

Beta diversity: distance-based Principal Response Curves (dbPRC)

Stijn Schreven

23 April 2022

Contents

Introduction	2
Load packages	2
Input files	3
1. dbPRC for the effect of Density on substrate microbiota	3
1.1. Subsets	3
1.2. Prepare data	3
1.3. Format for dbPRC	4
1.4. dbPRC	4
1.5. Statistical tests on density effect	8
2. dbPRC for the effect of Density on larval microbiota	11
2.1. Subsets	11
2.2. Prepare data	11
2.3. Format for dbPRC	12
2.4. dbPRC	12
2.5. Statistical tests on density effect	16
3. dbPRC for the effect of Sample type	19
3.1. Subsets	19
3.2. Prepare data	20
3.3. Format for dbPRC	20
3.4. dbPRC	20
3.5. Statistical tests on sample type effect	23

4. Pairwise dbPRC for the effect of Density on substrate microbiota	26
4.1. Subsets	27
4.2. Prepare data	27
4.3. Format for dbPRC	27
4.4. dbPRC function and default plot	28
4.5. Statistical test	29

Introduction

Constrained ordination with the temporal microbiota variation partialled out using a conditional argument in the constrained model of the dbRDA. The model output can be plotted as Principal Response Curves, showing only the microbiota variation explained by a treatment of interest relative to a baseline group. We produced weighted UniFrac distance-based Principal Response Curves, based on R code of Shankar *et al.* 2017. The analyses were done per diet separately based on relative abundance data at genus level.

Difference compared to Shankar *et al.* 2017:

- we use **dbrda** function (McArdle & Anderson 2001) instead of **capscale** (Legendre & Anderson 1999). This directly partitions the distance matrix.
- because we use **dbrda**, no species scores are calculated and these can be computed after **dbrda**, using **sppscores(dbrda.object) <- response**. In which **response** is the community matrix to calculate species scores.

Species selection threshold in PRC is based on R code of Eduard Szöcs, <http://edild.github.io/prc1/>, “Data in Environmental Science and Eco(toxico-)logy” - “Principal Response Curves with R (I)”, 1 December 2012.

Permutation tests and pairwise comparisons based on R code of Eduard Szöcs, <http://edild.github.io/prc2/>, “Data in Environmental Science and Eco(toxico-)logy” - “Principal Response Curves with R (II)”, 1 December 2012; with adaptations by Emmy van Daele (Laboratory of Microbiology, 24 September 2019), and some edits by the author (SS).

Alternative pairwise comparisons, *i.e.* pairwise Principal Response Curves (section for in this Markdown file), are based on Stam *et al.* (2016). R code made by the author (SS).

References: Stam JM, Dicke M, and Poelman EH (2016). Order of herbivore arrival on wild cabbage populations influences subsequent arthropod community development. In: Stam JM, *Plant-mediated insect interactions on a perennial plant: consequences for community dynamics*. PhD thesis, Wageningen University (pp. 71-105). <http://dx.doi.org/10.18174/386283>

Load packages

```
library(phyloseq)
library(microbiome)
library(microbiomeutilities)
library(dplyr)
library(vegan)
library(ggplot2)
library(ggvegan)
library(ggrepel)
```

```
library(ggpubr)
library(multcomp)
library(purrr)
library(knitr)
```

Input files

```
pstot <- readRDS("./phyobjects/ps1.exp.rds")

# genus, relative abundance
pstot.g <- aggregate_taxa(pstot, "Genus")
pstot.g.r <- microbiome::transform(pstot.g, "compositional")

# tax_table of total data, needed for genus names in PRC plot
tot.tax <- as.data.frame(tax_table(pstot.g.r))
tot.tax$OTU <- rownames(tot.tax)
tax_table(pstot.g.r) <- tax_table(as.matrix(tot.tax))
pstot.g.bh <- format_to_besthit(pstot.g.r)
tot.tax.bh <- as.data.frame(tax_table(pstot.g.bh))
colnames(tot.tax.bh)[7] <- "OTU"
```

1. dbPRC for the effect of Density on substrate microbiota

Supplementary Figure S2 in the manuscript of Chapter 3 in PhD thesis; Supplementary Figure S3 in the manuscript submitted to *Applied and Environmental Microbiology*. We investigated the effects of larval density on substrate microbiota composition and identified the genera with largest contribution to these effects.

1.1. Subsets

```
# per diet
CFs <- subset_samples(pstot.g.r, Diet == "CF" & Type == "substrate")
CFs <- prune_taxa(taxa_sums(otu_table(CFs)) > 0, CFs)
CSs <- subset_samples(pstot.g.r, Diet == "CS" & Type == "substrate")
CSs <- prune_taxa(taxa_sums(otu_table(CSs)) > 0, CSs)
CMs <- subset_samples(pstot.g.r, Diet == "CM" & Type == "substrate")
CMs <- prune_taxa(taxa_sums(otu_table(CMs)) > 0, CMs)
```

1.2. Prepare data

N.B.: change the phyloseq object (pseq) for diet (CFs, CSs, CMs) and run the downstream code chunks per diet.

```
# change the pseq dataset
pseq <- CMs
```

```
# generate input data formats
resp.s <- as.data.frame(t(abundances(pseq))) # response
env.s   <- meta(pseq)                      # metadata
s.wuf   <- distance(pseq, "wunifrac")      # distance matrix
```

1.3. Format for dbPRC

N.B.: `plot(prc)` is only working if treatment is called *treatment* and time is called *time* (the variables have to have these names)!

```
# otu table
response <- resp.s

# time
timef <- subset(env.s, select = "Timepoint")
time  <- as.factor(timef[,1])

# treatment
dens    <- subset(env.s, select = "Density")
treatment <- as.factor(dens[,1])

# Set up objects and matrices
y <- deparse(substitute(response))
x <- deparse(substitute(treatment))
z <- deparse(substitute(time))

# Create formulas and design matrices
fla    <- as.formula(paste("~", x, "+", z))
mf     <- model.frame(fla, response, na.action = na.pass)
fla.zx <- as.formula(paste("~", z, ":", x))
fla.z  <- as.formula(paste("~", z))
X      <- model.matrix(fla.zx, mf)[, -c(seq_len(nlevels(time)) + 1)]
Z      <- model.matrix(fla.z, mf)[, -1]
```

