Analysis Illumina data

Heleen

8/6/2020

1. Pre blast

Load table and check read numbers

Functions:

```
getData <- function(){
  otu <- fread("~/Documents/derep_illum/changedheader/otu.about")
  before <- nrow(otu)
  # Remove X. or X from colnames
  names(otu) <- sub("#", "", names(otu))
  otu <- column_to_rownames(otu, var = "OTU ID")}


remove_chimeras <- function(){
  chimeras <- read.csv("~/Documents/IlluminaAdaptertrimmedAllreps/thingremoved", header=FALSE, sep=";")
  otu <- column_to_rownames(otu, var = "OTU.ID")
  otu <- otu[ ! sub("^.*?:", "", otu$OTU.ID) %in% chimeras$V1,] ##remove all chimeras
  after <- nrow(otu)
  cat(paste("removed", before-after, "chimeric sequences\n\n"))
  rownames(otu) <- NULL
  return(otu)}

# optional chimera removal: otu <- remove_chimeras()

# print results nicely:
myOTUcat <- function(){
  #total read sum in all clusters
  total_reads <- sum(rowSums(otu))
  cat(paste('total reads (grand total with which clustering was done):\n',
            total_reads))
  cat("\n\n", 'Summary statistics of number of reads per OTU:\n')
  print(summary(rowSums(otu)))
  cat(paste("\n\n",
            'Total number of OTUs (including singletons):\n', nrow(otu)))
}
```

Execution:

```
otu <- getData()
myOTUcat()
```

```
## total reads (grand total with which clustering was done):
##  2512237
##
##  Summary statistics of number of reads per OTU:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     1.00     1.00     1.00     8.46     2.00 86074.00
##
##
##  Total number of OTUs (including singletons):
##  296884
```

Prepare Saba location data

Functions:

```
#function to change decimal to comma in one
decimal_to_comma <- function(data, column){
  data[,column] <- sub(",", ".",
                      data[,column],
                      fixed = TRUE)}

prepLocSaba <- function(){
  ## load the Saba sample location data
  locdata_saba <- read.delim("~/Downloads/NICO5-eDNA-64PE432-Metadata-MinIon - DataFilterSaba.txt")

  ## Change samplenames, colnames in metadatafile so they match the OTU file making merging is possible
  ## Change decimal to comma for computation.
  locdata_saba[,1] <- gsub("(?<![0-9])0+", "", locdata_saba[,1], perl = TRUE)
  locdata_saba[,1] <- gsub("\\.", "_", locdata_saba[,1], perl = TRUE)
  locdata_saba[,1] <- tolower(locdata_saba[,1])

  ## change long colnams to lat,long, altitude
  names(locdata_saba)[names(locdata_saba)=="geo_lat..in.decimalen..WGS84."] <- "lat"
  names(locdata_saba)[names(locdata_saba)=="geo_lon..in.decimalen..WGS84."] <- "long"
  names(locdata_saba)[names(locdata_saba)=="altitude..in.meters.aasl."] <- "altitude"
  names(locdata_saba)[1] <- "sample"

  ## change decimals to commas
  for (col in c("lat", "long")){
    locdata_saba[, col] <- as.numeric(decimal_to_comma(locdata_saba, col))}
  return(locdata_saba)}
```

Execution:

```
#  establish samples and controls
controls <- c("sxm_2018_62", "sxm_2018_63", "sxm_2018_64",
              "sxm_2018_65", "sxm_2018_66", "sxm_2018_70",
              "sxm_2018_71", "0", "unicon1",
              "unicon1A", "neg_controle")
samples <- names(otu)[-which(names(otu) %in% controls)]

locdata_saba <- prepLocSaba()

# take only the columns needed, and the samplenames needed
locdata_saba <- locdata_saba %>%
  select(sample, lat, long, altitude, habitat) %>%
  filter(sample %in% samples)
```

load and prepare Statia data

select only relevant columns and rows, and setnames, and change the numbers to depth

```
# take relevant columns, take out the samples that are not in the OTU table, and set the colnames to sa
locdata_statia <- read.delim("~/statia_location.txt") %>%
  select(Field.nr., lat, long, Average.depth) %>%   # select relevant columns
  filter(!Field.nr. %in% c(528, 529)) %>% # discard irrelevant rows
  setNames(c("sample", "lat", "long", "altitude")) %>% # change column names
  mutate(altitude = as.numeric(gsub('[+]', '', altitude)) * -1) # mutate altidue column to negative
```

