrac")
p <-
  plot_ordination(ps.fg, ps.fg.wuf.ord, color = "spatial.layer") +
  geom_point(
    aes(fill = spatial.layer),
    color = "black",
    shape = 21,
```

```
    size=4
  ) +
  scale_fill_manual(
    values = unlist(spatial.layer.col.ls)
  ) +
  ggtitle("DECON3 - WUF") +
  theme_classic()
p$data$spatial.layer = factor(p$data$spatial.layer, levels = c("top", "mid1", "mid2", "bottom"))
p
```




```
# PERMANOVA - s.type
ps.wuf.dist <- distance(subset_samples(ps, s.type != "food (leaves)"), method = "wunifrac")
adonis2(ps.wuf.dist ~s.type, ps.samdat.df %>% filter(s.type != "food (leaves)"))
```

Permanova

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.wuf.dist ~ s.type, data = ps.samdat.df %>% filter(s.type != "food (leaves)"))
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1  0.49031 0.43492 10.775  0.002 **
## Residual 14  0.63704 0.56508
## Total    15  1.12735 1.00000
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# PERMANOVA - spatial.layer
ps.fg.wuf.dist <- distance(ps.fg, method = "wunifrac")
adonis2(ps.fg.wuf.dist ~spatial.layer, ps.fg.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.fg.wuf.dist ~ spatial.layer, data = ps.fg.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     3  0.38823 0.67617 6.2642  0.001 ***
## Residual  9  0.18593 0.32383
## Total    12  0.57416 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Decon4

```
ps = psList.raref.rab$decon4 # appears to be FG only, ants got rarefied out

ps.samdat.df <- data.frame(sample_data(ps))

ps.fg = subset_samples(ps, s.type =="fungus garden")
```

Bray Curtis

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

NMDS

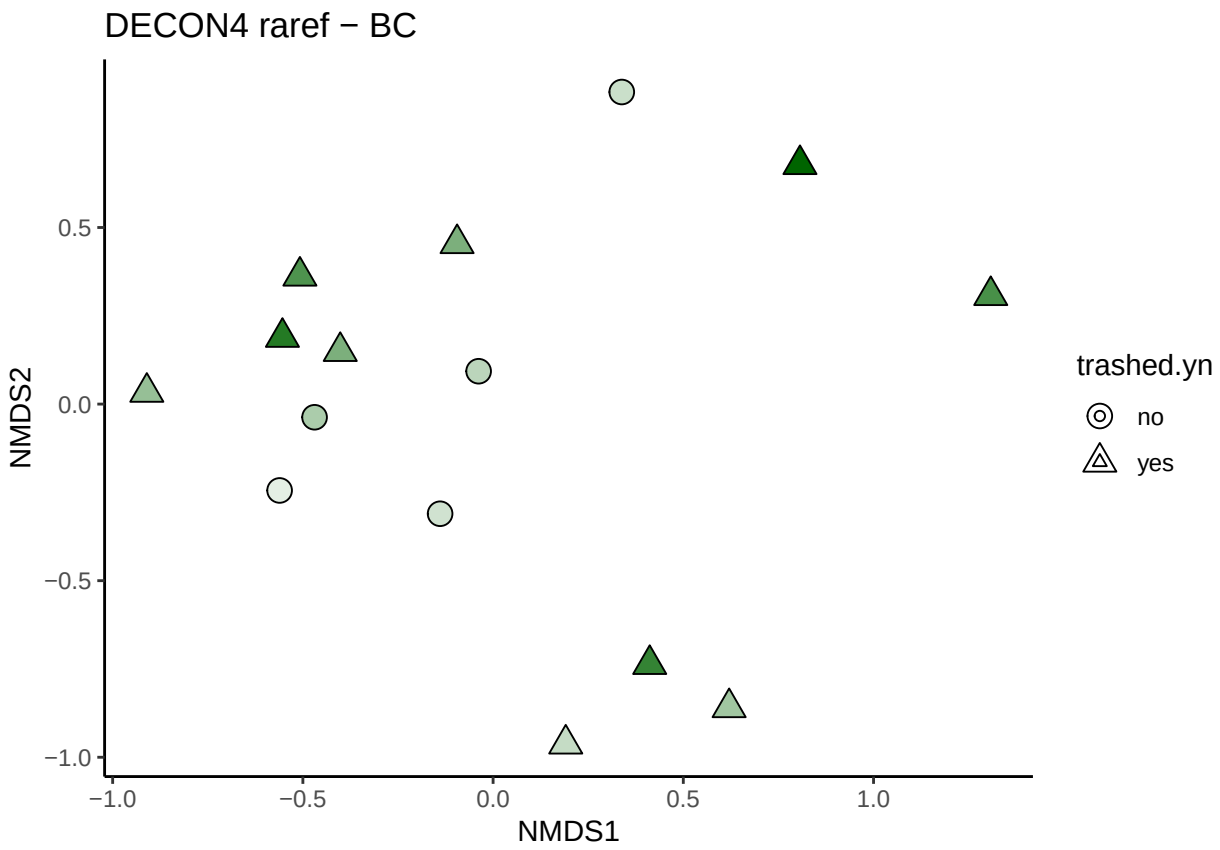
```
## Run 0 stress 0.1305038
## Run 1 stress 0.1305038
## ... Procrustes: rmse 1.509604e-06  max resid 2.833004e-06
## ... Similar to previous best
## Run 2 stress 0.1383441
## Run 3 stress 0.1383441
## Run 4 stress 0.1739109
## Run 5 stress 0.1305038
## ... Procrustes: rmse 3.620338e-06  max resid 6.799258e-06
## ... Similar to previous best
## Run 6 stress 0.1350375
## Run 7 stress 0.187256
## Run 8 stress 0.1649402
## Run 9 stress 0.1383442
## Run 10 stress 0.1305038
## ... Procrustes: rmse 1.867487e-06  max resid 3.972245e-06
## ... Similar to previous best
## Run 11 stress 0.1383441
## Run 12 stress 0.1305038
## ... Procrustes: rmse 2.597861e-06  max resid 5.706593e-06
## ... Similar to previous best
```

```
## Run 13 stress 0.1383441
## Run 14 stress 0.1649402
## Run 15 stress 0.1305038
## ... Procrustes: rmse 1.687498e-06  max resid 3.499694e-06
## ... Similar to previous best
## Run 16 stress 0.1305038
## ... Procrustes: rmse 1.100173e-06  max resid 2.459586e-06
## ... Similar to previous best
## Run 17 stress 0.1383441
## Run 18 stress 0.1875622
## Run 19 stress 0.1305038
## ... Procrustes: rmse 1.784978e-06  max resid 3.329097e-06
## ... Similar to previous best
## Run 20 stress 0.1777345
## *** Best solution repeated 7 times
```

```
p <-
  ps.fg %>%
  plot_ordination( .,
                   ps.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_distiller(palette = "Greens") +
  scale_fill_manual(
    values = setNames(full.inoc.dist.cols$col, as.factor(full.inoc.dist.cols$inoc.dist.cm)),
    guide = "none"
  ) +
  ggtitle("DECON4 raref - BC") +
  theme_classic(
  )
```

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

```
p
```




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

Permanova

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.bcdist ~ trashed.yn, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1   0.5040 0.13371 2.0066   0.05 *
## Residual 13   3.2652 0.86629
## Total    14   3.7692 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.bcdist ~ inoc.dist.cm, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1   0.4611 0.12233 1.8119  0.083 .
```

```
## Residual 13   3.3081 0.87767
## Total    14   3.7692 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

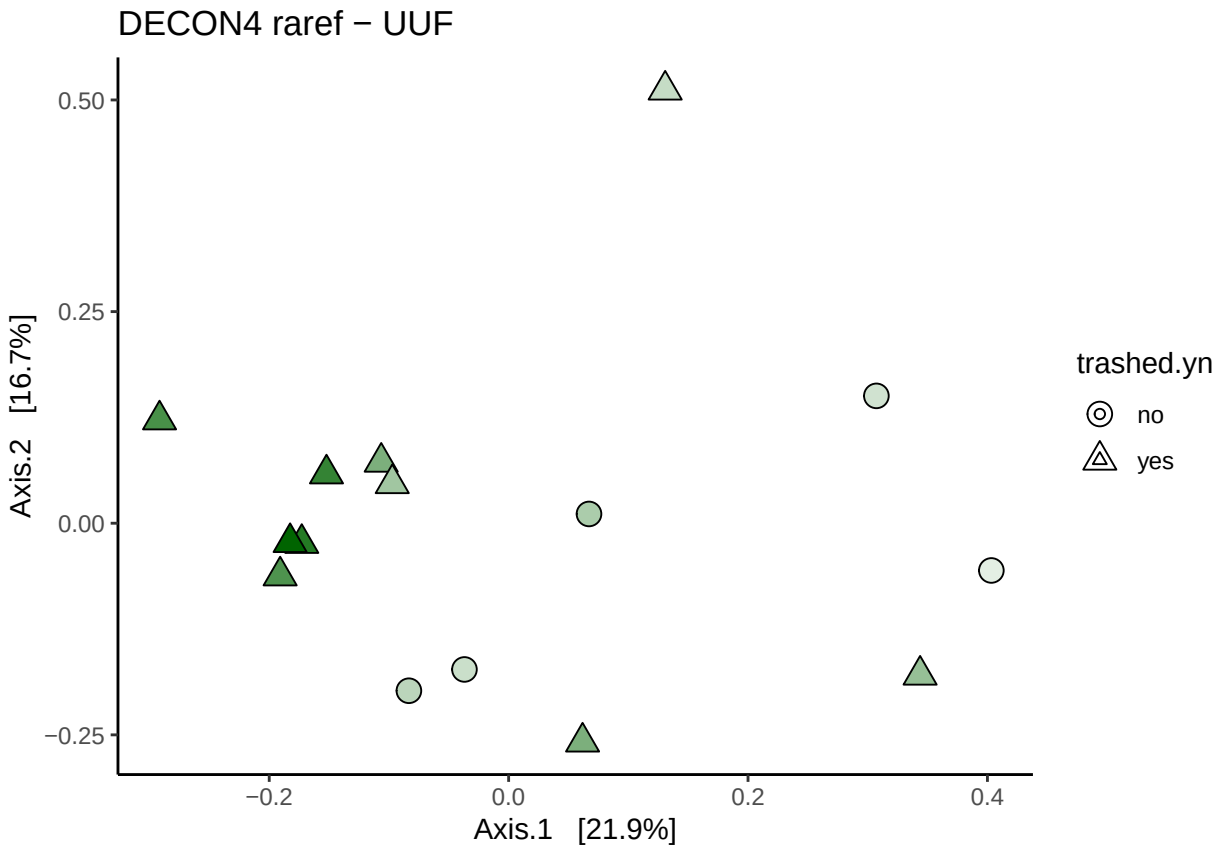
UUF - Unweighted UNIFRAC

```
ps.uuf.ord <- ordinate(ps.fg, method = "PCoA", distance = "unifrac")

# # Unweighted unifrac - trashed.yn
# ps.uuf.ord <- ordinate(ps, method = "PCoA", distance = "unifrac")
# plot_ordination(ps, ps.uuf.ord, color = "trashed.yn") +
#   geom_point(size = 3) +
#   # scale_color_distiller(palette="Greens") +
#   ggtitle("DECON4 - UUF") +
#   theme_classic()

# Unweighted unifrac - inoc.dist
p <-
  ps.fg %>%
  plot_ordination( .,
                   ps.uuf.ord,
                   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_distiller(palette = "Greens") +
  scale_fill_manual(
    values = setNames(full.inoc.dist.cols$col, as.factor(full.inoc.dist.cols$inoc.dist.cm)),
    guide = "none"
  ) +
  ggtitle("DECON4 raref - UUF") +
  theme_classic(
  )
```

PCoA Plot

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