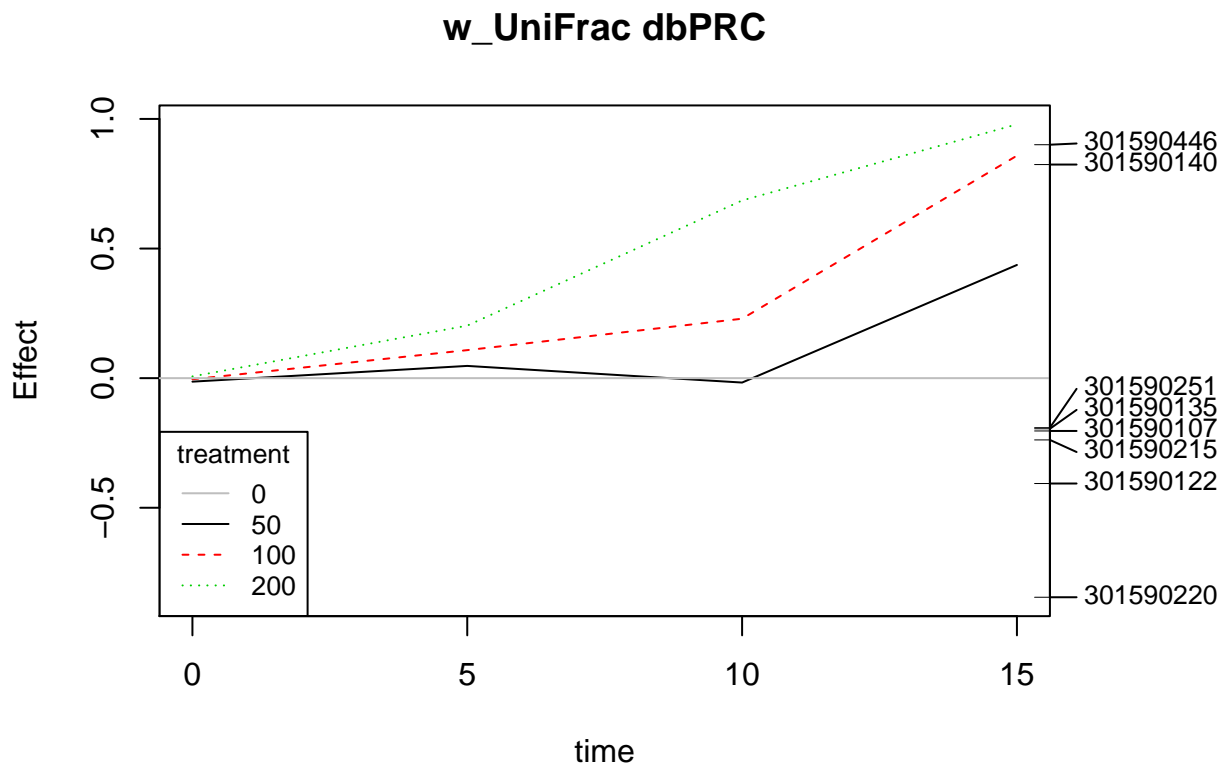
1.4. dbPRC

1.4.1. dbPRC function and default plot

```
# dbPRC
dbPRC.wuf <- dbrda(s.wuf ~ X + Condition(Z))
sppscores(dbPRC.wuf) <- response

# Conversion of dbRDA to PRC compatible format
dbPRC.wuf$terminfo$xlev <- list(levels(time), levels(treatment))
names(dbPRC.wuf$terminfo$xlev) <- c(paste(z), paste(x))
dbPRC.wuf$call <- match.call()
class(dbPRC.wuf) <- c("prc", class(dbPRC.wuf))

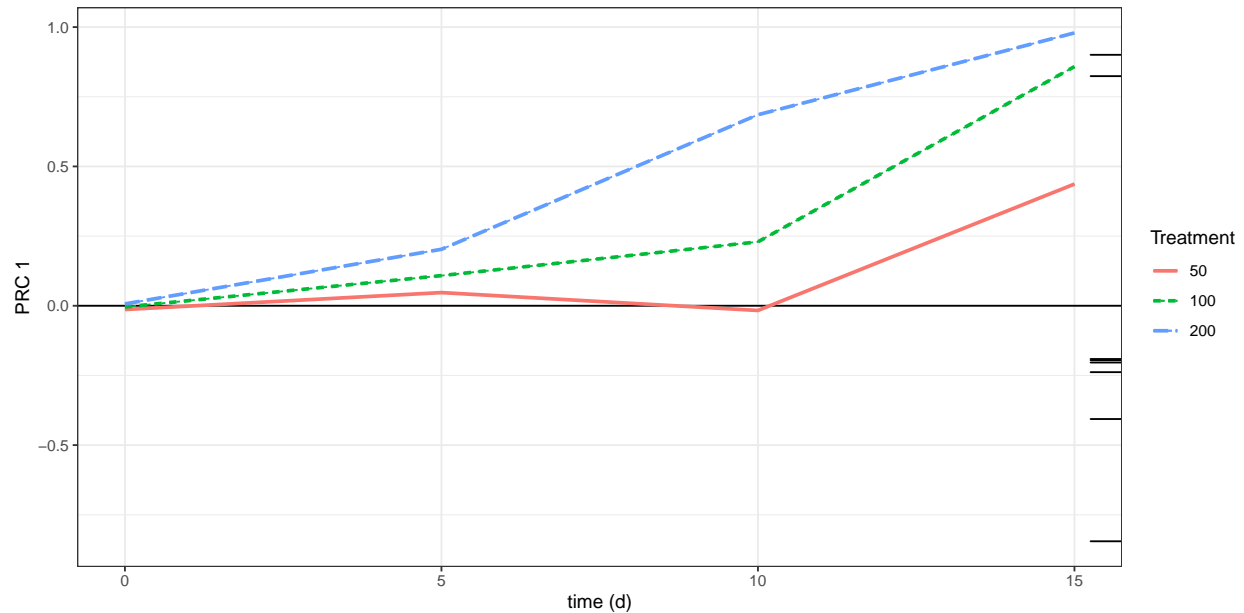
# plot
sum_dbprc2 <- summary(dbPRC.wuf)
plot(dbPRC.wuf, main = "w_UniFrac dbPRC", select = abs(sum_dbprc2$sp) > 0.25, scaling = 2)
```



1.4.2. dbPRC custom plot

Customizing the layout of the PRC plot using `ggvegan::autoplot.prc`.

```
pPRC <- autoplot(dbPRC.wuf, select = abs(sum_dbprc2$sp) > 0.25, scaling = 2) +  
  labs(x= "time (d)", y = "PRC 1") + geom_line(aes(linetype = Treatment), size = 1) +  
  theme_bw()  
pPRC
```



```
# extract data from ggplot object, including all species
prc.df <- autoplot(dbPRC.wuf, scaling = 2)$data      # treatment response
prc.sp <- autoplot(dbPRC.wuf, scaling = 2)$plot_env$spp # species scores
colnames(prc.sp)[2] <- "OTU"

# subset to top 10 species (abs(Response))
prc.sp.top <- top_n(prc.sp, n = 10, wt = abs(Response))
prc.sp.top$OTU <- droplevels(prc.sp.top$OTU)

# add tax levels to species scores
tax.prc <- subset(tot.tax.bh, OTU %in% prc.sp.top$OTU)

# remove pattern (OTU code) from $best_hit
tax.prc$best_hit <- sub(pattern = "OTU-[0-9]*:", replacement = "", tax.prc$best_hit)

# replace the "uncultured" with [Family]:uncultured
tax.prc$best_hit <- ifelse(tax.prc$best_hit == "uncultured",
  yes = paste("f__", tax.prc$Family, sep = "", "_uncultured"),
  no = paste(tax.prc$best_hit))

# add best_hit to species scores dataframe
prc.sp2 <- merge(prc.sp.top, tax.prc[,7:8], by = "OTU")

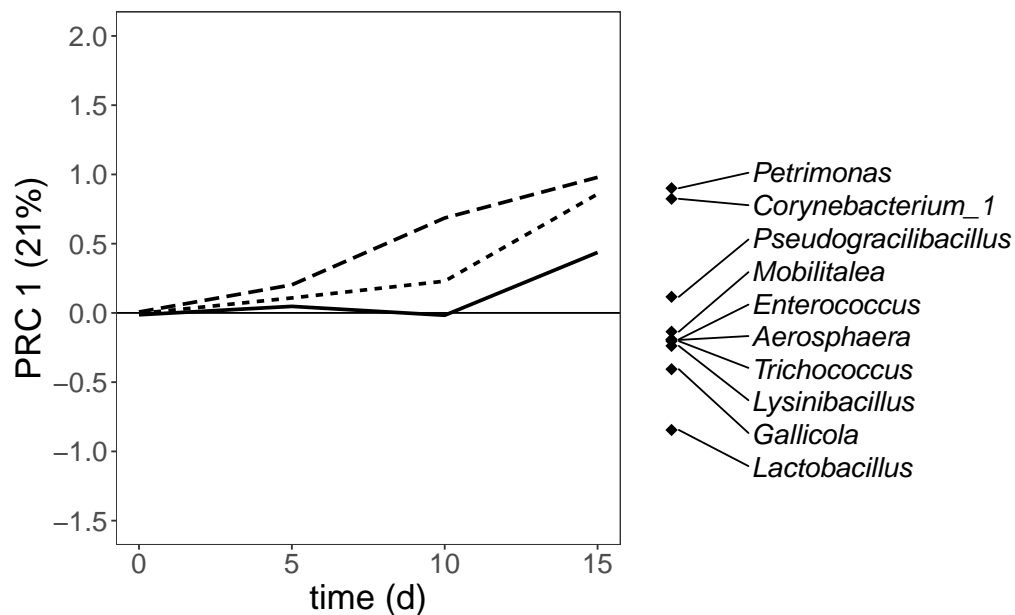
# plot with genus names
ax.prc1 <- round(RsquareAdj(dbPRC.wuf)$`r.squared`*100, 0)
pPRC2 <- ggplot(prc.df, aes(x = Time, y = Response)) +
  geom_line(aes(linetype = Treatment), size = 1) +
  labs(x = "time (d)", y = paste("PRC 1 (", ax.prc1, "%)", sep = ""),
    linetype = "Density") +
  geom_hline(yintercept = 0) +
  scale_y_continuous(limits = c(-1.5, 2), n.breaks = 7) +
  scale_x_continuous(limits = c(0, 15), n.breaks = 4) +
  theme_bw() + theme(text = element_text(size=20), panel.grid = element_blank())
```

```

# plot the species scores with the EXACT SAME y axis range
pPRC2b <- ggplot(prc.sp2, aes(y = Response, x = 1, label = best_hit)) +
  geom_point(shape = 18, size = 3) +
  scale_y_continuous(limits = c(-1.5,2)) +
  theme_classic() +
  theme(axis.line.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.x = element_blank(),
        axis.title.x = element_blank(),
        text = element_text(size = 25),
        axis.line.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        ) +
  xlim(1, 1.375) +
  geom_text_repel(aes(x = 1.005),
    nudge_x = .05,
    direction = "y",
    hjust = 0,
    size = 5.5,
    fontface = "italic")

pPRC3 <- ggarrange(pPRC2, pPRC2b, ncol = 2,
  legend = "none", align = "h", widths = c(1,1))
pPRC3