Bind Saba and statia data by row (to get merged data frame (mdf))

```
mdf <- plyr::rbind.fill(locdata_saba, locdata_statia)
```

boxcore altitude data is missing, so it's estimated by taking the nearest point geographically of which altitude data is available

```
#fill in missing boxcore altitude data wth nearestby latitude , the lowest value of that
mdf <- mdf %>%
  group_by(lat) %>%
  # arrange the groups by descending altitude within the groups
  arrange(desc(altitude), .by_group = TRUE) %>%
  # make new column with lowest altitude of group if the value is missing
  mutate(altitude = ifelse(is.na(altitude), min(altitude, na.rm = TRUE), altitude)) %>%
  # because for some boxcore samples it was taken at a slgihty different latitude, it does not belong t
  # Thus, R introduces infinite values which this command changes to NA values
  mutate(altitude = ifelse(is.infinite(altitude), NA, altitude)) %>%
  # needs to be ungrouped to fill it with the nearest & lowest altitude
  ungroup() %>%
  fill(altitude, .direction = 'down')
```

```
## Warning in min(altitude, na.rm = TRUE): no non-missing arguments to min;
## returning Inf

## Warning in min(altitude, na.rm = TRUE): no non-missing arguments to min;
## returning Inf
```

Put every sample in north, south or statia catogery based on latitude, to enable exchange testing. Add a tag indication what region the sampling location is in: Saba north, Saba south, or Statia.

```
mdf$tag <- ifelse(grepl("^[0-9]+$", mdf$sample), 'Statia',
                ifelse(mdf$lat > 17.55, "Saba North",
                       "Saba South")) %>% as.factor()
```