```
p
```




```
ps.uuf.dist <- distance(ps, method = "unifrac")
adonis2(ps.uuf.dist ~ trashed.yn, ps.samdat.df) # PERMANOVA - trashed.yn
```

Permanova

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.uuf.dist ~ trashed.yn, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1  0.29215 0.10017 1.4472  0.107
## Residual 13  2.62436 0.89983
## Total    14  2.91651 1.00000
```

```
adonis2(ps.uuf.dist ~ inoc.dist.cm, ps.samdat.df)  # PERMANOVA - inoc.dist.cm
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.uuf.dist ~ inoc.dist.cm, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1  0.38622 0.13243 1.9843   0.01 **
## Residual 13  2.53029 0.86757
## Total    14  2.91651 1.00000
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

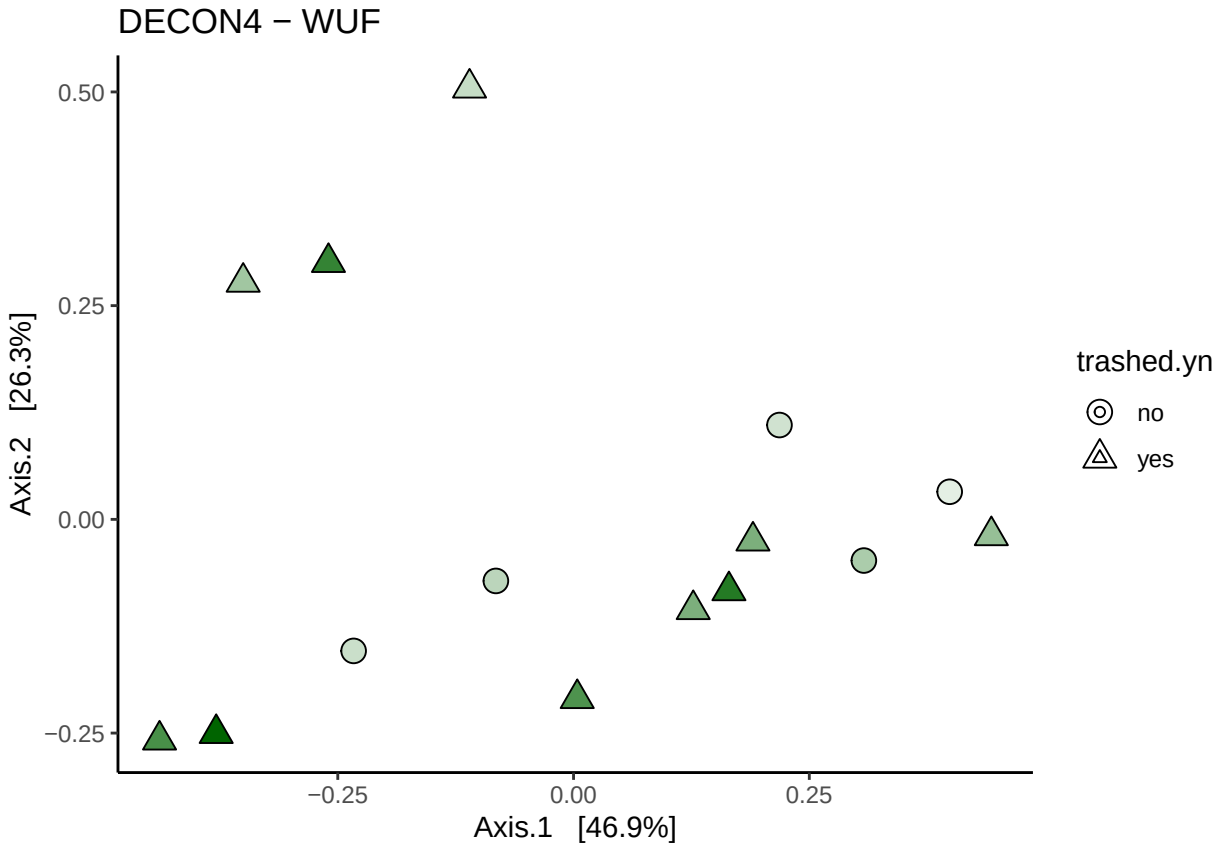
WUF - Weighted UNIFRAC

```
ps.fg =
  ps %>%
  subset_samples(s.type == "fungus garden")

ps.fg.wuf.ord <- ordinate(ps.fg, method = "PCoA", distance = "wunifrac")

# # WUF trashed.yn
# plot_ordination(ps, ps.wuf.ord, color = "trashed.yn") +
#   geom_point(size=3) +
#   ggtitle("DECON4 - WUF") +
#   theme_classic()

# WUF inoc.dist
p <-
  ps.fg %>%
  plot_ordination( .,
                   ps.fg.wuf.ord,
                   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_distiller(palette = "Greens") +
  scale_fill_manual(
    values = setNames(full.inoc.dist.cols$col, as.factor(full.inoc.dist.cols$inoc.dist.cm)),
    guide = "none"
  ) +
  ggtitle("DECON4 - WUF") +
  theme_classic(
  )
```

PCoA Plot

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

```
p
```




```
ps.wuf.dist <- distance(ps, method = "wunifrac")
adonis2(ps.wuf.dist ~ inoc.dist.cm, ps.samdat.df)  # PERMANOVA - inoc.dist
```

Permanova

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.wuf.dist ~ inoc.dist.cm, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1  0.37372 0.14903 2.2767  0.073 .
## Residual 13  2.13390 0.85097
## Total    14  2.50762 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
adonis2(ps.wuf.dist ~ trashed.yn, ps.samdat.df) # PERMANOVA - trashed.yn
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.wuf.dist ~ trashed.yn, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1  0.25146 0.10028 1.4489  0.198
## Residual 13  2.25615 0.89972
```

```
## Total    14  2.50762 1.00000
```

Decon5

```
ps = psList.raref.rab$decon5
ps.samdat.df <- data.frame(sample_data(ps))

ps.fg = ps %>% subset_samples(s.type == "fungus garden")
ps.fg.samdat.df = ps.samdat.df %>% filter(s.type == "fungus garden")
```

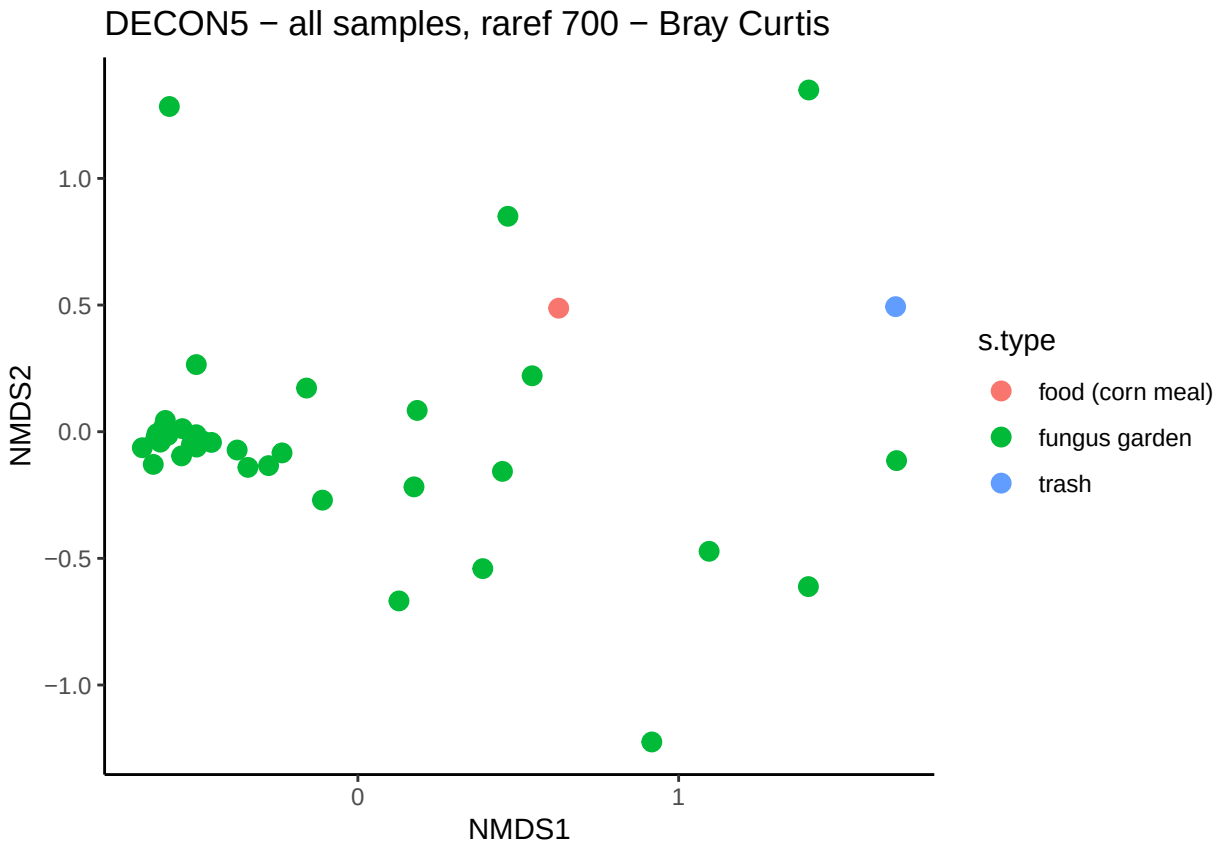
Bray Curtis

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

NMDS

```
## Run 0 stress 0.1181622
## Run 1 stress 0.131419
## Run 2 stress 0.1278612
## Run 3 stress 0.1298876
## Run 4 stress 0.1261617
## Run 5 stress 0.1768601
## Run 6 stress 0.12149
## Run 7 stress 0.1274496
## Run 8 stress 0.1190619
## Run 9 stress 0.12798
## Run 10 stress 0.123427
## Run 11 stress 0.1323138
## Run 12 stress 0.120326
## Run 13 stress 0.1230783
## Run 14 stress 0.1291381
## Run 15 stress 0.1271361
## Run 16 stress 0.1393763
## Run 17 stress 0.121883
## Run 18 stress 0.1416829
## Run 19 stress 0.1284636
## Run 20 stress 0.12149
## *** Best solution was not repeated -- monoMDS stopping criteria:
##     19: stress ratio > sratmax
##      1: scale factor of the gradient < sfgrmin
```

```
plot_ordination( ps,
                 ps.bcord,
                 color = "s.type") +
  geom_point(size = 3) +
  ggtitle("DECON5 - all samples, raref 700 - Bray Curtis") +
  theme_classic()
```

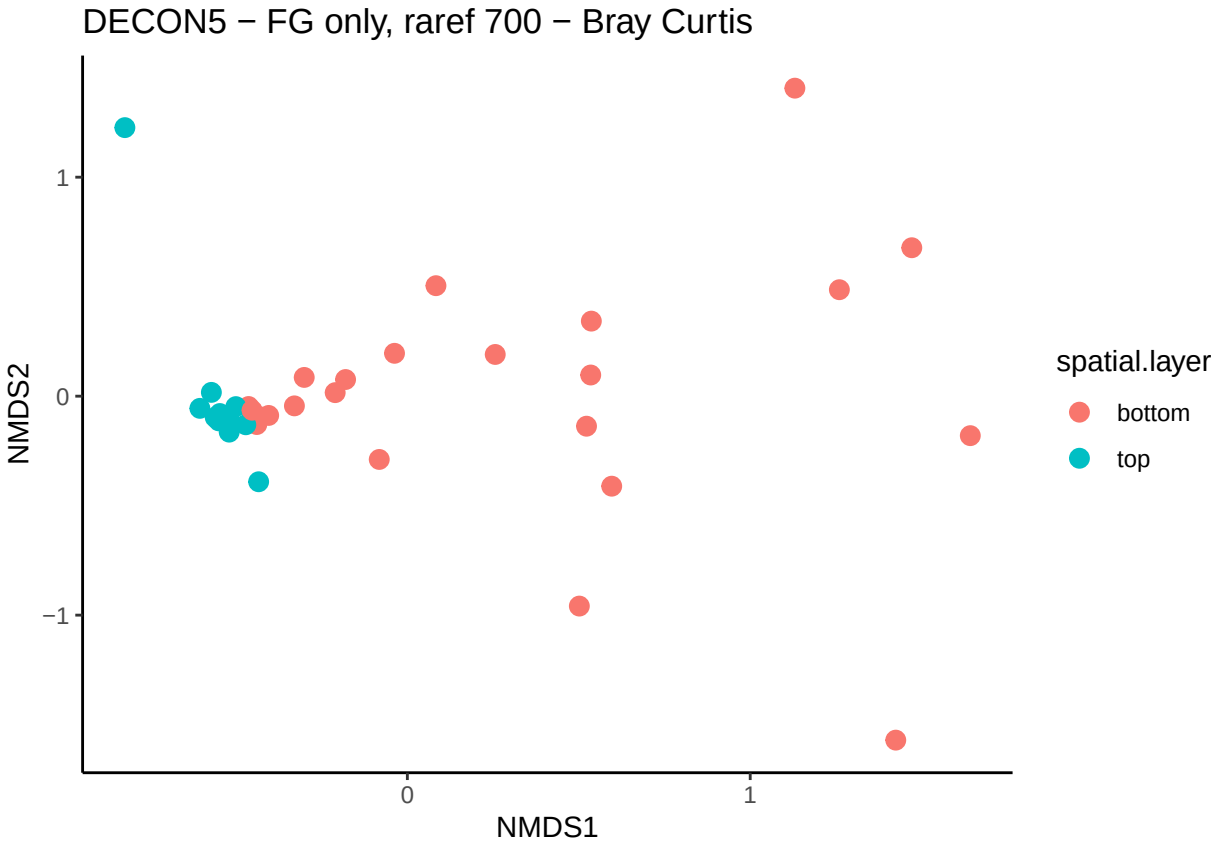



```
## fungus garden samples only
# Bray Curtis NMDS
ps.fg.bcord = ordinate(ps.fg, method= "NMDS", distance = "bray")
```

```
## Run 0 stress 0.1193341
## Run 1 stress 0.1268526
## Run 2 stress 0.1137504
## ... New best solution
## ... Procrustes: rmse 0.09845544  max resid 0.470874
## Run 3 stress 0.1113613
## ... New best solution
## ... Procrustes: rmse 0.0844767  max resid 0.3330779
## Run 4 stress 0.1227647
## Run 5 stress 0.1199943
## Run 6 stress 0.1247016
## Run 7 stress 0.1326607
## Run 8 stress 0.1294464
## Run 9 stress 0.1136154
## Run 10 stress 0.1136155
## Run 11 stress 0.1317112
## Run 12 stress 0.1259115
## Run 13 stress 0.1202921
## Run 14 stress 0.1113613
## ... New best solution
## ... Procrustes: rmse 5.424027e-05  max resid 0.0002275834
## ... Similar to previous best
## Run 15 stress 0.1212529
```

```
## Run 16 stress 0.1284985
## Run 17 stress 0.1113613
## ... New best solution
## ... Procrustes: rmse 0.0001152502  max resid 0.0005540743
## ... Similar to previous best
## Run 18 stress 0.1202032
## Run 19 stress 0.1137776
## Run 20 stress 0.1142713
## *** Best solution repeated 1 times
```

```
plot_ordination( ps.fg,
                 ps.fg.bcord,
                 color = "spatial.layer") +
  geom_point(size = 3) +
  ggtitle("DECON5 - FG only, raref 700 - Bray Curtis") +
  theme_classic()
```



Permanova Sample type

Can't do! Only one sample of food and one of trash!