```



1.4.3. Export dbPRC custom plot

N.B.: Change the filename to the correct diet.

```
ggsave(plot = pPRC3, "./figures/dbPRC_CM_wuf.png", h = 5, w = 10)
ggsave(plot = pPRC3, "./figures/dbPRC_CM_wuf.pdf", h = 100, w = 200, u = "mm")
```

1.5. Statistical tests on density effect

1.5.1. Overall effect

```
# % variance explained
dbPRC.wuf$CCA$eig/sum(dbPRC.wuf$CCA$eig) # % explained by RDA axes

##          dbRDA1          dbRDA2          dbRDA3          dbRDA4          dbRDA5
## 0.6097970159 0.2720554767 0.0747356567 0.0287754612 0.0239088208
##          dbRDA6          dbRDA7          dbRDA8          dbRDA9          idbRDA1
## 0.0147261222 0.0023837802 0.0020890811 0.0001514094 -0.0032252957
##          idbRDA2          idbRDA3
## -0.0051057755 -0.0202917531
```

```
dbPRC.wuf$CCA$tot.chi/dbPRC.wuf$tot.chi # % explained by Density
```

```
## [1] 0.2094851
```

```
dbPRC.wuf$pCCA$tot.chi/dbPRC.wuf$tot.chi # % explained by Timepoint
```

```
## [1] 0.6277048
```

```
RsquareAdj(dbPRC.wuf)
```

```
## $r.squared
## [1] 0.2094851
##
## $adj.r.squared
## [1] 0.1772217
```

```
# Permutation Anova, stratified for containerID (repeated measures)
anova(dbPRC.wuf, first = T,
      permutations = how(within = Within(type = "none"),
                        plots = Plots(strata = env.s$ContainerID,
                                     type = "free"), nperm = 999))
```

```
## Permutation test for dbrda under reduced model
## Plots: env.s$ContainerID, plot permutation: free
## Permutation: none
## Number of permutations: 999
##
## Model: match.call()
##          Df SumOfSqs          F Pr(>F)
## dbRDA1    1  0.85945 37.662 0.001 ***
## Residual 48  1.09537
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```


1.5.2. Effect per timepoint

```
dbPRCs <- list()
for (i in levels(time)){

  #subset data, data from this timepoint
  take_dist <- as.matrix(s.wuf)[time == i,] # distance matrix
  take_treatm <- treatment[time == i]      # treatment

  #RDA: without time because looking at one timepoint
  dbPRCs[[i]]$rda <- dbrda(take_dist ~ take_treatm)

  #permutation test
  dbPRCs[[i]]$anova <- anova(dbPRCs[[i]]$rda, by = 'terms', permutations = 999)
}

length(dbPRCs) # n timepoints
```

```
## [1] 4
```

```
# extract info: anova per timepoint
dbPRCs[[1]]$anova
```

```
## Permutation test for dbrda under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: dbrda(formula = take_dist ~ take_treatm)
##           Df SumOfSqs      F Pr(>F)
## take_treatm 3  0.06890 0.3272  0.986
## Residual    12  0.84242
```

```
dbPRCs[[2]]$anova
```

```
## Permutation test for dbrda under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: dbrda(formula = take_dist ~ take_treatm)
##           Df SumOfSqs      F Pr(>F)
## take_treatm 3   2.4738 1.7977  0.138
## Residual    12   5.5046
```

```
dbPRCs[[3]]$anova
```

```
## Permutation test for dbrda under reduced model
## Terms added sequentially (first to last)
## Permutation: free
```

```
## Number of permutations: 999
##
## Model: dbrda(formula = take_dist ~ take_treatm)
##           Df SumOfSqs      F Pr(>F)
## take_treatm 3  11.0552 15.737 0.001 ***
## Residual    12   2.8101
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dbPRCs[[4]]$anova
```

```
## Permutation test for dbrda under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: dbrda(formula = take_dist ~ take_treatm)
##           Df SumOfSqs      F Pr(>F)
## take_treatm 3  12.6959 10.684 0.001 ***
## Residual    12   4.7534
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# extract p-value
# simply apply function on each value in dbRDAs
sapply(dbPRCs, function(x) x$anova[1,4])
```

```
##      0      5     10     15
## 0.986 0.138 0.001 0.001
```

1.5.3. Pairwise comparisons within each timepoint

```
# Test per timepoint, which treatments differ from control:
prc.df <- data.frame(density = treatment, time = time)

# Tukey contrasts (all-pair comparisons)
test.tukey <- NULL

# loop through time, compute PCoA, extract scores and do test
for (i in levels(time)) {

  take_dist  <- as.matrix(s.wuf)[time == i,time == i]
  take_treatm <- treatment[time == i]

  # Compute dbrda()
  dbrda <- dbrda(take_dist ~ 1) # direct dbRDA unconstrained.

  # scores of first principle component
  dbrda_scores <- scores(dbrda, display = "sites", choices = 1)
  test.tukey[[i]] <- summary(glht(aov(dbrda_scores ~ density,
                                     data = prc.df[time == i,])),
```

```

        alternative = "t",
        linfct = mcp(density = "Tukey"))))
}

# extract p-values
d.combins      <- combn(levels(prc.df$density), 2)
d.params       <- as.data.frame(t(d.combins))
d.params$comp  <- interaction(d.params$V1, d.params$V2, sep = ".vs.", drop = T)
result.tukey   <- lapply(test.tukey, function(x)
  data.frame(comp = d.params$comp, pval = x$test$pvalues))

# shows the results of Tukey test on PCoA-scores per day:
tukey.df       <- data.frame(map(.x = result.tukey, .f = "pval"))
rownames(tukey.df) <- d.params$comp
colnames(tukey.df) <- levels(time)

tukey.df[,1:4] <- round(tukey.df[,1:4], digits = 3)

kable(tukey.df)

```

	0	5	10	15
0.vs.50	1.000	0.772	0.519	0.105
0.vs.100	0.982	0.567	1.000	0.000
0.vs.200	0.983	0.344	0.043	0.000
50.vs.100	0.966	0.983	0.501	0.033
50.vs.200	0.993	0.861	0.004	0.005
100.vs.200	0.882	0.973	0.045	0.680

2. dbPRC for the effect of Density on larval microbiota

Supplementary Figure S4 in the manuscript submitted to *Applied and Environmental Microbiology*; not included in manuscript of Chapter 3 in PhD thesis. We investigated the effects of larval density on larval microbiota composition and identified the genera with largest contribution to these effects.

2.1. Subsets

```

# per diet
CF1 <- subset_samples(pstot.g.r, Diet == "CF" & Type == "larvae")
CF1 <- prune_taxa(taxa_sums(otu_table(CF1)) > 0, CF1)
CS1 <- subset_samples(pstot.g.r, Diet == "CS" & Type == "larvae")
CS1 <- prune_taxa(taxa_sums(otu_table(CS1)) > 0, CS1)
CM1 <- subset_samples(pstot.g.r, Diet == "CM" & Type == "larvae")
CM1 <- prune_taxa(taxa_sums(otu_table(CM1)) > 0, CM1)

```

2.2. Prepare data

N.B.: change the phyloseq object (pseq) for diet (CF1, CS1, CM1) and run the downstream code chunks per diet.

```

# change the pseq dataset
pseq <- CF1

# generate input data formats
resp.l <- as.data.frame(t(abundances(pseq))) # response
env.l   <- meta(pseq)                       # metadata
l.wuf   <- distance(pseq, "wunifrac")       # distance matrix

```

2.3. Format for dbPRC

N.B.: `plot(prc)` is only working if treatment is called *treatment* and time is called *time* (the variables have to have these names)!