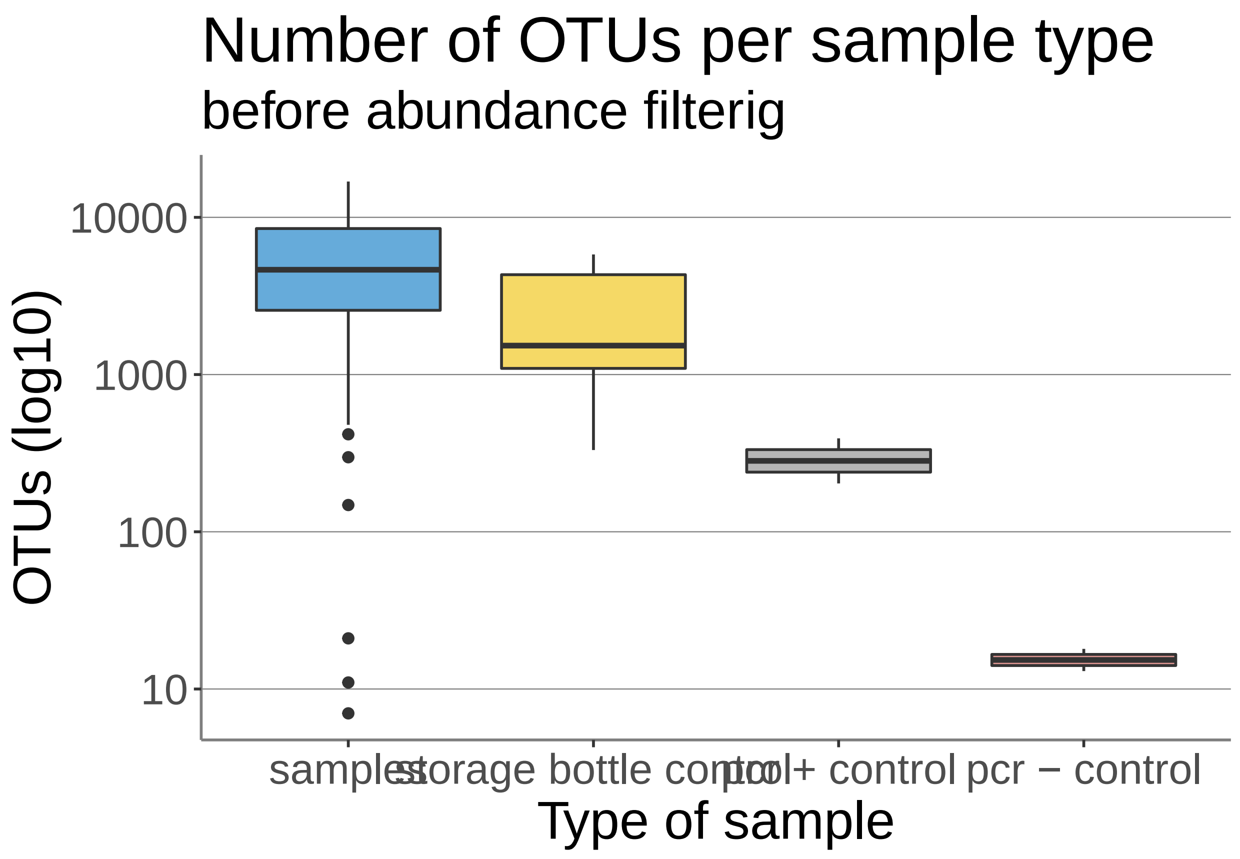
1.1

investigate control reads

```
#prep data
controlDf <- function(){
  copy <- otu
  copy[copy>0] <- 1
  per_type <- copy %>% colSums() %>%
    as.data.frame() %>% rownames_to_column(var = "sample")%>%
    mutate(type = ifelse(sample %in% bottlecontrol, "storage bottle control",
                         ifelse(grepl("unicon", sample), "pcr + control",
                                ifelse(sample %in% c("0", "neg_controle"),
                                       "pcr - control", "samples")))) %>%
    mutate(type = fct_reorder(type, desc(.)))
  return(per_type)}


plot_pertype <- function(df){
  control_plotOTU <-
    ggplot(df, aes(x = type,
                   y = .,
                   fill = type)) +
    geom_boxplot() +
    labs(title = "Number of OTUs per sample type",
         subtitle = "before abundance filterig",
       x = "Type of sample",
       y = "OTUs (log10)") +
    scale_fill_jco(alpha = 0.6) +
    scale_y_log10() +
  # edit lines and background
    theme(text = element_text(size = 20),
        panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_line("gray50", size = 0.2),
        panel.background = element_blank(),
        axis.line = element_line("gray50"),
        legend.position = "none")
  control_plotOTU}
```

##Execution:

```
controls <- c("sxm_2018_62", "sxm_2018_63", "sxm_2018_64",
              "sxm_2018_65", "sxm_2018_66", "sxm_2018_70",
              "sxm_2018_71", "0", "unicon1",
              "unicon1A", "neg_controle")
bottlecontrol <- c("sxm_2018_62", "sxm_2018_63", "sxm_2018_64",
              "sxm_2018_65", "sxm_2018_66", "sxm_2018_70",
              "sxm_2018_71")


controlDf <- controlDf()
plot_pertype(controlDf)
```



```
res.aov <- aov(d = controlDf, . ~ type)
summary(res.aov)
```

```
##             Df    Sum Sq  Mean Sq F value Pr(>F)
## type         3 1.763e+08 58764044   3.444 0.0198 *
## Residuals   96 1.638e+09 17060989
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
TukeyHSD(res.aov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = . ~ type, data = controlDf)
##
## $type
##                                          diff        lwr       upr     p adj
## storage bottle control-samples       -3137.205  -7376.563  1102.152 0.2204976
## pcr + control-samples                -5496.348 -13218.159  2225.463 0.2517563
## pcr - control-samples                -5778.848 -13500.659  1942.963 0.2117674
## pcr + control-storage bottle control -2359.143 -11018.102  6299.817 0.8919758
## pcr - control-storage bottle control -2641.643 -11300.602  6017.317 0.8553299
## pcr - control-pcr + control           -282.500 -11082.120 10517.120 0.9998844
```