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

Spatial layer test - Fungus gardens only

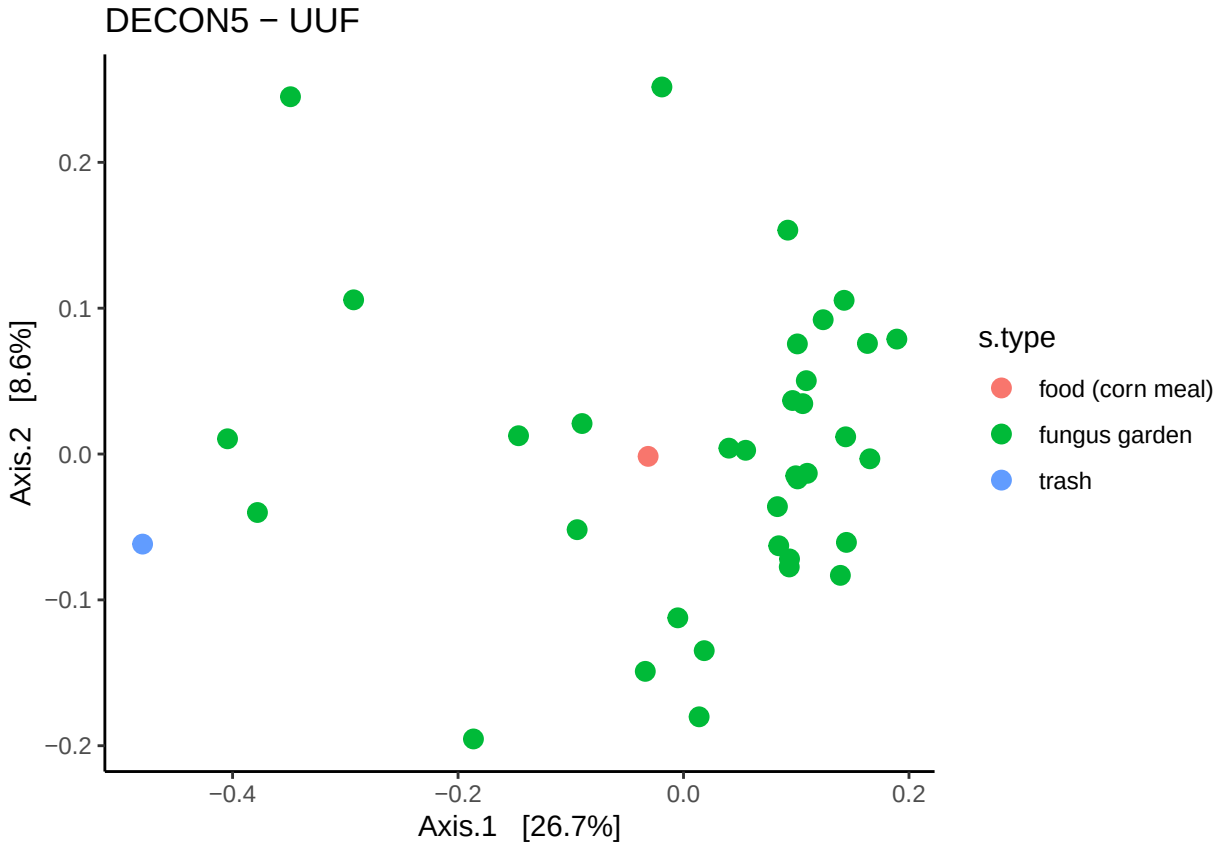
```
# Permanova - FG only
ps.fg.bcdist = distance(ps.fg, method ="bray")
```

```
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   0.8869 0.13976 5.3615  0.001 ***
## Residual 33   5.4588 0.86024
## Total    34   6.3457 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

UUF - Unweighted UNIFRAC

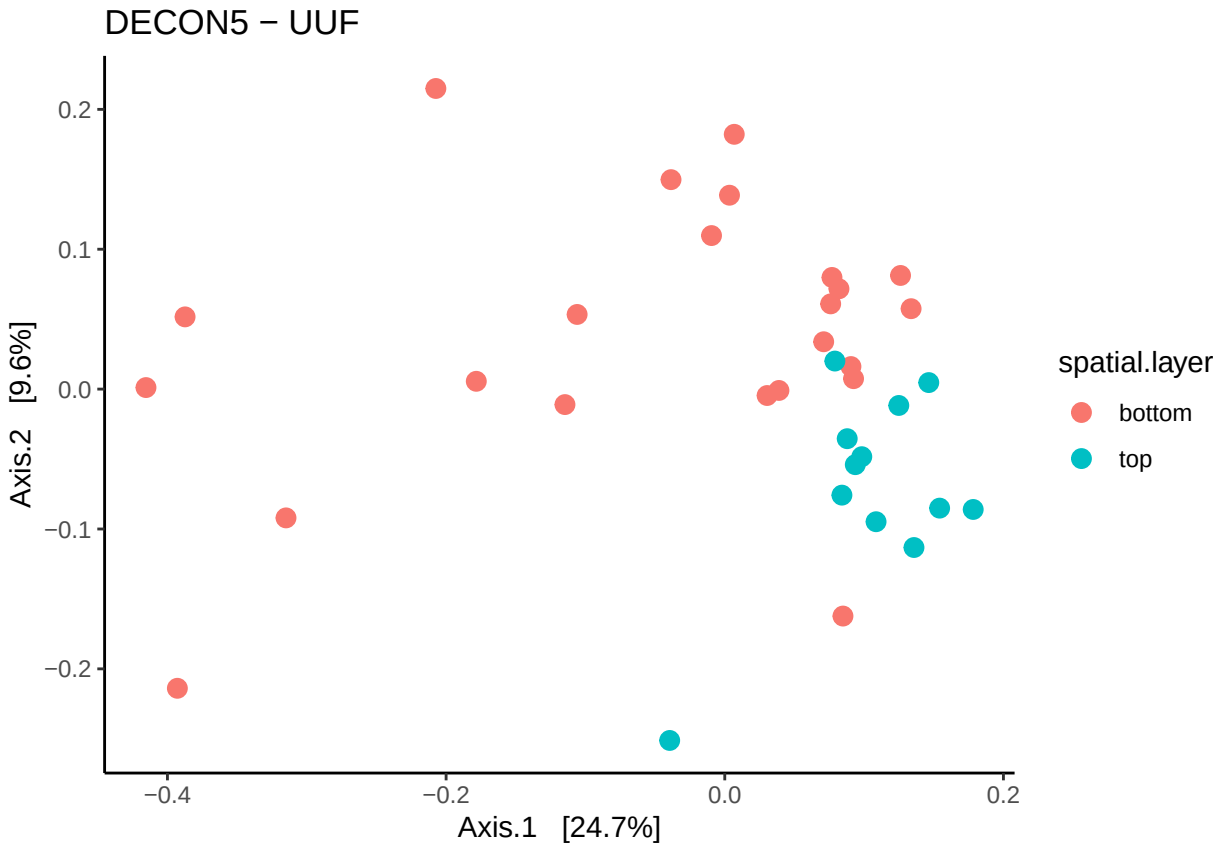
```
#Unweighted unifrac - s.type
ps.uuf.ord <- ordinate(ps, method = "PCoA", distance = "unifrac")
plot_ordination(ps, ps.uuf.ord, color = "s.type") +
  geom_point(size = 3) +
  ggtitle("DECON5 - UUF") +
  theme_classic()
```



PCoA

```
#Unweighted unifrac - spatial.layer
ps.fg.uuf.ord = ordinate(ps.fg, method = "PCoA", distance = "unifrac")
```

```
plot_ordination(ps.fg, ps.fg.uuf.ord, color = "spatial.layer") +
  geom_point(size = 3) +
  ggtitle("DECON5 - UUF") +
  theme_classic()
```




```
# PERMANOVA - s.type
ps.uuf.dist <- distance(ps, method = "unifrac")
adonis2(ps.uuf.dist ~ s.type, ps.samdat.df)
```

Permanova

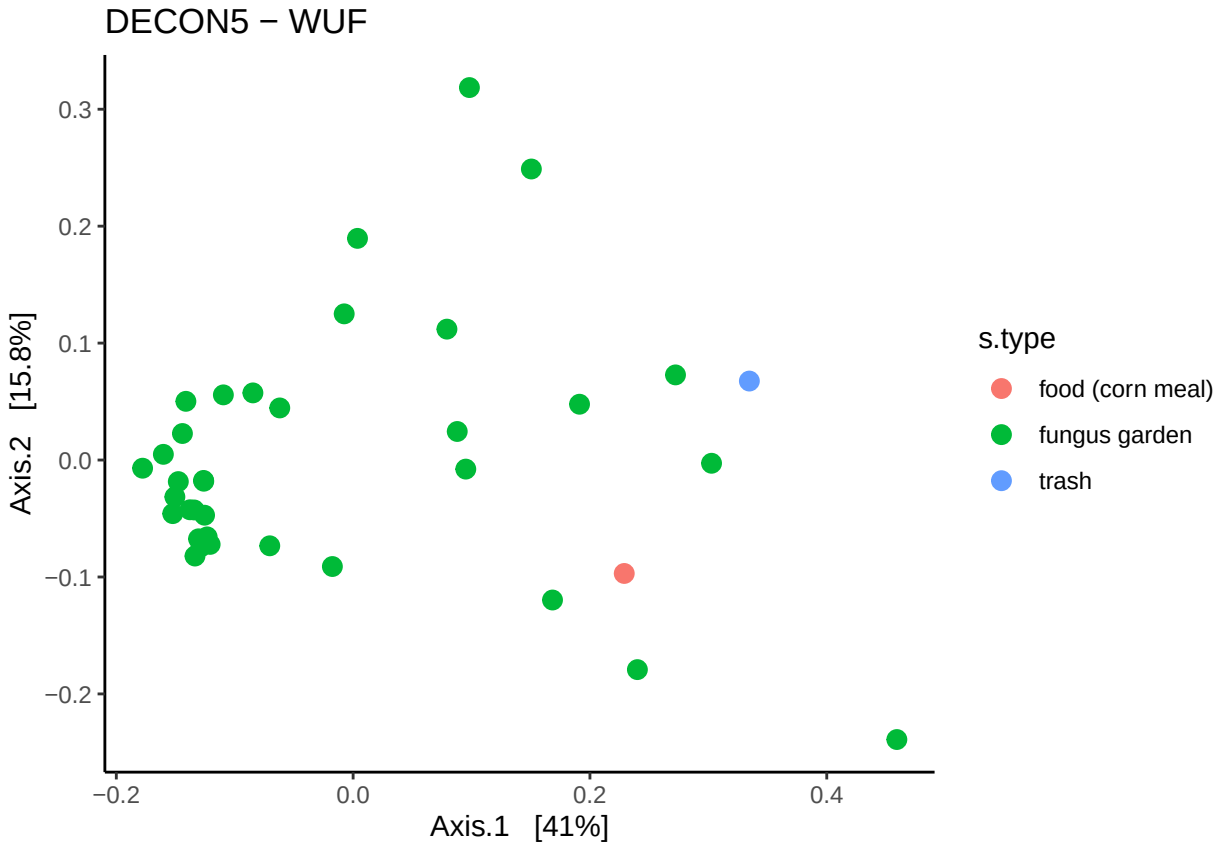
```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.uuf.dist ~ s.type, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     2   0.4719 0.11185 2.1408  0.023 *
## Residual 34   3.7472 0.88815
## Total    36   4.2191 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# PERMANOVA - spatial.layer
ps.fg.uuf.dist <- distance(ps.fg, method = "unifrac")
adonis2(ps.fg.uuf.dist ~ spatial.layer, ps.fg.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.fg.uuf.dist ~ spatial.layer, data = ps.fg.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1   0.3357 0.08959 3.2474  0.002 **
## Residual 33   3.4115 0.91041
## Total    34   3.7472 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