```

# otu table
response <- resp.l

# time
timef <- subset(env.l, select = "Timepoint")
time  <- as.factor(timef[,1])

# treatment
dens    <- subset(env.l, select = "Density")
treatment <- as.factor(dens[,1])

# Set up objects and matrices
y <- deparse(substitute(response))
x <- deparse(substitute(treatment))
z <- deparse(substitute(time))

# Create formulas and design matrices
fla    <- as.formula(paste("~", x, "+", z))
mf     <- model.frame(fla, response, na.action = na.pass)
fla.zx <- as.formula(paste("~", z, ":", x))
fla.z  <- as.formula(paste("~", z))
X      <- model.matrix(fla.zx, mf)[, -c(seq_len(nlevels(time)) + 1)]
Z      <- model.matrix(fla.z, mf)[, -1]

```

2.4. dbPRC

2.4.1. dbPRC function and default plot

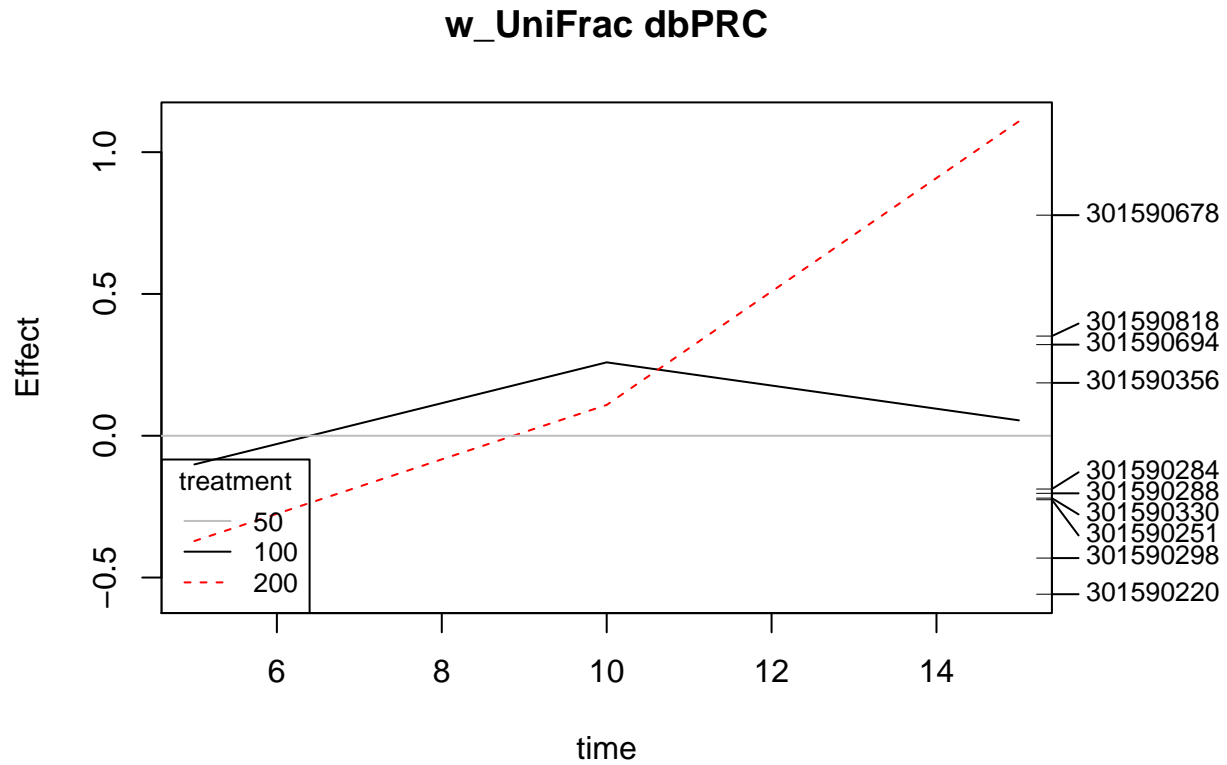
```

# dbPRC
dbPRC.wuf <- dbrda(l.wuf ~ X + Condition(Z))
sppscores(dbPRC.wuf) <- response

# Conversion of dbRDA to PRC compatible format
dbPRC.wuf$terminfo$xlev <- list(levels(time), levels(treatment))
names(dbPRC.wuf$terminfo$xlev) <- c(paste(z), paste(x))
dbPRC.wuf$call <- match.call()
class(dbPRC.wuf) <- c("prc", class(dbPRC.wuf))

```

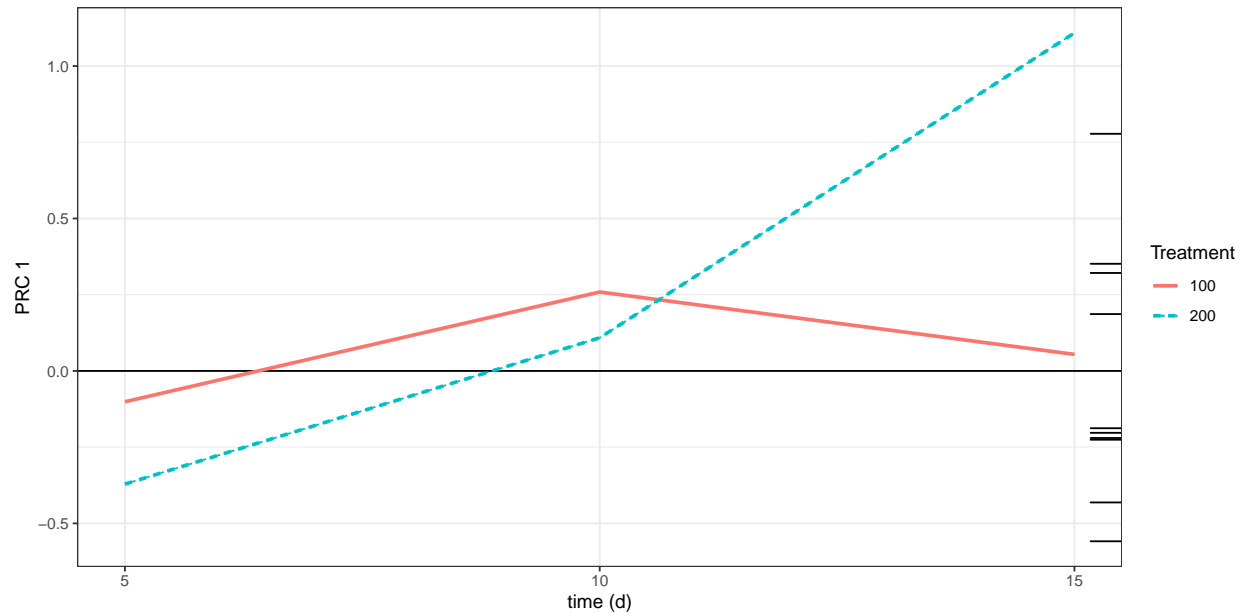
```
# plot
sum_dbprc2 <- summary(dbPRC.wuf)
plot(dbPRC.wuf, main = "w_UniFrac dbPRC", select = abs(sum_dbprc2$sp) > 0.25, scaling = 2)
```



2.4.2. dbPRC custom plot

Customizing the layout of the PRC plot using `ggvegan::autoplot.prc`.

```
pPRC <- autoplot(dbPRC.wuf, select = abs(sum_dbprc2$sp) > 0.25, scaling = 2) +
  labs(x = "time (d)", y = "PRC 1") + geom_line(aes(linetype = Treatment), size = 1) +
  theme_bw()
pPRC
```



```
# extract data from ggplot object, including all species
prc.df <- autoplot(dbPRC.wuf, scaling = 2)$data      # treatment response
prc.sp <- autoplot(dbPRC.wuf, scaling = 2)$plot_env$spp # species scores
colnames(prc.sp)[2] <- "OTU"

# subset to top 10 species (abs(Response))
prc.sp.top <- top_n(prc.sp, n = 10, wt = abs(Response))
prc.sp.top$OTU <- droplevels(prc.sp.top$OTU)

# add tax levels to species scores
tax.prc <- subset(tot.tax.bh, OTU %in% prc.sp.top$OTU)

# remove pattern (OTU code) from $best_hit
tax.prc$best_hit <- sub(pattern = "OTU-[0-9]*:", replacement = "", tax.prc$best_hit)

# replace the "uncultured" with [Family]:uncultured
tax.prc$best_hit <- ifelse(tax.prc$best_hit == "uncultured",
  yes = paste("f__", tax.prc$Family, sep = "", "_uncultured"),
  no = paste(tax.prc$best_hit))