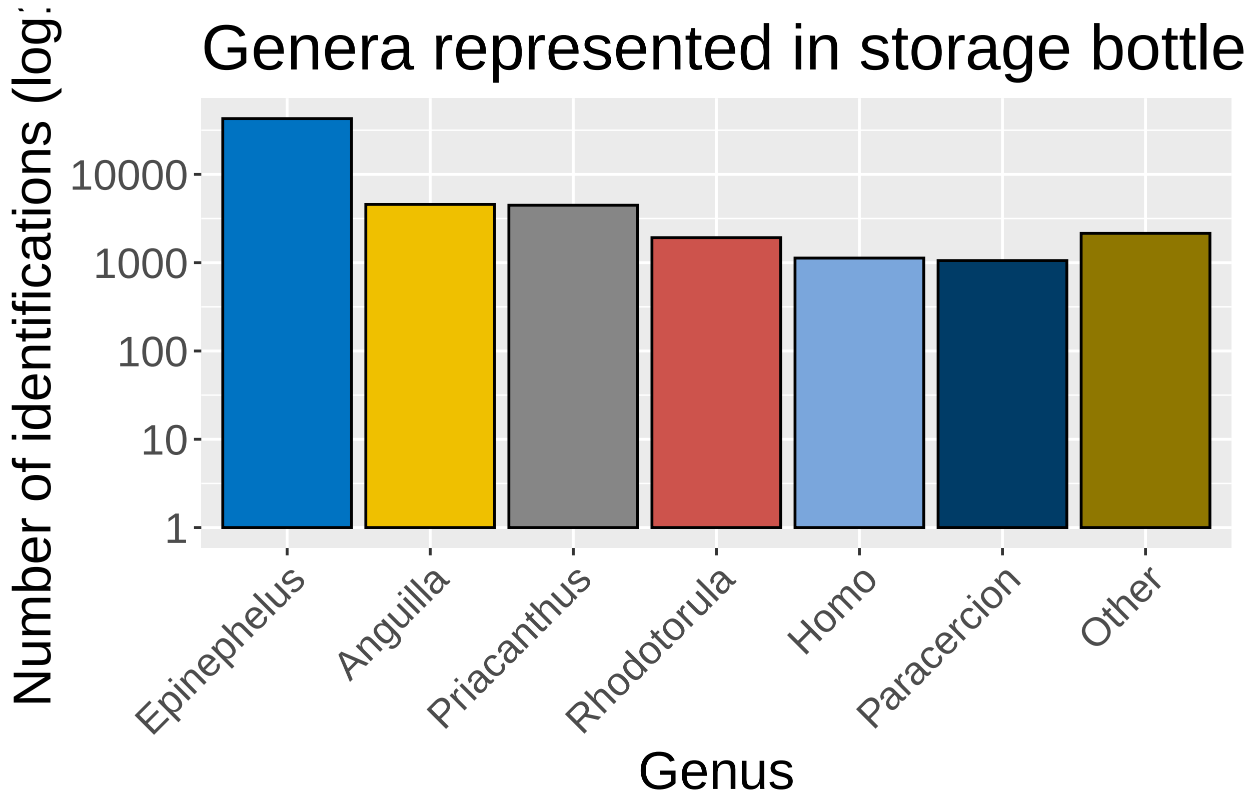
```r
# check for assumptions
check_assumption <- function(){
  plot(res.aov, 1) # homogeneity of variances
  plot(res.aov, 2) # normality of residuals
  shapiro.test(residuals(res.aov))} # shapiro wilk of anova residuals
```