WUF - Weighted UNIFRAC

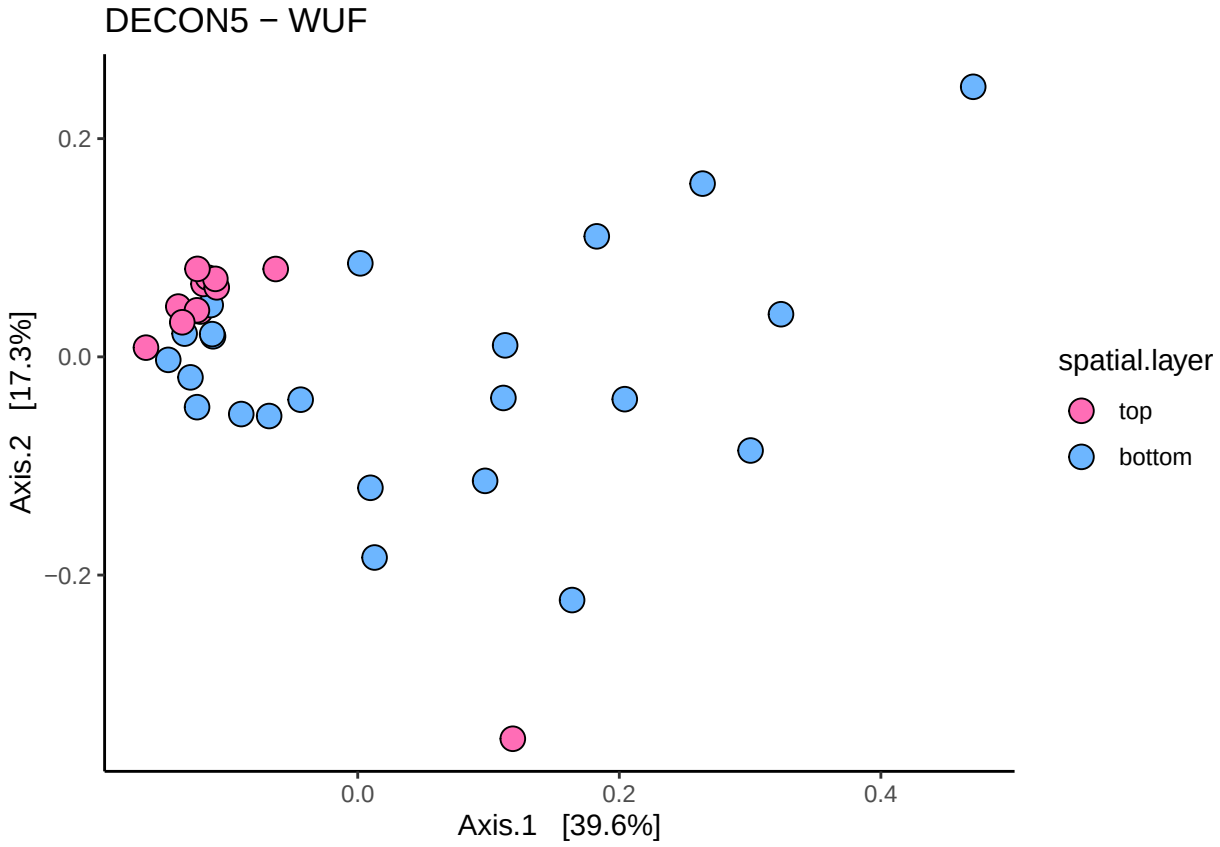
```
#weighted unifrac - s.type
ps.wuf.ord <- ordinate(ps, method = "PCoA", distance = "wunifrac")
plot_ordination(ps, ps.wuf.ord, color = "s.type") +
  geom_point(size = 3) +
  ggtitle("DECON5 - WUF") +
  theme_classic()
```



PCoA

```
#weighted unifrac - spatial.layer
ps.fg.wuf.ord = ordinate(ps.fg, method = "PCoA", distance = "wunifrac")
p <-
  plot_ordination(ps.fg, ps.fg.wuf.ord, color = "spatial.layer") +
  geom_point(
    aes(fill = spatial.layer),
```

```
    color = "black",
    shape = 21,
    size=4
  ) +
  scale_fill_manual(
    values = unlist(spatial.layer.col.ls)
  ) +
  ggtitle("DECON5 - WUF") +
  theme_classic()
p$data$spatial.layer = factor(p$data$spatial.layer, levels=c("top","bottom"))
p
```




```
# PERMANOVA - s.type
ps.wuf.dist <- distance(ps, method = "wunifrac")
adonis2(ps.wuf.dist ~s.type, ps.samdat.df)
```

Permanova

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.wuf.dist ~ s.type, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     2  0.30426 0.11564 2.2229  0.154
## Residual 34  2.32689 0.88436
```

```
## Total    36  2.63116 1.00000
```

```
# PERMANOVA - spatial.layer
ps.fg.wuf.dist <- distance(ps.fg, method = "wunifrac")
adonis2(ps.fg.wuf.dist ~spatial.layer, ps.fg.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.fg.wuf.dist ~ spatial.layer, data = ps.fg.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1  0.26848 0.11538 4.3042  0.003 **
## Residual 33  2.05841 0.88462
## Total    34  2.32689 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

TSinf

```
ps = psList.raref.rab$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")
  )

ps.samdat.df <- data.frame(sample_data(ps))
```

Bray Curtis

```
exp.inf.col.ls = list("yes" = "#004949",
                      "maybe" = "#009292",
                      "no"= "#b6dbff")
# Bray Curtis NMDS
```

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

ord.expect.inf.treat.ants

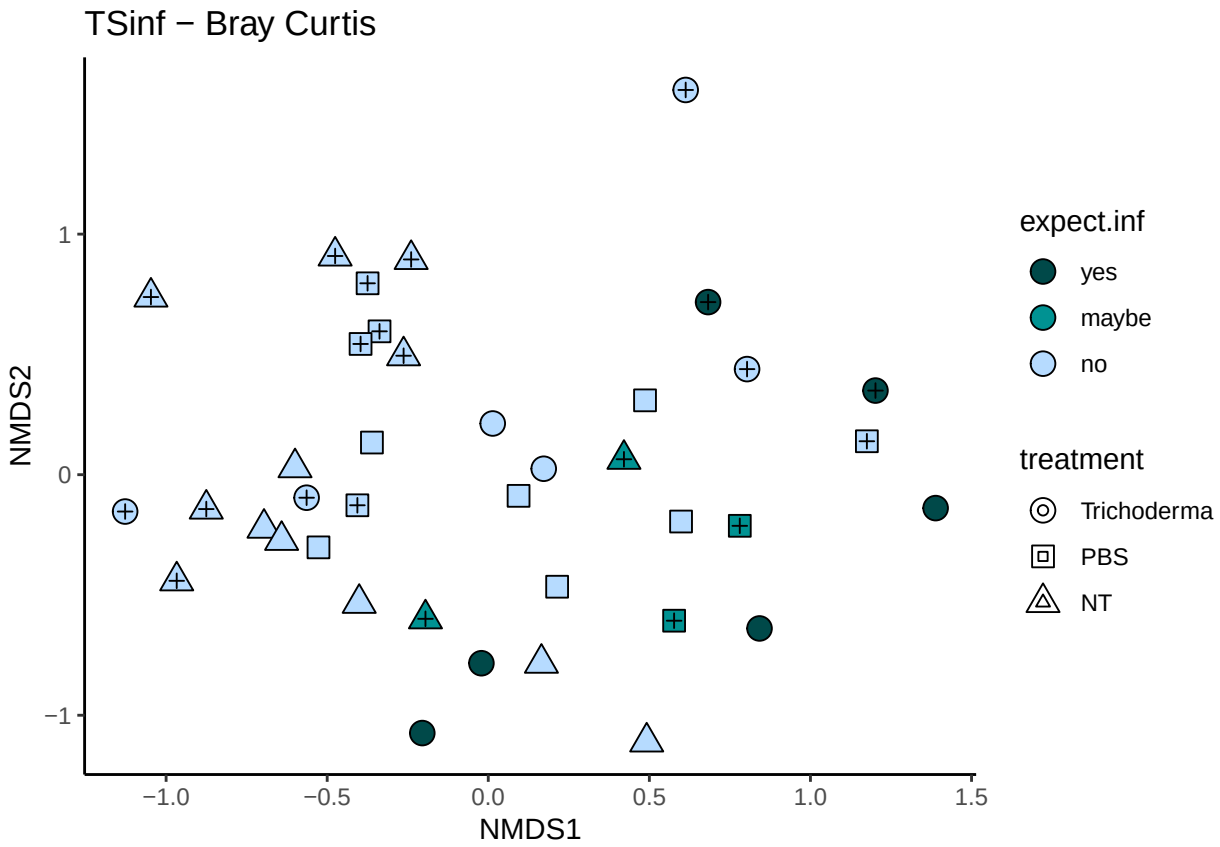
```
## Run 0 stress 0.2070791
## Run 1 stress 0.2232888
## Run 2 stress 0.282032
## Run 3 stress 0.2070791
## ... New best solution
## ... Procrustes: rmse 2.798772e-05  max resid 8.084485e-05
## ... Similar to previous best
## Run 4 stress 0.2597168
## Run 5 stress 0.2193524
## Run 6 stress 0.2135854
## Run 7 stress 0.2241212
## Run 8 stress 0.2195224
## Run 9 stress 0.240197
## Run 10 stress 0.2070792
## ... Procrustes: rmse 4.150151e-05  max resid 0.0001245899
## ... Similar to previous best
## Run 11 stress 0.2070791
## ... Procrustes: rmse 6.636665e-06  max resid 3.200301e-05
## ... Similar to previous best
## Run 12 stress 0.2250984
## Run 13 stress 0.2993819
## Run 14 stress 0.216397
## Run 15 stress 0.2195233
## Run 16 stress 0.2135854
## Run 17 stress 0.216397
## Run 18 stress 0.2070791
## ... Procrustes: rmse 1.020098e-05  max resid 3.362979e-05
## ... Similar to previous best
## Run 19 stress 0.2070791
## ... Procrustes: rmse 6.97978e-06  max resid 3.574336e-05
## ... Similar to previous best
## Run 20 stress 0.2375185
## *** Best solution repeated 5 times
```

```
p <-
  plot_ordination( ps,
                   ps.bcord,
                   color = "black",
                   shape = "treatment"
  ) +
  geom_point(
    aes(fill = expect.inf),
    size = 4
  ) +
  # geom_point(
  #   data = ants.y.bcord,
  #   aes()
  # ) +
  ggtitle("TSinf - Bray Curtis") +
  scale_fill_manual(
    values = exp.inf.col.ls
```

```
    # values = c("white", "lightblue", "darkgreen")
  ) +
  scale_shape_manual(
    values = c(21, 22, 24)
  ) +
  theme_classic() +
  guides(fill = guide_legend(override.aes = list(shape = 21)))
```

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

```
# order legend
p$data$expect.inf = factor(p$data$expect.inf, levels = c("yes", "maybe", "no"))
p$data$treatment = factor(p$data$treatment, levels = c("Trichoderma", "PBS", "NT"))
# 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,
  # stroke = 1.1
)
```