# add best_hit to species scores dataframe
prc.sp2 <- merge(prc.sp.top, tax.prc[,7:8], by = "OTU")

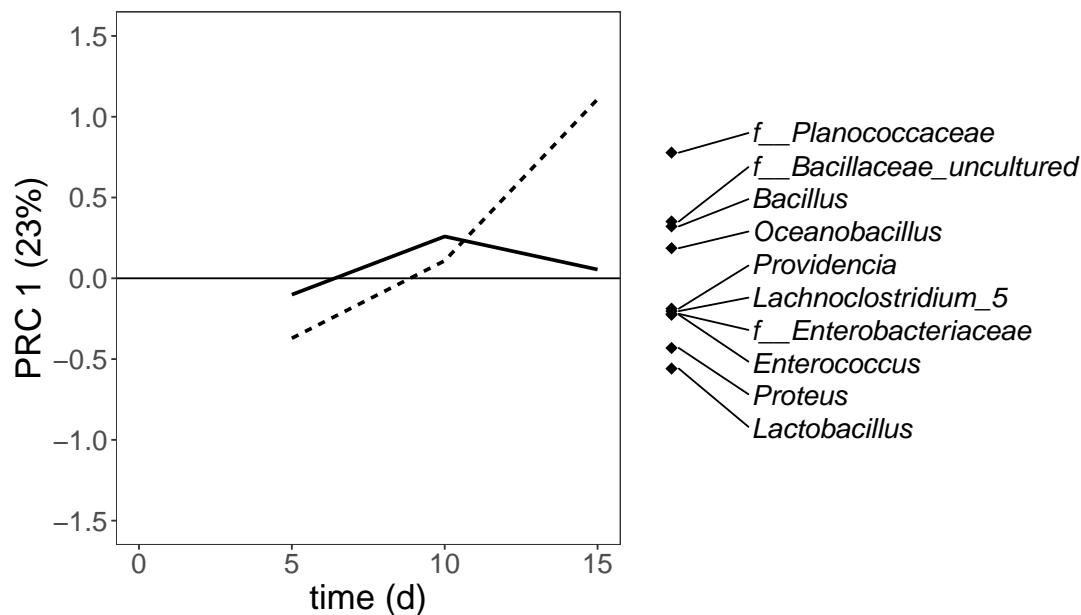
# plot with genus names
ax.prc1 <- round(RsquareAdj(dbPRC.wuf)$`r.squared`*100, 0)
pPRC2 <- ggplot(prc.df, aes(x = Time, y = Response)) +
  geom_line(aes(linetype = Treatment), size = 1) +
  labs(x = "time (d)", y = paste("PRC 1 (", ax.prc1, "%)", sep = ""),
    linetype = "Density") +
  geom_hline(yintercept = 0) +
  scale_y_continuous(limits = c(-1.5,1.5), n.breaks = 7) +
  scale_x_continuous(limits = c(0,15), n.breaks = 4) +
  theme_bw() + theme(text = element_text(size=20), panel.grid = element_blank())
```

```

# plot the species scores with the EXACT SAME y axis range
pPRC2b <- ggplot(prc.sp2, aes(y = Response, x = 1, label = best_hit)) +
  geom_point(shape = 18, size = 3) +
  scale_y_continuous(limits = c(-1.5,1.5)) +
  theme_classic() +
  theme(axis.line.x = element_blank(),
        axis.ticks.x = element_blank(),
        axis.text.x = element_blank(),
        axis.title.x = element_blank(),
        text = element_text(size = 25),
        axis.line.y = element_blank(),
        axis.ticks.y = element_blank(),
        axis.text.y = element_blank(),
        axis.title.y = element_blank(),
        ) +
  xlim(1, 1.375) +
  geom_text_repel(aes(x = 1.005),
    nudge_x = .05,
    direction = "y",
    hjust = 0,
    size = 5.5,
    fontface = "italic")

pPRC3 <- ggarrange(pPRC2, pPRC2b, ncol = 2,
  legend = "none", align = "h", widths = c(1,1))
pPRC3

```



2.4.3. Export dbPRC custom plot

N.B.: Change the filename to the correct diet.

```
ggsave(plot = pPRC3, "./figures/dbPRC_CFl_wuf.png", h = 5, w = 10)
ggsave(plot = pPRC3, "./figures/dbPRC_CFl_wuf.pdf", h = 100, w = 200, u = "mm")
```

2.5. Statistical tests on density effect

2.5.1. Overall effect

```
# % variance explained
dbPRC.wuf$CCA$eig/sum(dbPRC.wuf$CCA$eig) # % explained by RDA axes

##          dbRDA1          dbRDA2          dbRDA3          dbRDA4          dbRDA5          dbRDA6
## 0.593505504 0.212517841 0.097978099 0.064751371 0.029600761 0.001646423
```

```
dbPRC.wuf$CCA$tot.chi/dbPRC.wuf$tot.chi # % explained by Density
```

```
## [1] 0.2344953
```

```
dbPRC.wuf$pCCA$tot.chi/dbPRC.wuf$tot.chi # % explained by Timepoint
```

```
## [1] 0.370505
```

```
RsquareAdj(dbPRC.wuf)
```

```
## $r.squared
## [1] 0.2344953
##
## $adj.r.squared
## [1] 0.1556096
```

```
# Permutation Anova, stratified for containerID (repeated measures)
anova(dbPRC.wuf, first = T,
      permutations = how(within = Within(type = "none"),
                        plots = Plots(strata = env.l$ContainerID,
                                    type = "free"), nperm = 999))
```

```
## Permutation test for dbrda under reduced model
## Plots: env.l$ContainerID, plot permutation: free
## Permutation: none
## Number of permutations: 999
##
## Model: match.call()
##          Df SumOfSqs          F Pr(>F)
## dbRDA1    1  0.55961 9.5132  0.001 ***
## Residual 27  1.58827
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```


2.5.2. Effect per timepoint

```
dbPRCs <- list()
for (i in levels(time)){

  #subset data, data from this timepoint
  take_dist <- as.matrix(l.wuf)[time == i,] # distance matrix
  take_treatm <- treatment[time == i]      # treatment

  #RDA: without time because looking at one timepoint
  dbPRCs[[i]]$rda <- dbrda(take_dist ~ take_treatm)

  #permutation test
  dbPRCs[[i]]$anova <- anova(dbPRCs[[i]]$rda, by = 'terms', permutations = 999)
}

length(dbPRCs) # n timepoints
```

```
## [1] 3
```

```
# extract info: anova per timepoint
dbPRCs[[1]]$anova
```

```
## Permutation test for dbrda under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: dbrda(formula = take_dist ~ take_treatm)
##           Df SumOfSqs      F Pr(>F)
## take_treatm  2   1.5857 3.2965 0.046 *
## Residual     9   2.1646
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
dbPRCs[[2]]$anova
```

```
## Permutation test for dbrda under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: dbrda(formula = take_dist ~ take_treatm)
##           Df SumOfSqs      F Pr(>F)
## take_treatm  2   1.8902 1.8291 0.146
## Residual     9   4.6503
```

```
dbPRCs[[3]]$anova
```

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

```
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: dbrda(formula = take_dist ~ take_treatm)
##           Df SumOfSqs      F Pr(>F)
## take_treatm 2   3.8408 4.8138 0.006 **
## Residual    9   3.5904
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# extract p-value
# simply apply function on each value in dbRDAs
sapply(dbPRCs, function(x) x$anova[1,4])
```

```
##      5      10      15
## 0.046 0.146 0.006
```

2.5.3. Pairwise comparisons within each timepoint

```
# Test per timepoint, which treatments differ from control:
prc.df <- data.frame(density = treatment, time = time)

# Tukey contrasts (all-pair comparisons)
test.tukey <- NULL

# loop through time, compute PCoA, extract scores and do test
for (i in levels(time)) {

  take_dist  <- as.matrix(l.wuf)[time == i,time == i]
  take_treatm <- treatment[time == i]

  # Compute dbrda()
  dbrda <- dbrda(take_dist ~ 1) # direct dbRDA unconstrained.

  # scores of first principle component
  dbrda_scores <- scores(dbrda, display = "sites", choices = 1)
  test.tukey[[i]] <- summary(glht(aov(dbrda_scores ~ density,
                                     data = prc.df[time == i,]),
                               alternative = "t",
                               linfct = mcp(density = "Tukey"))))
}

# extract p-values
d.combins <- combn(levels(prc.df$density), 2)
d.params <- as.data.frame(t(d.combins))
d.params$comp <- interaction(d.params$V1, d.params$V2, sep = ".vs.", drop = T)
result.tukey <- lapply(test.tukey, function(x)
  data.frame(comp = d.params$comp, pval = x$test$pvalues))

# shows the results of Tukey test on PCoA-scores per day:
tukey.df <- data.frame(map(.x = result.tukey, .f = "pval"))
```

```
rownames(tukey.df) <- d.params$comp
colnames(tukey.df) <- levels(time)

tukey.df[,1:3] <- round(tukey.df[,1:3], digits = 3)

kable(tukey.df)
```

	5	10	15
50.vs.100	0.790	0.465	0.719
50.vs.200	0.049	0.962	0.000
100.vs.200	0.135	0.337	0.001

3. dbPRC for the effect of Sample type

Supplementary Figure S3 in the manuscript of Chapter 3 in PhD thesis; Supplementary Figure S5 in the manuscript submitted to *Applied and Environmental Microbiology*. We investigated the difference between larval and substrate microbiota and identified the genera with largest contribution to these differences. Set sample Type = “substrate” as the baseline (first level of a factor), “larvae” as treatment. Tested per diet x density combination separately.