investigate storage bottle control

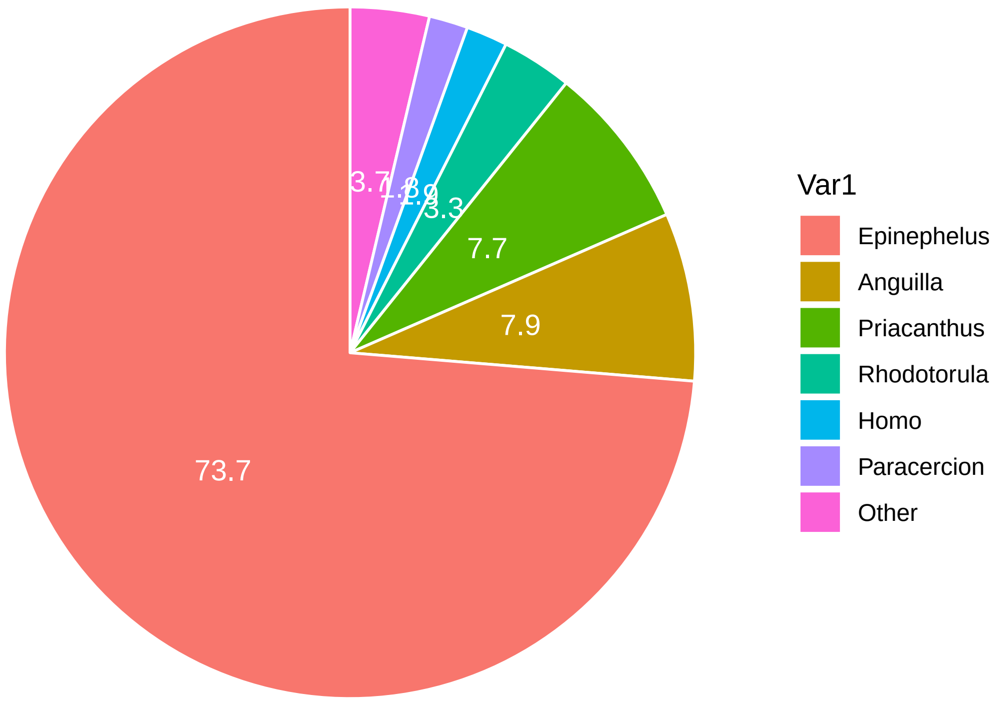
```r
`lca storage` <- read.delim("~/Documents/derep_illum/controls/underep/taxadded/lca") ## load lca file o

species <- table(`lca storage`$X.genus) %>%
  data.frame() %>%
  mutate(Var1 = ifelse(Freq < 1000, "Other", as.character(Var1))) %>%
  filter(!Var1=="no identification") %>%
  group_by(Var1) %>%
  dplyr::summarise(Freq = sum(Freq)) %>%
  mutate(Prop = (Freq/sum(Freq))*100) %>%
  ungroup() %>%
  mutate(Var1 = fct_reorder(Var1, desc(Freq))) %>%
  mutate(Var1 = fct_relevel(Var1, "Other", after = Inf))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
get_col <- function(){
  colorcount <- length(genuscount$Var1)
  qual_col <- brewer.pal.info[brewer.pal.info$category == "qual",]
  col_vector <- unlist(mapply(brewer.pal,
                              qual_col$maxcolors, rownames(qual_col)))
  mycol <- sample(col_vector, colorcount)}


ggplot(species, aes(x = Var1, y = Freq, fill = Var1)) +
  geom_bar(stat = "identity", color = "black") +
  theme(text = element_text(size = 20),
        axis.text.x = element_text(angle = 45, hjust =1, size = 15),
        legend.position = "none") +
  scale_fill_jco() +
  labs(title = "Genera represented in storage bottles",
       x = "Genus\n" ,
       y = "Number of identifications (log10)") +
  scale_y_log10()
```



```
# Pie Chart
# add position of label
count.data <- species %>%
  arrange(desc(Var1)) %>%
  mutate(lab.ypos = cumsum(Prop) - 0.5*Prop)

ggplot(count.data, aes(x = "", y = Prop, fill = Var1)) +
     geom_bar(width = 1, stat = "identity", color = "white") +
     coord_polar("y", start = 0)+
     geom_text(aes(y = lab.ypos, label = round(Prop,1)), color = "white")+
     theme_void()
```