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

ord.s.type

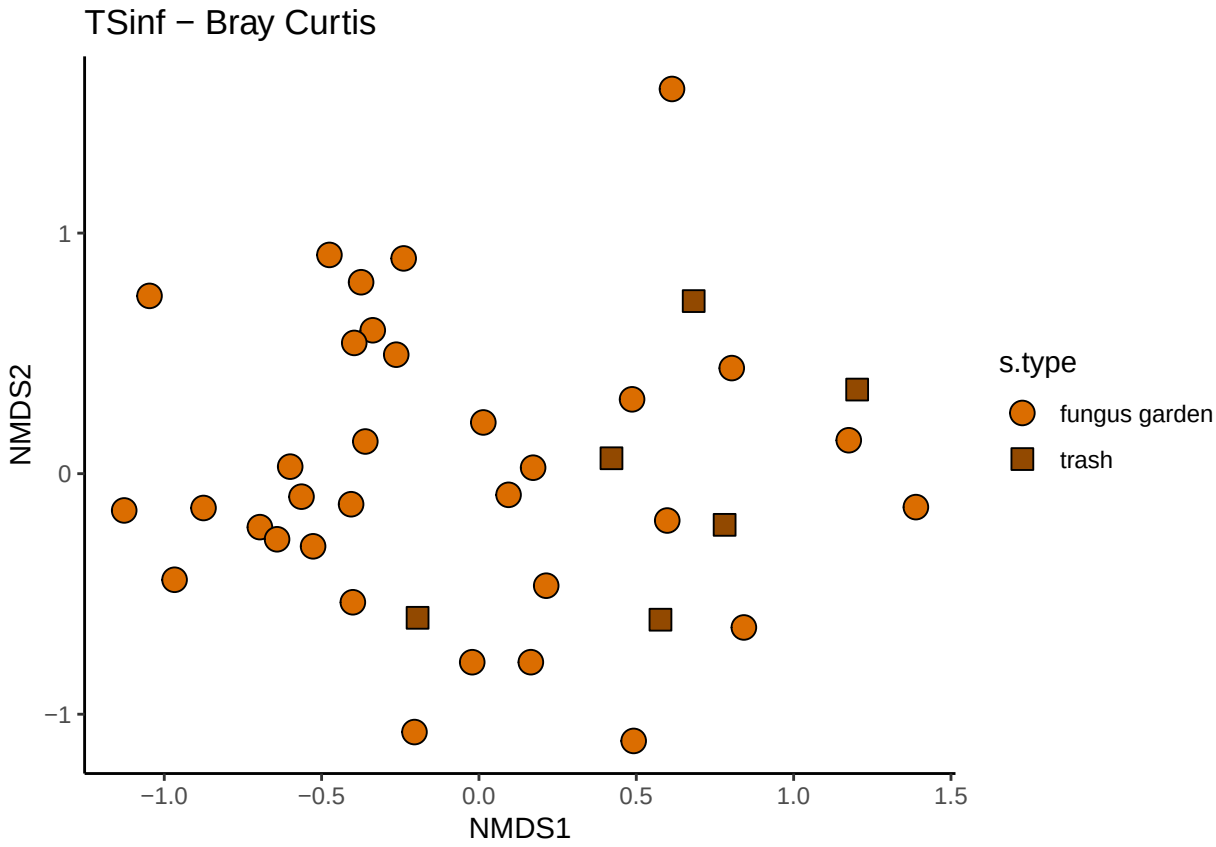
```
## Run 0 stress 0.2070791
## Run 1 stress 0.2347229
## Run 2 stress 0.2740233
## Run 3 stress 0.2135854
## Run 4 stress 0.2135854
## Run 5 stress 0.2070792
## ... Procrustes: rmse 3.276633e-05  max resid 0.0001017686
## ... Similar to previous best
## Run 6 stress 0.2365334
## Run 7 stress 0.2070792
## ... Procrustes: rmse 2.578025e-05  max resid 8.750104e-05
## ... Similar to previous best
## Run 8 stress 0.2251063
## Run 9 stress 0.2446553
## Run 10 stress 0.246872
## Run 11 stress 0.2135854
## Run 12 stress 0.2070792
## ... Procrustes: rmse 5.948437e-05  max resid 0.0002057891
## ... Similar to previous best
## Run 13 stress 0.225099
## Run 14 stress 0.2241095
## Run 15 stress 0.2508027
## Run 16 stress 0.2166055
## Run 17 stress 0.2070792
## ... Procrustes: rmse 2.663357e-05  max resid 7.609519e-05
## ... Similar to previous best
## Run 18 stress 0.2392374
## Run 19 stress 0.2706407
## Run 20 stress 0.2070791
## ... New best solution
## ... Procrustes: rmse 3.769847e-05  max resid 0.0001272788
## ... Similar to previous best
## *** Best solution repeated 1 times
```

```
p <-
  plot_ordination( ps,
                   ps.bcord,
                   color = "black",
                   shape = "s.type"
  ) +
  geom_point(
    aes(fill = s.type),
    size = 4,
    # shape = 21
  ) +
  ggtitle("TSinf - Bray Curtis") +
  scale_fill_manual(
    values = s.type.col.ls
  ) +
```

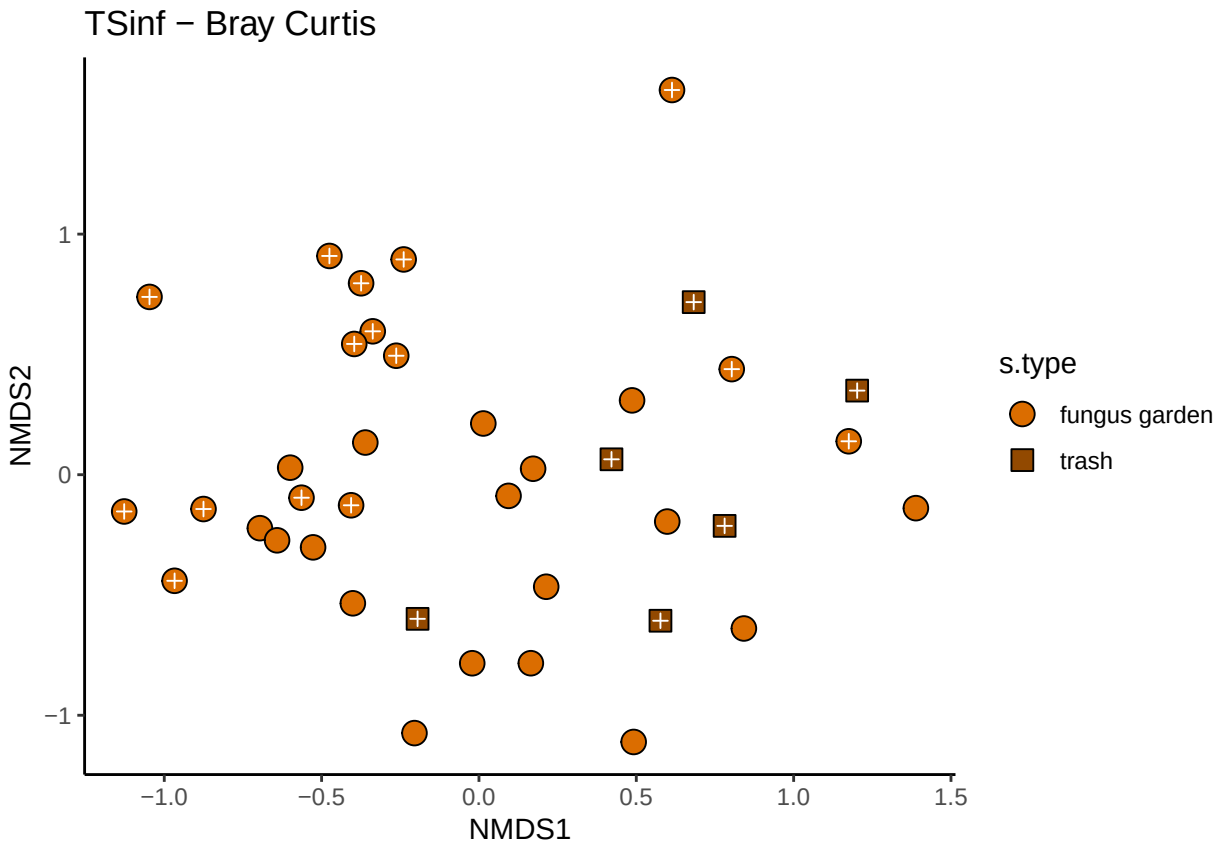
```
  scale_shape_manual(
    values = c(21, 22, 24)
  ) +
  theme_classic() +
  guides(fill = guide_legend(override.aes = list(shape = c(21,22))))
```

```
## 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
)
```



ord.ants.yn

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

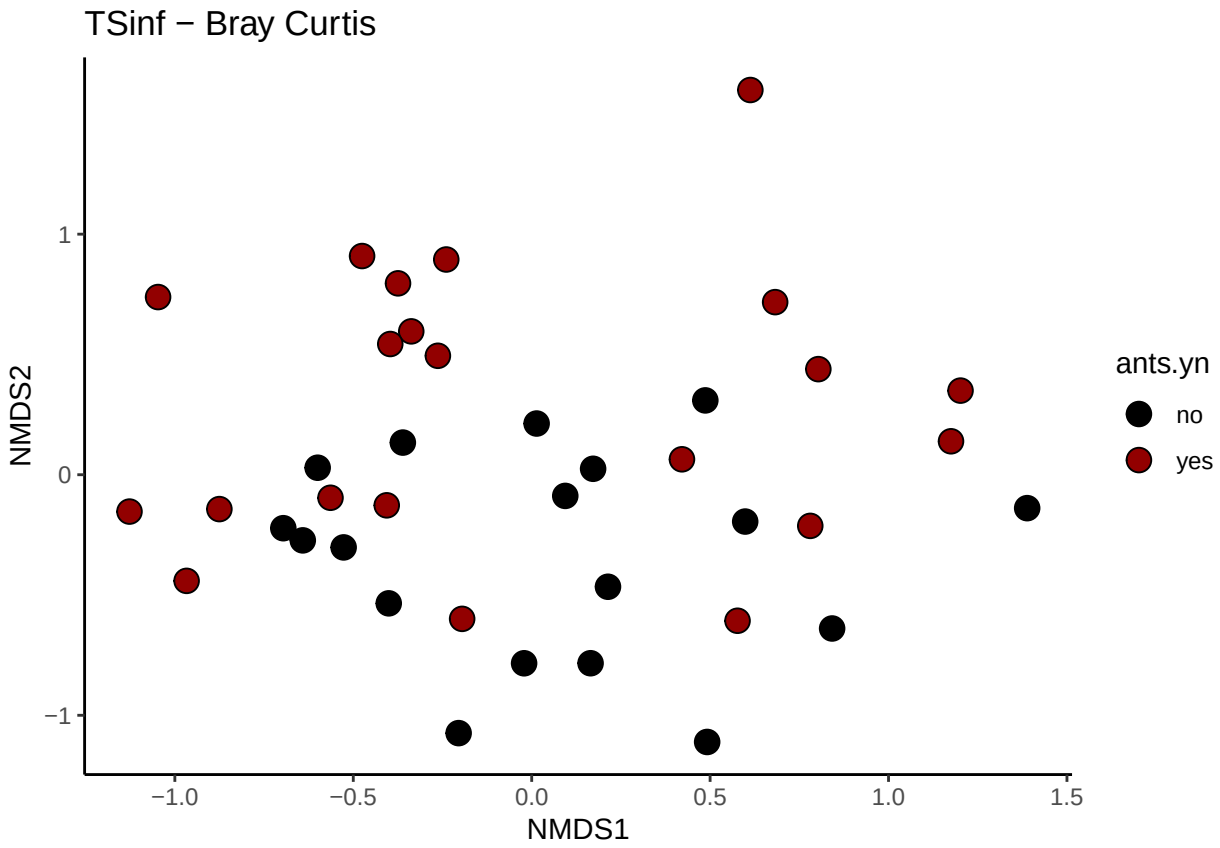
```
## Run 0 stress 0.2070791
## Run 1 stress 0.2395873
## Run 2 stress 0.2343102
## Run 3 stress 0.2177004
## Run 4 stress 0.2352884
## Run 5 stress 0.2168574
## Run 6 stress 0.2135854
## Run 7 stress 0.216397
## Run 8 stress 0.2732131
## Run 9 stress 0.2241094
## Run 10 stress 0.2070792
## ... Procrustes: rmse 3.213933e-05  max resid 9.767065e-05
## ... Similar to previous best
## Run 11 stress 0.2500355
## Run 12 stress 0.2381661
## Run 13 stress 0.2354778
## Run 14 stress 0.2070791
## ... New best solution
## ... Procrustes: rmse 1.552691e-05  max resid 4.669569e-05
## ... Similar to previous best
## Run 15 stress 0.2166057
## Run 16 stress 0.2250982
## Run 17 stress 0.2932066
```

```
## Run 18 stress 0.2133436
## Run 19 stress 0.2070791
## ... New best solution
## ... Procrustes: rmse 1.344092e-05  max resid 4.369792e-05
## ... Similar to previous best
## Run 20 stress 0.2431339
## *** Best solution repeated 1 times
```