3.1. Subsets

```
# subset to larva-substrate pairs
ls <- subset_samples(pstot.g.r, Timepoint != 0 & Density != 0)
ls <- prune_taxa(taxa_sums(otu_table(ls)) > 0, ls)
ls <- prune_samples(sample_sums(ls) > 0, ls)

# subset for diets
CF.ls <- subset_samples(ls, Diet == "CF")
CF.ls <- prune_taxa(taxa_sums(otu_table(CF.ls)) > 0, CF.ls)
CF.ls1 <- subset_samples(CF.ls, Density == "50")
CF.ls1 <- prune_taxa(taxa_sums(otu_table(CF.ls1)) > 0, CF.ls1)
CF.ls2 <- subset_samples(CF.ls, Density == "100")
CF.ls2 <- prune_taxa(taxa_sums(otu_table(CF.ls2)) > 0, CF.ls2)
CF.ls3 <- subset_samples(CF.ls, Density == "200")
CF.ls3 <- prune_taxa(taxa_sums(otu_table(CF.ls3)) > 0, CF.ls3)

CS.ls <- subset_samples(ls, Diet == "CS")
CS.ls <- prune_taxa(taxa_sums(otu_table(CS.ls)) > 0, CS.ls)
CS.ls1 <- subset_samples(CS.ls, Density == "50")
CS.ls1 <- prune_taxa(taxa_sums(otu_table(CS.ls1)) > 0, CS.ls1)
CS.ls2 <- subset_samples(CS.ls, Density == "100")
CS.ls2 <- prune_taxa(taxa_sums(otu_table(CS.ls2)) > 0, CS.ls2)
CS.ls3 <- subset_samples(CS.ls, Density == "200")
CS.ls3 <- prune_taxa(taxa_sums(otu_table(CS.ls3)) > 0, CS.ls3)

CM.ls <- subset_samples(ls, Diet == "CM")
CM.ls <- prune_taxa(taxa_sums(otu_table(CM.ls)) > 0, CM.ls)
```

```

CM.ls1 <- subset_samples(CM.ls, Density == "50")
CM.ls1 <- prune_taxa(taxa_sums(otu_table(CM.ls1)) > 0, CM.ls1)
CM.ls2 <- subset_samples(CM.ls, Density == "100")
CM.ls2 <- prune_taxa(taxa_sums(otu_table(CM.ls2)) > 0, CM.ls2)
CM.ls3 <- subset_samples(CM.ls, Density == "200")
CM.ls3 <- prune_taxa(taxa_sums(otu_table(CM.ls3)) > 0, CM.ls3)

```

3.2. Prepare data

N.B.: change the phyloseq object (pseq) for a diet x density subset (CF.ls1-3, CS.ls1-3, CM.ls1-3) and run the downstream code chunks per diet x density subset.

```

# change the pseq dataset
pseq <- CF.ls1

# generate input data formats
ls.gen <- as.data.frame(t(abundances(pseq))) # response
ls.env <- meta(pseq) # metadata
ls.wuf <- distance(pseq, "wunifrac") # distance matrix

```

3.3. Format for dbPRC

```

# otu table
response <- ls.gen
# time
timef <- subset(ls.env, select = "Timepoint")
time <- as.factor(timef[,1])

# treatment
type <- subset(ls.env, select = "Type")
treatment <- as.factor(type[,1])

# Set up objects and matrices
y <- deparse(substitute(response))
x <- deparse(substitute(treatment))
z <- deparse(substitute(time))

# Create formulas and design matrices
fla <- as.formula(paste("~", x, "+", z))
mf <- model.frame(fla, response, na.action = na.pass)
fla.zx <- as.formula(paste("~", z, ":", x))
fla.z <- as.formula(paste("~", z))
X <- model.matrix(fla.zx, mf)[, -c(seq_len(nlevels(time)) + 1)]
Z <- model.matrix(fla.z, mf)[, -1]

```

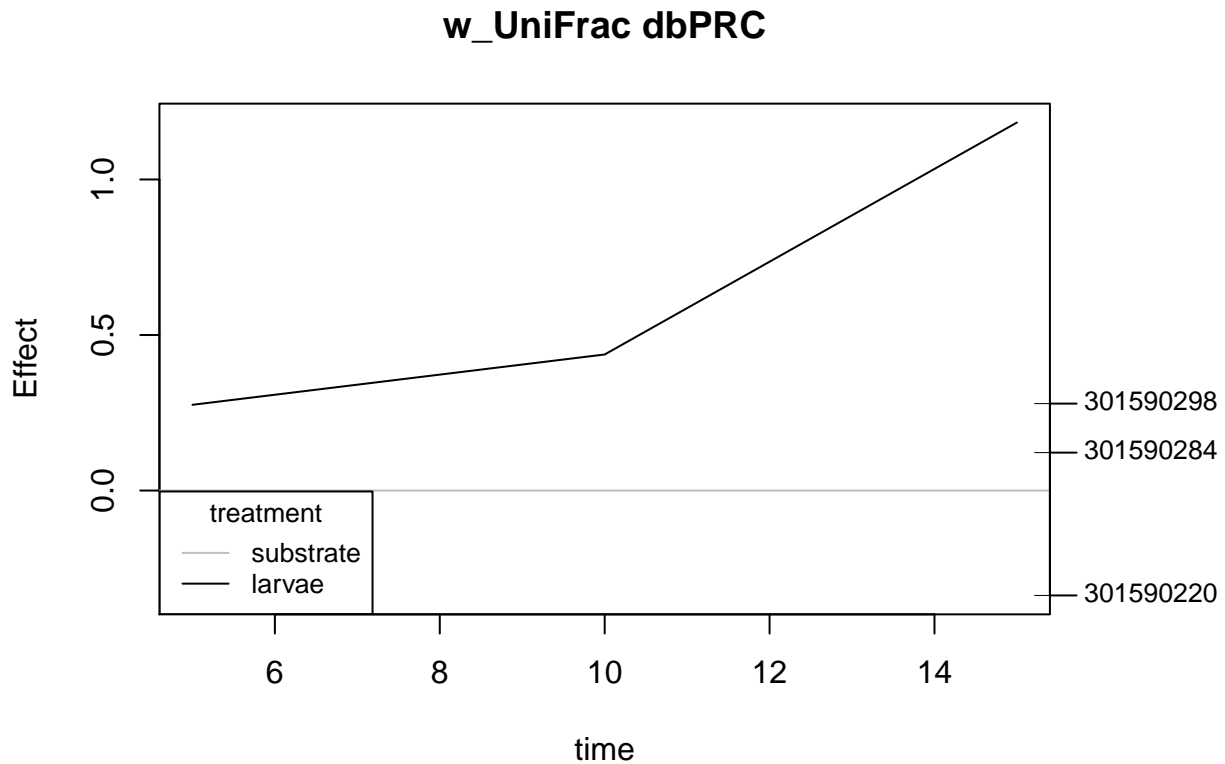
3.4. dbPRC

3.4.1. dbPRC function and default plot

```
# Calculation dbPRC
dbPRC.ls.wuf <- dbrda(ls.wuf ~ X + Condition(Z))
sppscores(dbPRC.ls.wuf) <- response

# Conversion of dbRDA to PRC compatible format
dbPRC.ls.wuf$terminfo$xlev <- list(levels(time), levels(treatment))
names(dbPRC.ls.wuf$terminfo$xlev) <- c(paste(z), paste(x))
dbPRC.ls.wuf$call <- match.call()
class(dbPRC.ls.wuf) <- c("prc", class(dbPRC.ls.wuf))

# plot
sum_dbprc3 <- summary(dbPRC.ls.wuf)
plot(dbPRC.ls.wuf, main = "w_UniFrac dbPRC", select = abs(sum_dbprc3$sp) > 0.25,
      scaling = 2)
```