2

##Load OTU table and check read numbers Functions:

```
getData <- function(){
  otu <- fread("~/Documents/derep_illum/changedheader/otu.about")
  before <- nrow(otu)
  # Remove X. or X from colnames
  names(otu) <- sub("#", "", names(otu))
  otu <- column_to_rownames(otu, var = "OTU ID")}


remove_chimeras <- function(){
  chimeras <- read.csv("~/Documents/IlluminaAdaptertrimmedAllreps/thingremoved", header=FALSE, sep=";")
  otu <- column_to_rownames(otu, var = "OTU.ID")
  otu <- otu[ ! sub("^.*?:", "", otu$OTU.ID) %in% chimeras$V1,] ##remove all chimeras
  after <- nrow(otu)
  cat(paste("removed", before-after, "chimeric sequences\n\n"))
  rownames(otu) <- NULL
  return(otu)}

# optional chimera removal: otu <- remove_chimeras()

# print results nicely:
```

```
myOTUcat <- function(){
  #total read sum in all clusters
  total_reads <- sum(rowSums(otu))
  cat(paste('total reads (grand total with which clustering was done):\n',
            total_reads))
  cat("\n\n", 'Summary statistics of number of reads per OTU:\n')
  print(summary(rowSums(otu)))
  cat(paste("\n\n",
            'Total number of OTUs (including singletons):\n', nrow(otu)))
}
```

Execution:

```
otu <- getData()
myOTUcat()
```

```
## total reads (grand total with which clustering was done):
##  2512237
##
##  Summary statistics of number of reads per OTU:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##     1.00     1.00     1.00     8.46     2.00 86074.00
##
##
##  Total number of OTUs (including singletons):
##  296884
```

Filter out out controls

If a OTU also contains control reads, these need to be filtered out of the samples contain them in frequencies that are close to the control frequencies. This could be contamination from the bottles the sample was stored in, or PCR contamination.

```
posContamination <- function(){
  contam <- otu %>% filter(unicon1 > 5000) %>% select(!c("unicon1",
             "unicon1A")) %>% sum()
  total <- otu %>% filter(unicon1 > 5000) %>% sum()
  freq <- contam/total
  cat(paste("Contamination percentage of positive control in other samples: \t", round(freq*100, 5), "\
  return(freq)}

negContamination <- function(){
  contam <- otu %>% select(neg_controle) %>% sum()
  total <- otu %>% filter(neg_controle>0) %>% sum()
  freq <- contam/total
  cat(paste("Contamination percentage in negative samples: \t", round(freq*100, 5)))}


rate <- posContamination()
```

```
## Contamination percentage of positive control in other samples:    0.00671
```

```r
negContamination()
```

```
## Contamination percentage in negative samples:     0.0612
```

Low abundance filter

the rate of contamination in the positive control was used as low abundance filter rate.

```r
lowAbuncanceFilter <- function(rate){
  before <- nrow(otu)
  colsum <- colSums(otu)
  min_read <- colsum * rate  # if OTU contains less than this many reads, filter out
  otu <-
    mapply(col = otu, min = min_read, function(col, min){
    col[col < min] <- 0
    col}) %>%
    as.data.frame () %>%
    `rownames<-`(rownames(otu)) %>% filter(!rowSums(.[samples]) == 0) # take out "empty" otus
  after <- nrow(otu)
  percenage_ret <- ((before-after)/before)*100
  cat(paste("filtered out ", before-after, " OTUs, which is ",
            round(percenage_ret, 2), "% of original OTUs
            \n\n",
            after, " OTUs were retained", sep = ""))
  return(otu)
}

controls <- c("sxm_2018_62", "sxm_2018_63", "sxm_2018_64",
              "sxm_2018_65", "sxm_2018_66", "sxm_2018_70",
              "sxm_2018_71", "0", "unicon1",
              "unicon1A", "neg_controle")
bottlecontrol <- c("sxm_2018_62", "sxm_2018_63", "sxm_2018_64",
              "sxm_2018_65", "sxm_2018_66", "sxm_2018_70",
              "sxm_2018_71")
samples <- names(otu)[-which(names(otu) %in% controls)]

otu <- lowAbuncanceFilter(rate = rate)
```

```
## filtered out 244221 OTUs, which is 82.26% of original OTUs
##
##
## 52663 OTUs were retained
```