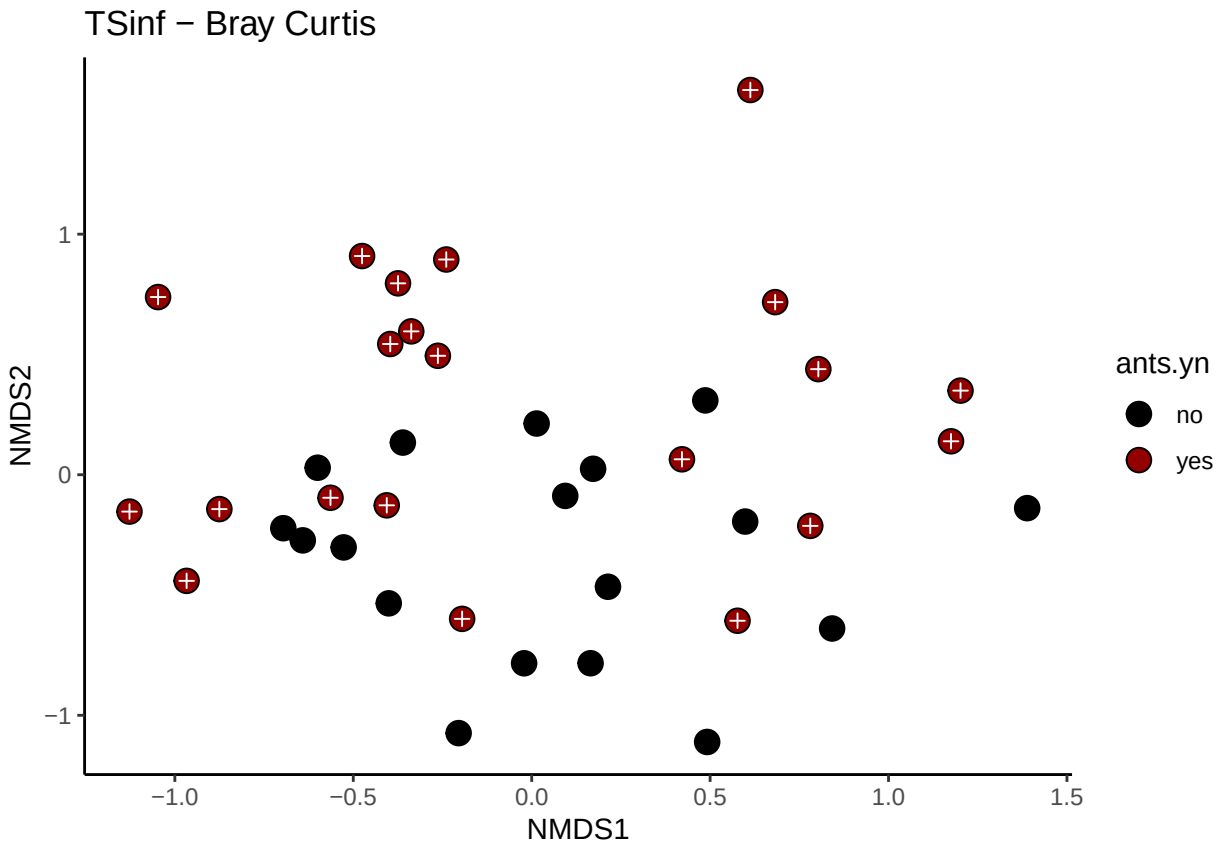
```
p <-
  plot_ordination( ps,
                   ps.bcord,
                   color = "black",
                   # shape = "s.type"
  ) +
  geom_point(
    aes(fill = ants.yn),
    size = 4,
    shape = 21
  ) +
  ggtitle("TSinf - Bray Curtis") +
  scale_fill_manual(
    values = ants.yn.col.ls
 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

```
## Warning in plot_ordination(ps, ps.bcord, color = "black", ): 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
)
```




```
ps.bcdist = distance(ps, method ="bray")
s.type.col.ls = list("fungus garden" = "#db6d00",
                     "trash" = "#924900" )
```

Permanova Test: sample type

```
# permanova
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   0.7941 0.06163 2.4301  0.011 *
## Residual 37  12.0902 0.93837
## Total    38  12.8843 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test: sample is infected?

```
# permanova
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   1.5169 0.11774 2.402  0.003 **
## Residual 36  11.3674 0.88226
## Total    38  12.8843 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test: treatment

```
# permanova
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   1.1812 0.09168 1.8168   0.02 *
## Residual 36  11.7031 0.90832
## Total    38  12.8843 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test: +/- ants

```
# permanova
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   0.6297 0.04887 1.9013  0.044 *
## Residual 37  12.2546 0.95113
## Total    38  12.8843 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test Interaction: treatment*ants

```
# permanova
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   2.5373 0.19693 1.6185  0.006 **
## Residual 33  10.3470 0.80307
## Total    38  12.8843 1.00000
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

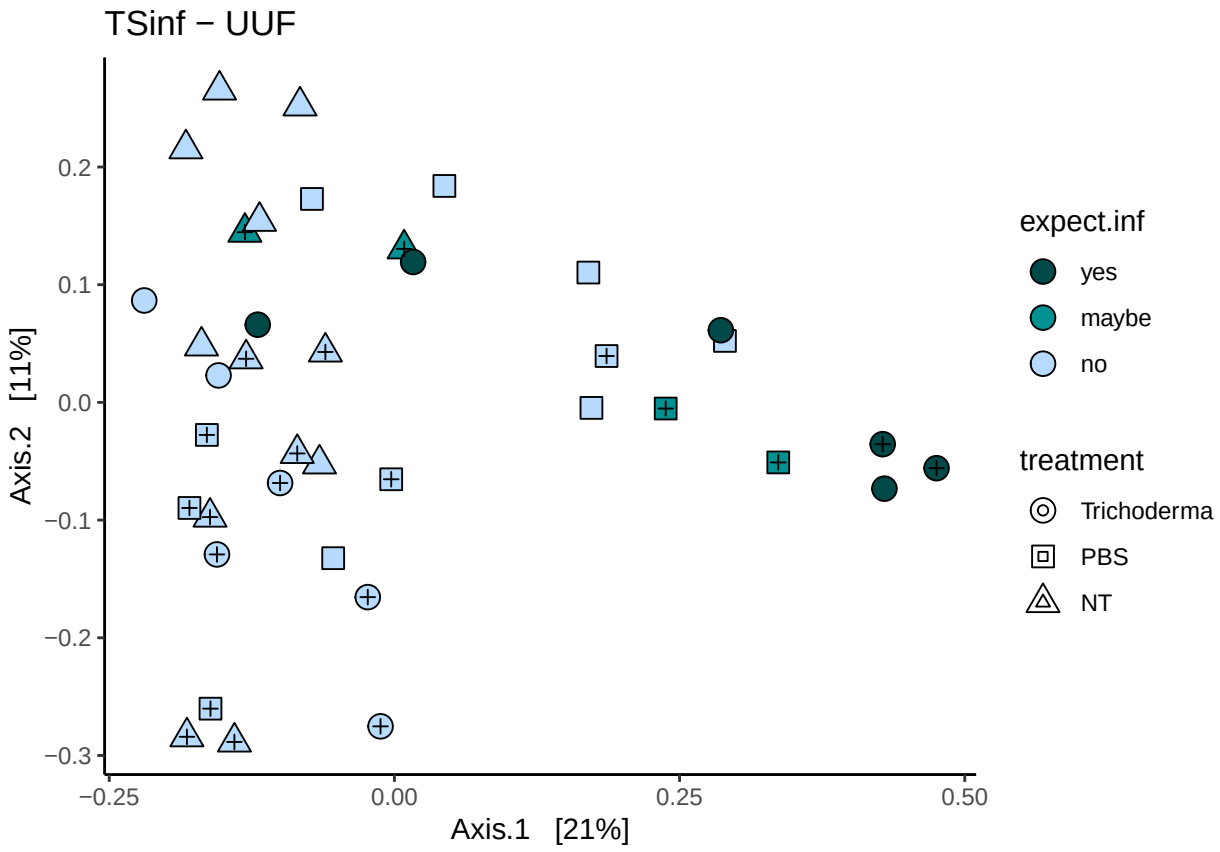
UUF - unweighted UNIFRAC

```
#Unweighted unifrac - s.type
ps.uuf.ord <- ordinate(ps, method = "PCoA", distance = "unifrac")

# plot: s.type
p <-
  plot_ordination( ps,
                   ps.uuf.ord,
                   color = "black",
                   shape = "treatment"
  ) +
  geom_point(
    aes(fill = expect.inf),
    size = 4
  ) +
  # geom_point(
  #   data = ants.y.bcord,
  #   aes()
  # ) +
  ggtitle("TSinf - UUF") +
  scale_fill_manual(
    values = exp.inf.col.ls
    # values = c("white", "lightblue", "darkgreen")
  ) +
  scale_shape_manual(
    values = c(21, 22, 24)
  ) +
  theme_classic() +
  guides(fill = guide_legend(override.aes = list(shape = 21)))
```

ord.exp.inf.treat.ants

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

```
# order legend
p$data$expect.inf = factor(p$data$expect.inf, levels = c("yes", "maybe", "no"))
p$data$treatment = factor(p$data$treatment, levels = c("Trichoderma", "PBS", "NT"))
# plot +ants on top
ants.y.uuford =
  p$data %>%
  filter(ants.yn == "yes")
p + geom_point(
  data = ants.y.uuford,
  aes(x=Axis.1, y=Axis.2),
  shape = 3,
  # stroke = 1.1
)
```

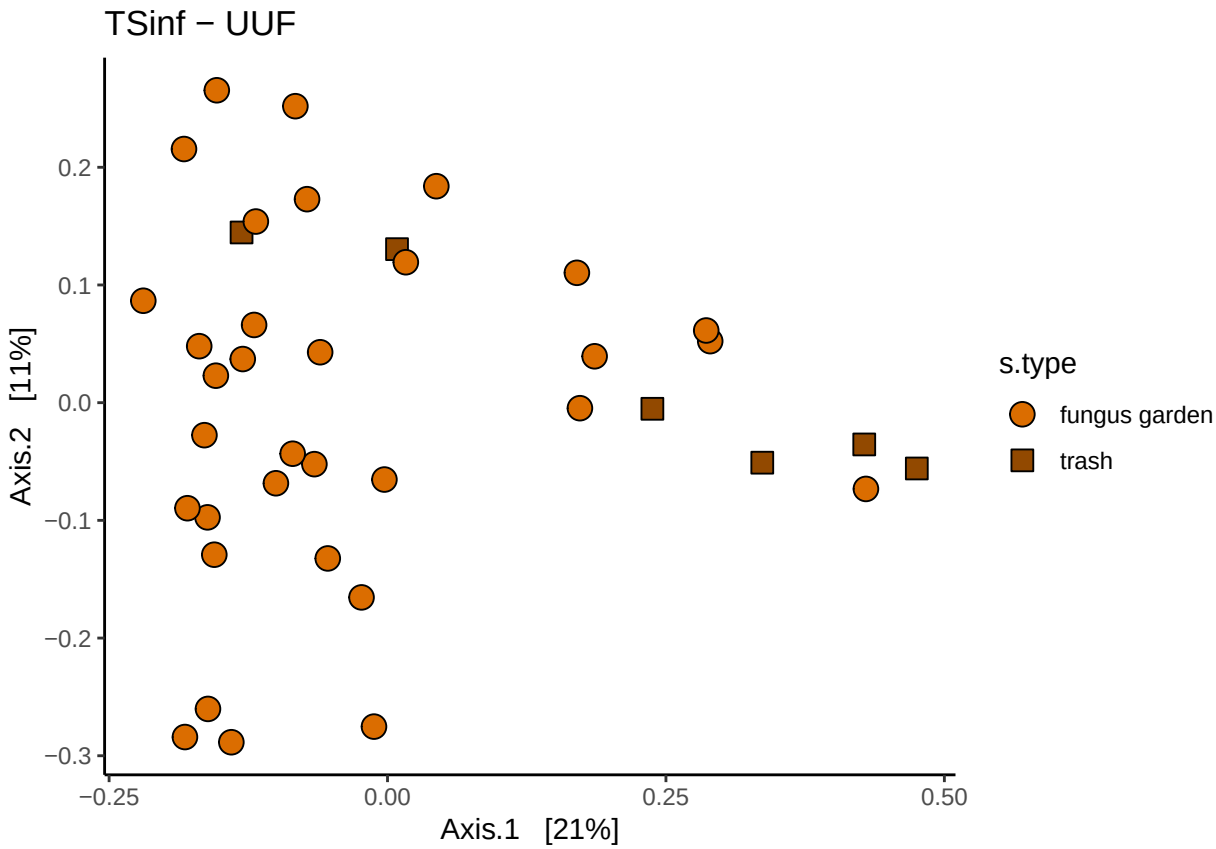



```
# Bray Curtis NMDS
ps.uuf.ord <- ordinate(ps, method = "PCoA", distance = "unifrac")

p <-
  plot_ordination( ps,
                   ps.uuf.ord,
                   color = "black",
                   shape = "s.type"
  ) +
  geom_point(
    aes(fill = s.type),
    size = 4,
    # alpha = 0.75
    # shape = 21
  ) +
  ggtitle("TSinf - UUF") +
  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))))
```