3.4.2. dbPRC custom plot

Customizing the layout of the PRC plot using `ggvegan::autoplot.prc`.

```
# extract data from ggplot object, incl all species
prc.df <- autoplot(dbPRC.ls.wuf, scaling = 2)$data # treatment response
prc.sp <- autoplot(dbPRC.ls.wuf, scaling = 2)$plot_env$spp # species scores
```

```

colnames(prc.sp)[2] <- "OTU"

# subset to top 10 species (abs(Response))
prc.sp.top <- top_n(prc.sp, n = 10, wt = abs(Response))
prc.sp.top$OTU <- droplevels(prc.sp.top$OTU)

# add tax levels to species scores
tax.prc <- subset(tot.tax.bh, OTU %in% prc.sp.top$OTU)

# remove pattern (OTU code) from $best_hit
tax.prc$best_hit <- sub(pattern = "OTU-[0-9]*:", replacement = "", tax.prc$best_hit)

# replace the "uncultured" with [Family]:uncultured
tax.prc$best_hit <- ifelse(tax.prc$best_hit == "uncultured",
  yes = paste("f__", tax.prc$Family, sep = "", "_uncultured"),
  no = paste(tax.prc$best_hit))

# add best_hit to species scores dataframe
prc.sp2 <- merge(prc.sp.top, tax.prc[,7:8], by = "OTU")

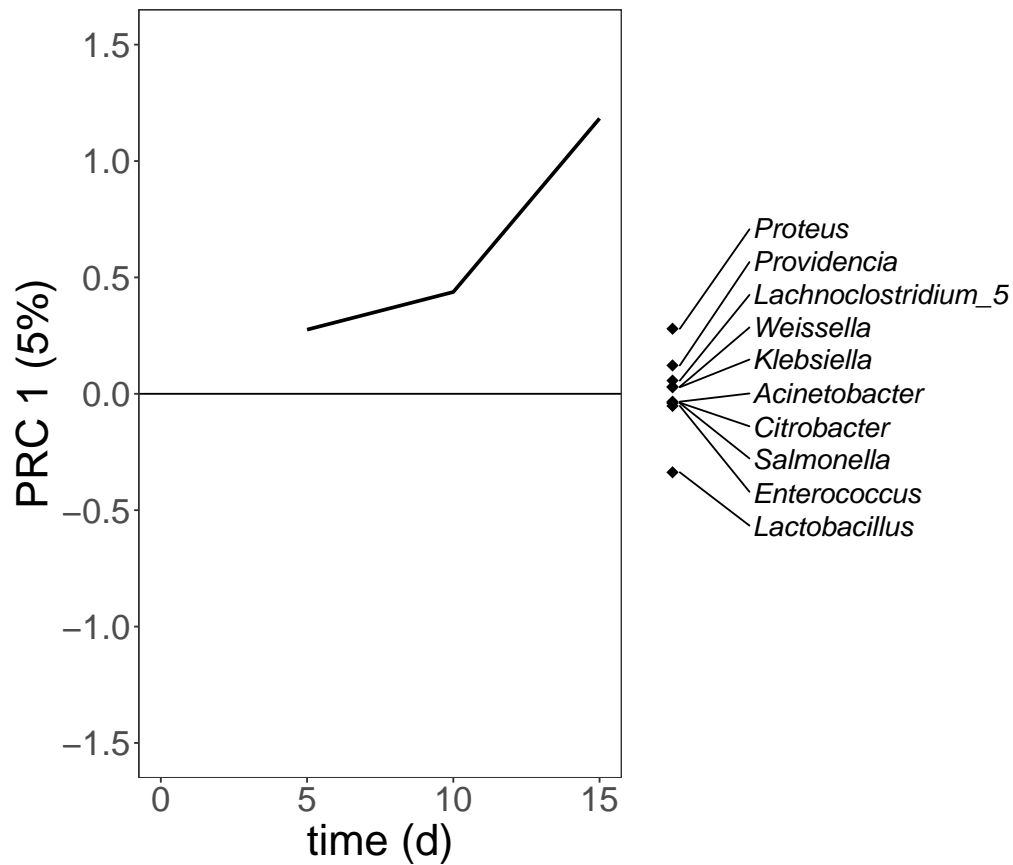
# plot with genus names
ax.ls.prc1 <- round(RsquareAdj(dbPRC.ls.wuf)$`r.squared`*100, 0)
pPRC2 <- ggplot(prc.df, aes(x = Time, y = Response)) +
  geom_line(size = 1) +
  labs(x = "time (d)", y = paste("PRC 1 (", ax.ls.prc1, "%)", sep = "")) +
  geom_hline(yintercept = 0) +
  scale_y_continuous(limits = c(-1.5,1.5), n.breaks = 7) +
  scale_x_continuous(limits = c(0,15), n.breaks = 4) +
  theme_bw() + theme(text = element_text(size=25), panel.grid = element_blank())

# plot the species scores with the same y axis range
pPRC2b <- ggplot(prc.sp2, aes(y = Response, x = 1, label = best_hit)) +
  geom_point(shape = 18, size = 3) +
  scale_y_continuous(limits = c(-1.5,1.5)) +
  theme_classic() +
  theme(axis.line.x = element_blank(),
    axis.ticks.x = element_blank(),
    axis.text.x = element_blank(),
    axis.title.x = element_blank(),
    text = element_text(size = 25),
    axis.line.y = element_blank(),
    axis.ticks.y = element_blank(),
    axis.text.y = element_blank(),
    axis.title.y = element_blank()) +
  xlim(1, 1.375) +
  geom_text_repel(aes(x = 1.005),
    nudge_x = .05,
    direction = "y",
    hjust = 0,
    size = 5.5,
    fontface = "italic")

pPRC3 <- ggarrange(pPRC2, pPRC2b, ncol = 2, align = "h", widths = c(1,1))

```

pPRC3



3.4.3. Export dbPRC custom plot

N.B.: change the filename to the correct diet x density combination.

```
ggsave(plot = pPRC3, "./figures/dbPRC_CF50_wuf_larv-sub.png", h = 7, w = 10)
ggsave(plot = pPRC3, "./figures/dbPRC_CF50_wuf_larv-sub.pdf", h = 140, w = 200, u = "mm")
```

3.5. Statistical tests on sample type effect

3.5.1. Overall effect

```
# % variance explained
dbPRC.ls.wuf$CCA$eig/sum(dbPRC.ls.wuf$CCA$eig) # % explained by RDA axes
```

```
##      dbRDA1      dbRDA2      dbRDA3
## 0.58074928 0.35055018 0.06870054
```

```
dbPRC.ls.wuf$CCA$tot.chi/dbPRC.ls.wuf$tot.chi # % explained by Type
```

```
## [1] 0.05180863
```

```
dbPRC.ls.wuf$pCCA$tot.chi/dbPRC.ls.wuf$tot.chi # % explained by Timepoint
```

```
## [1] 0.4969344
```

```
RsquareAdj(dbPRC.ls.wuf)
```

```
## $r.squared  
## [1] 0.05180863  
##  
## $adj.r.squared  
## [1] -0.02562952
```

```
# Permutation Anova, stratified for containerID (repeated measures)  
anova(dbPRC.ls.wuf, first = T,  
       permutations = how(within = Within(type = "none"),  
                           plots = Plots(strata = ls.env$ContainerID,  
                                           type = "free"), nperm = 999))
```

```
## 'nperm' >= set of all permutations: complete enumeration.
```

```
## Set of permutations < 'minperm'. Generating entire set.
```

```
## Permutation test for dbrda under reduced model  
## Plots: ls.env$ContainerID, plot permutation: free  
## Permutation: none  
## Number of permutations: 23  
##  
## Model: match.call()  
##           Df SumOfSqs      F Pr(>F)  
## dbRDA1    1  0.07066 1.2002 0.04167 *  
## Residual 18  1.05971  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.5.2. Effect per timepoint

```
dbPRCs <- list()  
for (i in levels(time)){  
  #subset data, data from this timepoint  
  take_dist <- as.matrix(ls.wuf)[time == i, time == i] # distance matrix  
  take_treatm <- treatment[time == i] # treatment  
  #RDA: without time because looking at one timepoint  
  dbPRCs[[i]]$rda <- dbrda(take_dist ~ take_treatm)  
  #permutation test
```



```
dbPRCs[[i]]$anova <- anova(dbPRCs[[i]]$rda, by = 'terms', permutations = 999)
}
```

```
length(dbPRCs) # n timepoints
```

```
## [1] 3
```

```
# extract info: anova per timepoint
dbPRCs[[1]]$anova
```

```
## Permutation test for dbrda under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: dbrda(formula = take_dist ~ take_treatm)
##           Df SumOfSqs      F Pr(>F)
## take_treatm 1 0.011744 0.9927 0.422
## Residual    6 0.070988
```

```
dbPRCs[[2]]$anova
```

```
## Permutation test for dbrda under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: dbrda(formula = take_dist ~ take_treatm)
##           Df SumOfSqs      F Pr(>F)
## take_treatm 1 0.04457 0.441 0.862
## Residual    6 0.60637
```

```
dbPRCs[[3]]$anova
```

```
## Permutation test for dbrda under reduced model
## Terms added sequentially (first to last)
## Permutation: free
## Number of permutations: 999
##
## Model: dbrda(formula = take_dist ~ take_treatm)
##           Df SumOfSqs      F Pr(>F)
## take_treatm 1 0.06536 1.0256 0.423
## Residual    6 0.38236
```

```
# extract p-value
# simply apply function on each value in dbPRCs
sapply(dbPRCs, function(x) x$anova[1,4])
```

```
##      5      10      15
## 0.422 0.862 0.423
```

3.5.3. Pairwise comparisons within each timepoint

```
# Test per timepoint, which treatments differ from control:
prc.df <- data.frame(type = treatment, time = time)

# Tukey contrasts (all-pair comparisons)
test.tukey2 <- NULL

# loop through time, compute PCoA, extract scores and do test
for (i in levels(time)) {

  take_dist <- as.matrix(ls.wuf)[time == i,time == i]
  take_treatm <- treatment[time == i]