remove singleton OTUs

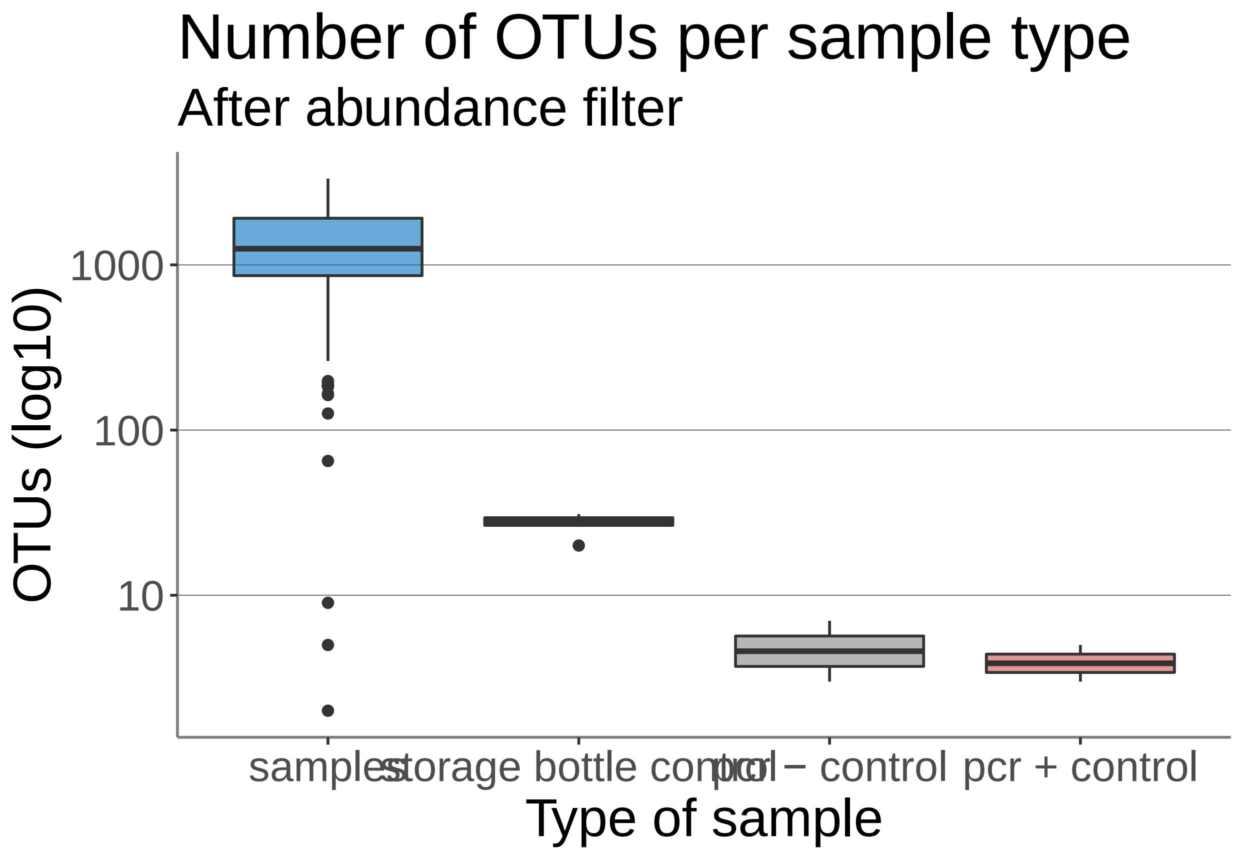
```r
remove_singletons <- function(){
  OTUbefore <- nrow(otu)
  otu <- otu %>% filter(!rowSums(.[samples]) < 2) # remove all rows where rowSum == 1 (singleton OTU)
  OTUafter <- nrow(otu)
  removed <- OTUbefore-OTUafter
  retained_percent <- round(OTUafter/OTUbefore*100, 2)
  cat(paste("removed", removed, "singletons\n",
            "retained", OTUafter, "OTUs, which means", retained_percent, "percent of OTUs was retained"
            sep = " "))
  return(otu)}

otu <- remove_singletons()
```

```
## removed 12254 singletons
##  retained 40409 OTUs, which means 76.73 percent of OTUs was retained
```