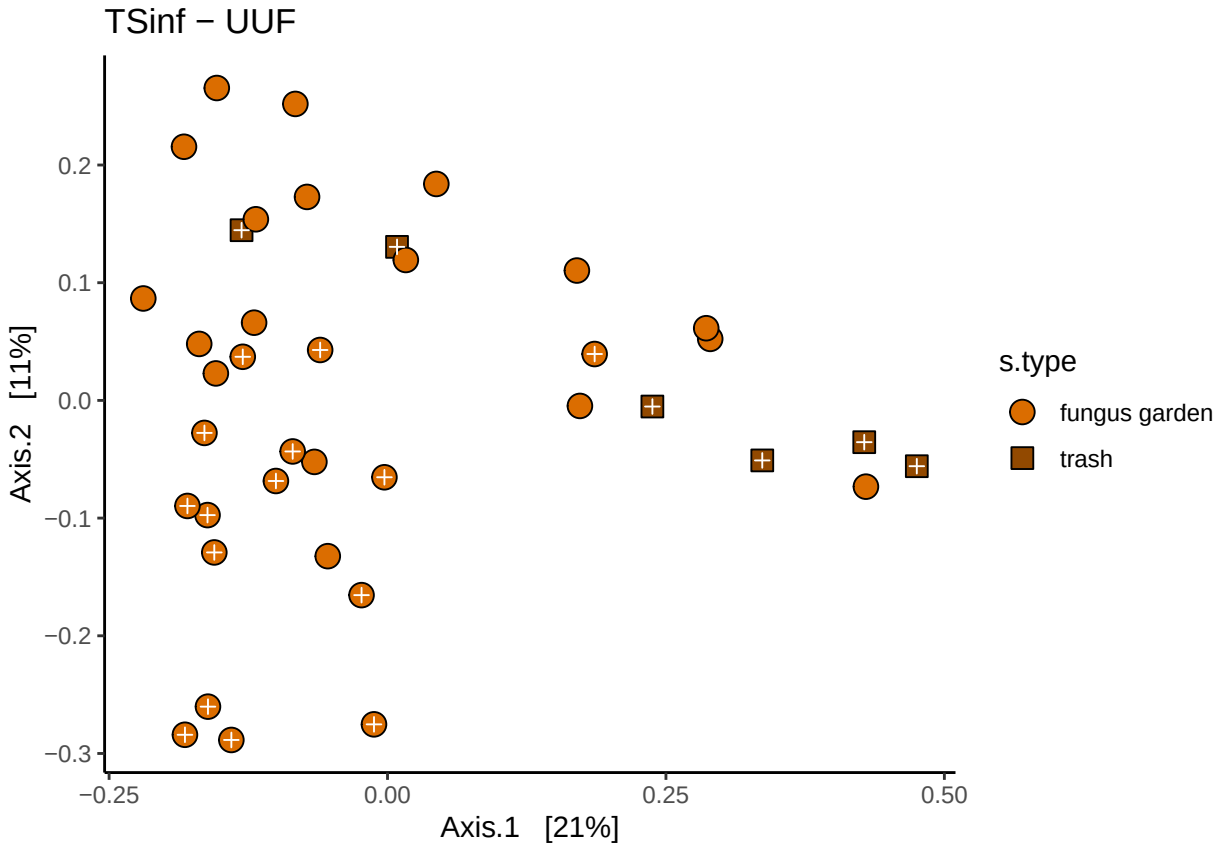
ord.s.type

```
## Warning in plot_ordination(ps, ps.uuf.ord, 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.uuford =
  p$data %>%
  filter(ants.yn == "yes")
p + geom_point(
  data = ants.y.uuford,
  aes(x=Axis.1, y=Axis.2),
  shape = 3,
  color = "white"
  # stroke = 1.1
)
```




```
ps.uuf.dist <- distance(ps, method = "unifrac")
```

Permanova Test: sample type

```
adonis2(ps.uuf.dist ~ s.type, ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.uuf.dist ~ s.type, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1   0.5016 0.07147 2.8479  0.001 ***
## Residual 37   6.5172 0.92853
## Total    38   7.0188 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test: expect sample to be infected?

```
adonis2(ps.uuf.dist ~ expect.inf, ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.uuf.dist ~ expect.inf, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
```

```
## Model     2   0.8899 0.12678 2.6134  0.002 **
## Residual 36   6.1290 0.87322
## Total    38   7.0188 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test: treatment

```
adonis2(ps.uuf.dist ~ treatment, ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.uuf.dist ~ treatment, data = ps.samdat.df)
##          Df SumOfSqs     R2     F Pr(>F)
## Model     2   0.6141 0.0875 1.726  0.013 *
## Residual 36   6.4047 0.9125
## Total    38   7.0188 1.0000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test: +/- ants

```
adonis2(ps.uuf.dist ~ ants.yn, ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.uuf.dist ~ ants.yn, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1   0.3519 0.05013 1.9528  0.017 *
## Residual 37   6.6670 0.94987
## Total    38   7.0188 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test Interaction: treatment*ants

```
adonis2(ps.uuf.dist ~ treatment*ants.yn, ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.uuf.dist ~ treatment * ants.yn, data = ps.samdat.df)
##          Df SumOfSqs      R2     F Pr(>F)
## Model     5   1.3026 0.18559 1.504  0.005 **
## Residual 33   5.7162 0.81441
## Total    38   7.0188 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

WUF - weighted UNIFRAC

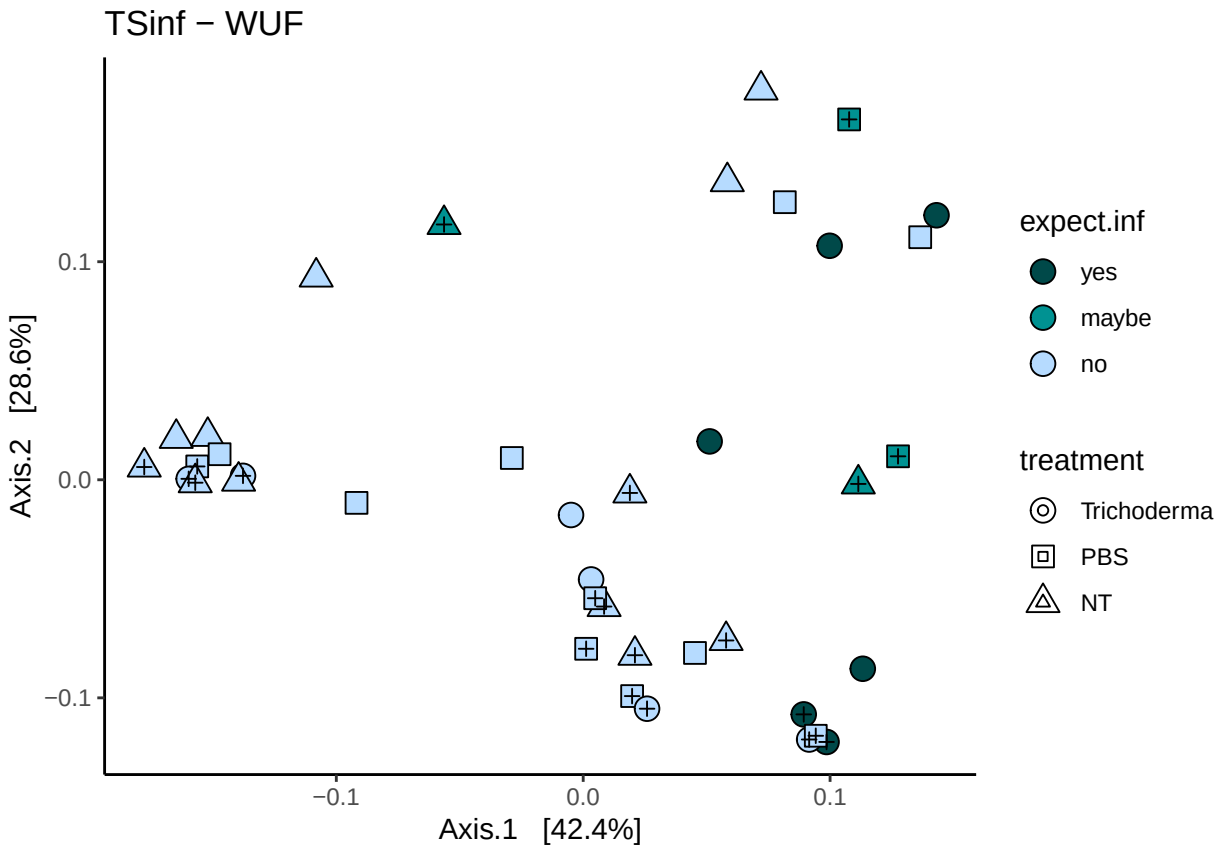
```
#weighted unifrac - s.type
```

```
ps.wuf.ord <- ordinate(ps, method = "PCoA", distance = "wunifrac")

# plot: s.type
p <-
  plot_ordination( ps,
                   ps.wuf.ord,
                   color = "black",
                   shape = "treatment"
  ) +
  geom_point(
    aes(fill = expect.inf),
    size = 4,
    # alpha = 0.8
  ) +
  # geom_point(
  #   data = ants.y.bcord,
  #   aes()
  # ) +
  ggtitle("TSinf - WUF") +
  scale_fill_manual(
    values = exp.inf.col.ls
    # values = c("white", "lightblue", "darkgreen")
  ) +
  scale_shape_manual(
    values = c(21, 22, 24)
  ) +
  theme_classic() +
  guides(fill = guide_legend(override.aes = list(shape = 21)))
```

ord.expinf.treat.ants

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

```
# order legend
p$data$expect.inf = factor(p$data$expect.inf, levels = c("yes", "maybe", "no"))
p$data$treatment = factor(p$data$treatment, levels = c("Trichoderma", "PBS", "NT"))
# plot +ants on top
ants.y.wuford =
  p$data %>%
  filter(ants.yn == "yes")
p + geom_point(
  data = ants.y.wuford,
  aes(x=Axis.1, y=Axis.2),
  shape = 3,
  # stroke = 1.1
)
```



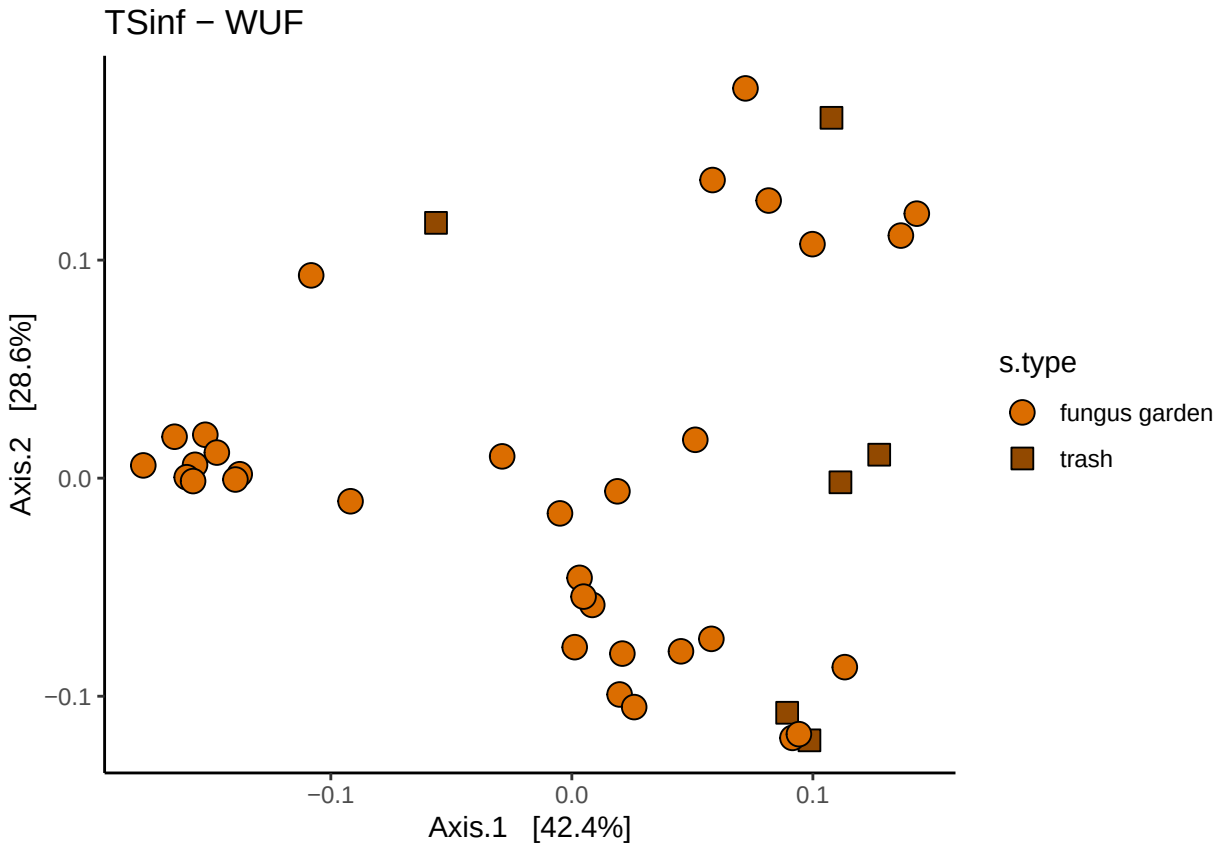