  # Compute dbrda
  dbrda <- dbrda(take_dist ~ 1) # dbrda unconstrained.

  # scores of first principle component
  dbrda_scores <- scores(dbrda, display = "sites", choices = 1)
  test.tukey2[[i]] <- summary(glht(aov(dbrda_scores ~ type,
                                     data = prc.df[time == i,]),
                                alternative = "t",
                                linfct = mcp(type = "Tukey"))))
}

# extract p-values
d.combins <- combn(levels(prc.df$type), 2)
d.params <- as.data.frame(t(d.combins))
d.params$comp <- interaction(d.params$V1, d.params$V2, sep = ".vs.", drop = T)
result.tukey2 <- lapply(test.tukey2, function(x)
  data.frame(comp = d.params$comp, pval = x$test$pvalues))

# shows the results of Tukey test on PCoA-scores per day:
tukey.df2 <- data.frame(map(.x = result.tukey2, .f = "pval"))
rownames(tukey.df2) <- d.params$comp
colnames(tukey.df2) <- levels(time)

tukey.df2[,1:3] <- round(tukey.df2[,1:3], digits = 4)

kable(tukey.df2)
```

	5	10	15
substrate.vs.larvae	0.541	0.886	0.3095

4. Pairwise dbPRC for the effect of Density on substrate microbiota

Continuing on the analysis in section 1, this method makes pairwise comparisons of principal response curves, *i.e.* pairwise comparison of microbiota from two treatments (larval densities) across timepoints. Method from Stam *et al.* (2016). Input for this pairwise dbPRC is always a subset of the data concerning samples from two densities. P-values of statistical test are corrected for the number of comparisons (Bonferroni).

N.B.: This is the method used for the letters in Figure S2 in the manuscript of Chapter 3 in the PhD thesis (p.79). The figure caption mistakenly refers to the Tukey contrasts in paragraph 1.5.3 of this Markdown file as the explanation of the letters.

4.1. Subsets

Creating the subsets for pairwise contrasts. Some examples of pairwise contrasts given in the code chunk below. CMs14 is defined as the data of Chicken manure (CM) substrate microbiota (s) with larval density 0 (treatment 1) or 200 (treatment 4). Similarly, one can create the following subsets:

- Chicken feed: CFs12, CFs13, CFs14, CFs23, CFs24, CFs34;
- Camelina: CSs12, CSs13, CSs14, CSs23, CSs24, CSs34;
- Chicken manure: CMs12, CMs13, CMs14, CMs23, CMs24, CMs34.

```
# per diet and density pair
CMs14 <- subset_samples(CMs, Density %in% c(0,200))
CMs14 <- prune_taxa(taxa_sums(otu_table(CMs14)) > 0, CMs14)

CMs12 <- subset_samples(CMs, Density %in% c(0,50))
CMs12 <- prune_taxa(taxa_sums(otu_table(CMs12)) > 0, CMs12)

CMs24 <- subset_samples(CMs, Density %in% c(50,200))
CMs24 <- prune_taxa(taxa_sums(otu_table(CMs24)) > 0, CMs24)
```

4.2. Prepare data

N.B.: change the phyloseq object (pseq) for diet (CF112, CF113, etc) and run the downstream code chunks per diet.

```
# change the pseq dataset
pseq <- CMs24

# generate input data formats
resp.s <- as.data.frame(t(abundances(pseq))) # response
env.s <- meta(pseq) # metadata
s.wuf <- distance(pseq, "wunifrac") # distance matrix
```

4.3. Format for dbPRC

N.B.: `plot(prc)` is only working if treatment is called *treatment* and time is called *time* (the variables have to have these names)!

```
# otu table
response <- resp.s

# time
timef <- subset(env.s, select = "Timepoint")
time <- as.factor(timef[,1])

# treatment
```

```

dens      <- subset(env.s, select = "Density")
treatment <- as.factor(dens[,1])

# Set up objects and matrices
y <- deparse(substitute(response))
x <- deparse(substitute(treatment))
z <- deparse(substitute(time))

# Create formulas and design matrices
fla      <- as.formula(paste("~", x, "+", z))
mf       <- model.frame(fla, response, na.action = na.pass)
fla.zx   <- as.formula(paste("~", z, ":", x))
fla.z    <- as.formula(paste("~", z))
X        <- model.matrix(fla.zx, mf)[, -c(seq_len(nlevels(time)) + 1)]
Z        <- model.matrix(fla.z, mf)[, -1]

```

4.4. dbPRC function and default plot

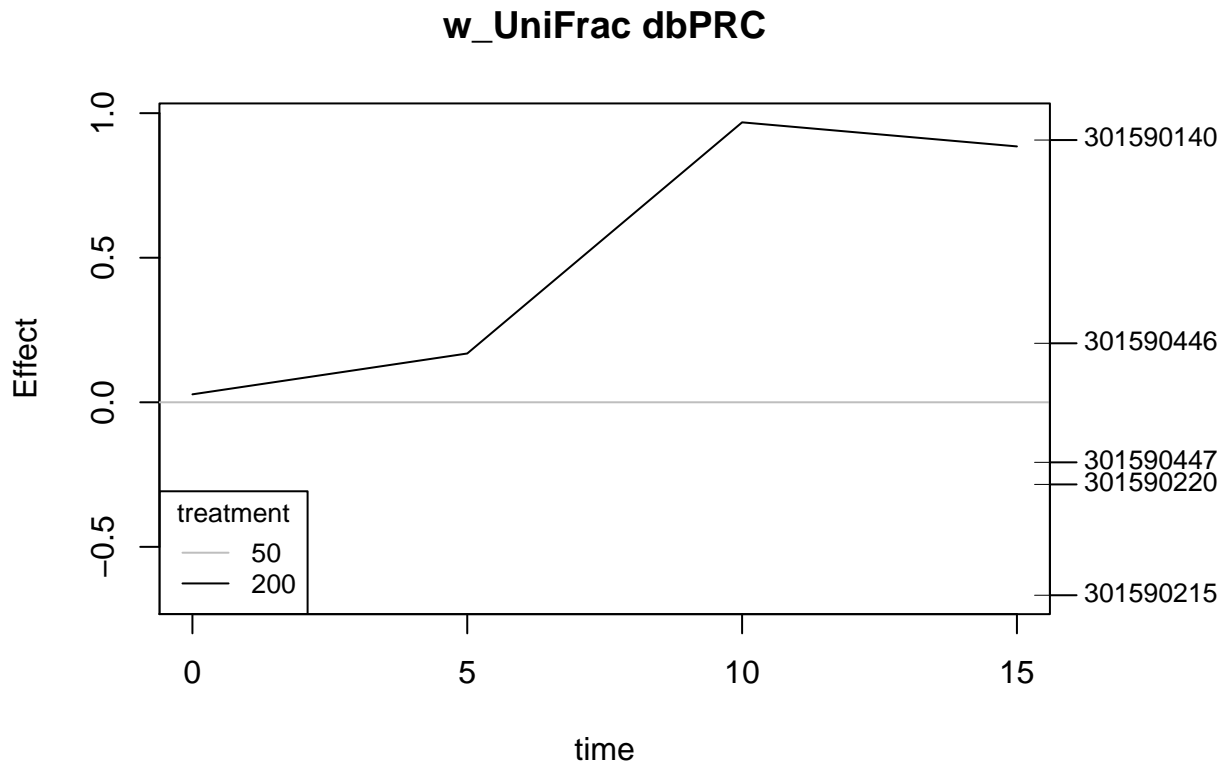
```

# dbPRC
dbPRC.wuf      <- dbrda(s.wuf ~ X + Condition(Z))
sppscores(dbPRC.wuf) <- response

# Conversion of dbRDA to PRC compatible format
dbPRC.wuf$terminfo$xlev      <- list(levels(time), levels(treatment))
names(dbPRC.wuf$terminfo$xlev) <- c(paste(z), paste(x))
dbPRC.wuf$call               <- match.call()
class(dbPRC.wuf)             <- c("prc", class(dbPRC.wuf))

# plot
sum_dbprc2 <- summary(dbPRC.wuf)
plot(dbPRC.wuf, main = "w_UniFrac dbPRC", select = abs(sum_dbprc2$sp) > 0.25, scaling = 2)

```



4.5. Statistical test

```
# Permutational Multivariate Anova, stratified for containerID (repeated measures)
anova(dbPRC.wuf, first = T,
      permutations = how(within = Within(type = "none"),
                        plots = Plots(strata = env.s$ContainerID,
                                    type = "free"), nperm = 999))
```

```
## Permutation test for dbrda under reduced model
## Plots: env.s$ContainerID, plot permutation: free
## Permutation: none
## Number of permutations: 999
##
## Model: match.call()
##      Df SumOfSqs      F Pr(>F)
## dbRDA1    1  0.4935 26.568 0.001 ***
## Residual 24  0.4458
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```