```
ps.wuf.ord <- ordinate(ps, method = "PCoA", distance = "wunifrac")

p <-
  plot_ordination( ps,
                   ps.wuf.ord,
                   color = "black",
                   shape = "s.type"
  ) +
  geom_point(
    aes(fill = s.type),
    size = 4,
    # shape = 21
  ) +
  ggtitle("TSinf - WUF") +
  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

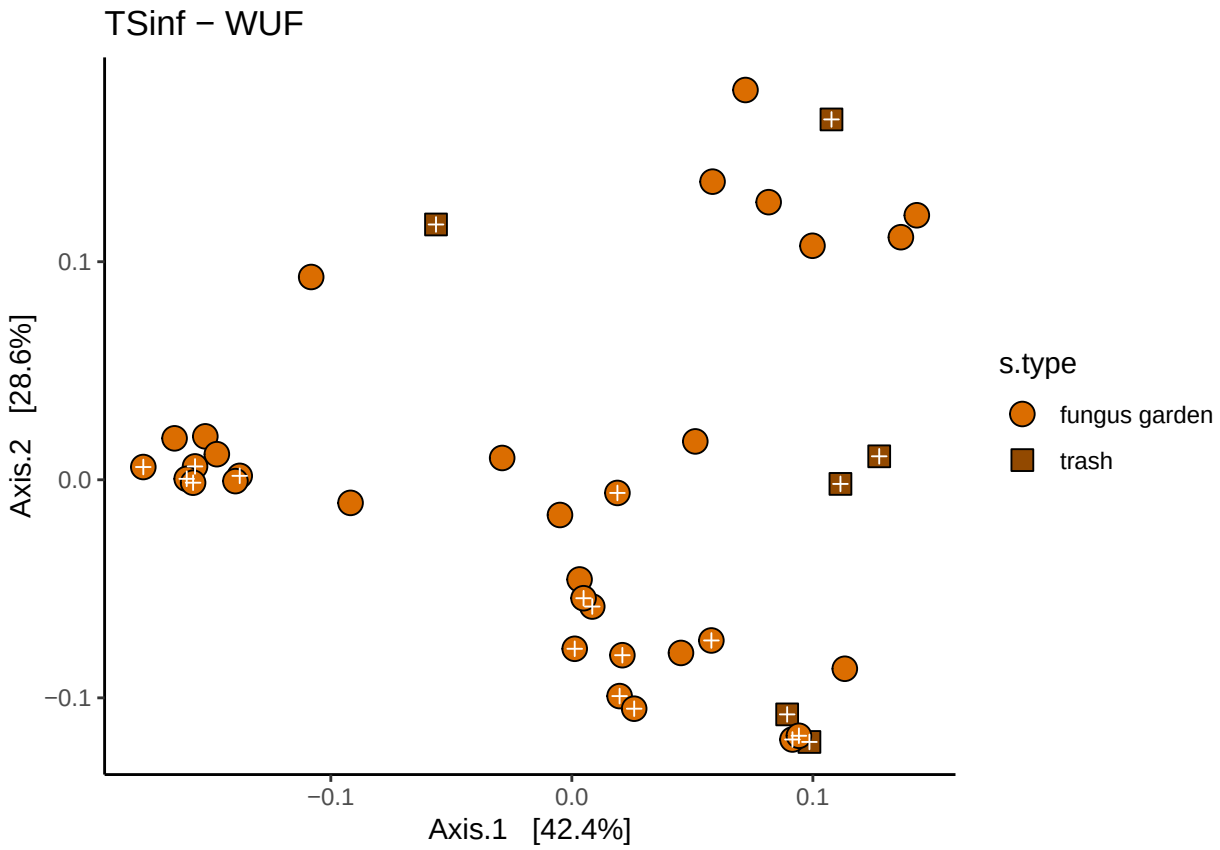
```
## Warning in plot_ordination(ps, ps.wuf.ord, color = "black", shape = "s.type"):
```

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



Test: sample type

```
ps.wuf.dist <- distance(ps, method = "wunifrac")
adonis2(ps.wuf.dist ~ s.type, ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.wuf.dist ~ s.type, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1  0.06059 0.06373 2.5184  0.043 *
## Residual 37  0.89023 0.93627
## Total    38  0.95083 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test: expect samples to be infected?

```
adonis2(ps.wuf.dist ~ expect.inf, ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.wuf.dist ~ expect.inf, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     2  0.15619 0.16427 3.5379  0.003 **
## Residual 36  0.79464 0.83573
```

```
## Total    38  0.95083 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test: treatment

```
adonis2(ps.wuf.dist ~ treatment, ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.wuf.dist ~ treatment, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     2  0.09149 0.09623 1.9165   0.06 .
## Residual 36  0.85933 0.90377
## Total    38  0.95083 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test: +/- ants

```
adonis2(ps.wuf.dist ~ ants.yn, ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.wuf.dist ~ ants.yn, data = ps.samdat.df)
##          Df SumOfSqs      R2      F Pr(>F)
## Model     1  0.06684 0.07029 2.7975   0.03 *
## Residual 37  0.88399 0.92971
## Total    38  0.95083 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Test Interaction: treatment*ants

```
adonis2(ps.wuf.dist ~ treatment*ants.yn, ps.samdat.df)
```

```
## Permutation test for adonis under reduced model
## Permutation: free
## Number of permutations: 999
##
## adonis2(formula = ps.wuf.dist ~ treatment * ants.yn, data = ps.samdat.df)
##          Df SumOfSqs      R2     F Pr(>F)
## Model     5  0.20755 0.21828 1.843  0.022 *
## Residual 33  0.74328 0.78172
## Total    38  0.95083 1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ps = psList.raref$decon4
inoc.dist.cm.df =
  sample_data(ps) %>%
  as_tibble() %>%
  select(inoc.dist.cm)
inoc.dist.cm.df = data.frame(inoc.dist.cm.df, row.names = sample_data(ps)$seq.id)
```

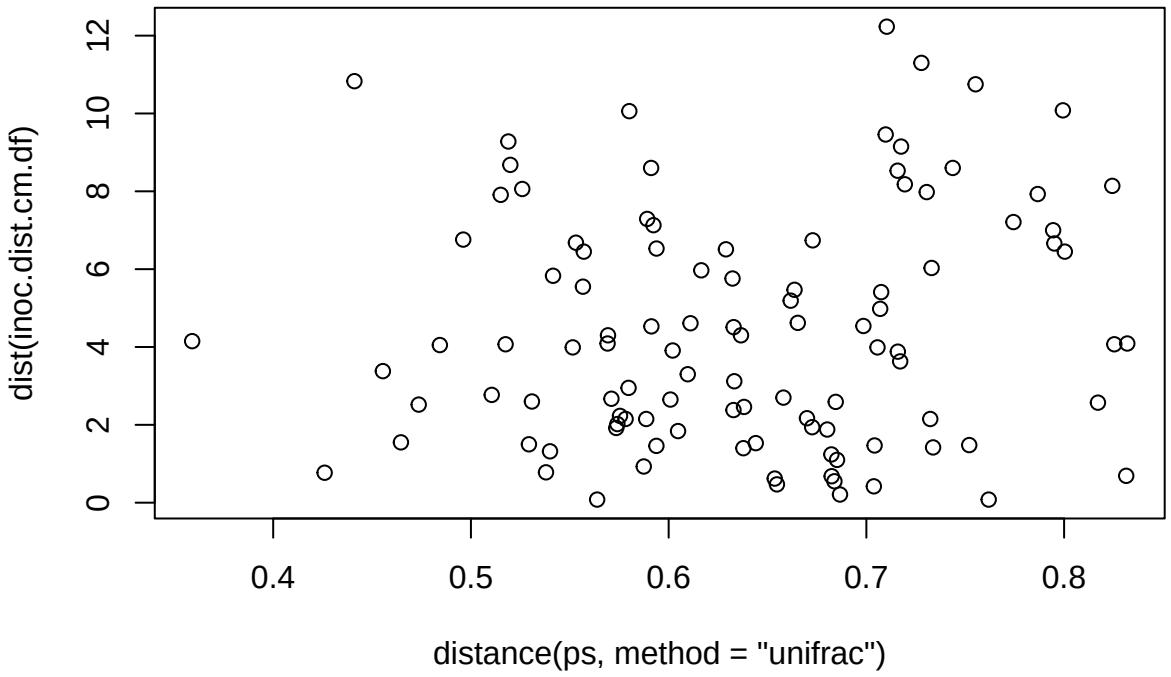
```
inoc.dist.cm.df
```

```
##                 inoc.dist.cm
## d4e03rcd10              2.15
## d4e04rcd10             10.75
## d4e06rcd10             12.23
## d4e09rcd10              6.76
## d4e10rcd10              8.18
## d4e13rcd10             10.06
## d4e14rcd10             10.83
## d4e15rcd10              8.60
## d4e16rcd10              9.28
## d4e18rcd10              6.68
## d4e19rcd10retry         4.09
## d4e22rcd10              2.77
## d4e25rcd10              0.00
## d4e28rcd10             11.30
## d4e29rcd10              4.30
```

```
mantel(
  distance(ps, method = "unifrac"),
  dist(inoc.dist.cm.df)
)
```

```
##
## Mantel statistic based on Pearson's product-moment correlation
##
## Call:
## mantel(xdis = distance(ps, method = "unifrac"), ydis = dist(inoc.dist.cm.df))
##
## Mantel statistic r: 0.1171
##       Significance: 0.184
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.177 0.230 0.262 0.315
## Permutation: free
## Number of permutations: 999
```

```
plot(
  x=distance(ps, method = "unifrac"),
  y=dist(inoc.dist.cm.df)
  )
```



Distance metrics tables

This is from untitled.R on new macbook desktop... Seems possibly useful but figure it out later.

```
# ## ---- distance metrics ----------------------------------------------------------------------------------------
# ps = psList.raref.rab$tsinf
#
# ps.filt.asvtab = as.data.frame(otu_table(ps.filt))
# #### ---- alpha diversity?
# ps.filt.diversity.df <-
#   as.data.frame( list(
#       shan = diversity(ps.filt.asvtab, index = "shannon", MARGIN = 1, base = exp(1)),
#       simp = diversity(ps.filt.asvtab, index = "simpson", MARGIN = 1),
#       isimp = diversity(ps.filt.asvtab, index = "invsimpson", MARGIN = 1),
#       fish = fisher.alpha(ps.filt.asvtab, MARGIN = 1),
#       specn = specnumber(ps.filt.asvtab, MARGIN = 1)
#   )   )
# write.table(ps.filt.diversity.df, "ps.filt.diversity.df",quote=F)
#
# ps.filt.distance.df <-
#   as.data.frame( list(
#       bc = distance(ps.filt, method="bray"),
#       uu = distance(ps.filt, method="uunifrac"),
#       wu = distance(ps.filt, method="wunifrac"),
#       jc = distance(ps.filt, method="jaccard", binary=T)
#   ))
#
#
# ordLs = list(
#   decons.bc.ords = ordinate(decons.ps.filt, method="NMDS", distance="bray"),
#   decons.uu.ords = ordinate(decons.ps.filt, method="PCoA", distance="uunifrac"),
#   decons.wu.ords = ordinate(decons.ps.filt, method="PCoA", distance="wunifrac"),
```

```
#   tsinf.bc.ords = ordinate(tsinf.ps.filt, method="NMDS", distance="bray"),
#   tsinf.uu.ords = ordinate(tsinf.ps.filt, method="PCoA", distance="uunifrac"),
#   tsinf.wu.ords = ordinate(tsinf.ps.filt, method="PCoA", distance="wunifrac")#,
#   #zymo.bc.ords = ordinate(zymo.ps.filt, method="NMDS", distance="bray"),
#   #zymo.uu.ords = ordinate(zymo.ps.filt, method="PCoA", distance="uunifrac"),
#   #zymo.wu.ords = ordinate(zymo.ps.filt, method="PCoA", distance="wunifrac"),
#
# )
```