plot number of otus per sample

```
saba <- samples[grepl("sxm", samples)]


copy <- otu
copy[copy>0] <- 1
copy <- copy %>% colSums() %>%
  as.data.frame() %>% rownames_to_column(var = "sample")%>%
  mutate(type = ifelse(sample %in% bottlecontrol, "storage bottle control",
                       ifelse(grepl("unicon", sample), "pcr + control",
                              ifelse(sample %in% c("0", "neg_controle"),
                                     "pcr - control", "samples")))) %>%
  mutate(type = fct_reorder(type, desc(.)))


control_plotOTU <-
  ggplot(copy, aes(x = type,
                          y = .,
                          fill = type)) +
  geom_boxplot() +
  labs(title = "Number of OTUs per sample type",
       subtitle = "After abundance filter",
       x = "Type of sample",
       y = "OTUs (log10)") +
  scale_fill_jco(alpha = 0.6) +
  scale_y_log10() +
  # edit lines and background
  theme(text = element_text(size = 20),
        panel.grid.major.x = element_blank(),
        panel.grid.major.y = element_line("gray50", size = 0.2),
        panel.background = element_blank(),
        axis.line = element_line("gray50"),
        legend.position = "none")

control_plotOTU
```

```
## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```



```
ggsave("controlplot AFTER abundance filter", plot = control_plotOTU, device = "png", height = 7, width =
```

```
## Warning: Transformation introduced infinite values in continuous y-axis
```

```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```

Additional bottle control contamination check for saba samples

And remove controls from otu table

```
# establish controls and samples
controls <- c("sxm_2018_62", "sxm_2018_63", "sxm_2018_64",
              "sxm_2018_65", "sxm_2018_66", "sxm_2018_70",
              "sxm_2018_71", "0", "unicon1",
              "unicon1A", "neg_controle")
bottlecontrol <- c("sxm_2018_62", "sxm_2018_63", "sxm_2018_64",
              "sxm_2018_65", "sxm_2018_66", "sxm_2018_70",
              "sxm_2018_71")
samples <- names(otu)[-which(names(otu) %in% controls)]
saba <- samples[grepl("sxm", samples)]

# go over the rows(OTUs) where there are control reads and change any reads to 0 if they contain less t
filter_controls <- function(){
  OTUbefore <- nrow(otu)
  mcr <- do.call(pmax, otu[bottlecontrol]) # max control value for each otu
  mcp <- mcr > 0                          # control values  > 0
  otu[mcp, saba][otu[mcp, saba] <  2*mcr[mcp]] <- 0
  # discard controls, and OTUs that have no reads associated bc of control filter
  otu <- otu %>%
    select(all_of(samples)) %>% #only keep samples
    filter(!rowSums(.) == 0)# discard OTUs that have no reads because of filtering
  ncolbefore <- ncol(otu)
  OTUafter <- nrow(otu)
  cat(paste("Control filtering removed", OTUbefore-OTUafter, "OTUs, which is ",
          round(((OTUbefore-OTUafter)/OTUbefore)*100, 2)), "%")
  otu <- otu[,colSums(otu) > 2000]
  ncolafter <- ncol(otu)
  after2000 <- nrow(otu)
  cat(paste("\n\nanother", OTUafter-after2000, "OTUs, were removed by removing ", ncolbefore-ncolafter,
  return(otu)}

otu <- filter_controls()
```

```
## Control filtering removed 33 OTUs, which is  0.08 %
##
## another 0 OTUs, were removed by removing  3 samples below 2000 reads
```

```
# select only samples that have read counts of higher than two thousand
```

write sequences to blast to file

now that all control reads and singletons have been filtered out, the remaining OTUs can be blasted. For this, the OTU centroid sequences from filtering step at 98% are extracted and then blasted -> taxadded -> dummyadded(for LCA script to work) -> lca script. Then its back to R

```
# make file with which sequences to be blasted can be selected (singletons filtered out)
write.table(rownames(otu),
            file = '~/OTUcentroids.txt',
            row.names = FALSE,
            quote = FALSE,
            col.names = FALSE)
nrow(otu)
```

```
## [1] 40376
```

3. LCA

remove bacteria hits

```
genbank <- read.delim("~/Downloads/Galaxy6-[filtOTUseqs40376.fasta_BLAST_original_taxonomy_lca].tabular

bact <- genbank %>% filter(X.kingdom == "Bacteria")
```

```r
toremove <- bact$X.Query
length(toremove)
```

```
## [1] 22428
```

```r
otu <- otu[!row.names(otu) %in% toremove,]
nrow(otu)
```

```
## [1] 17948
```

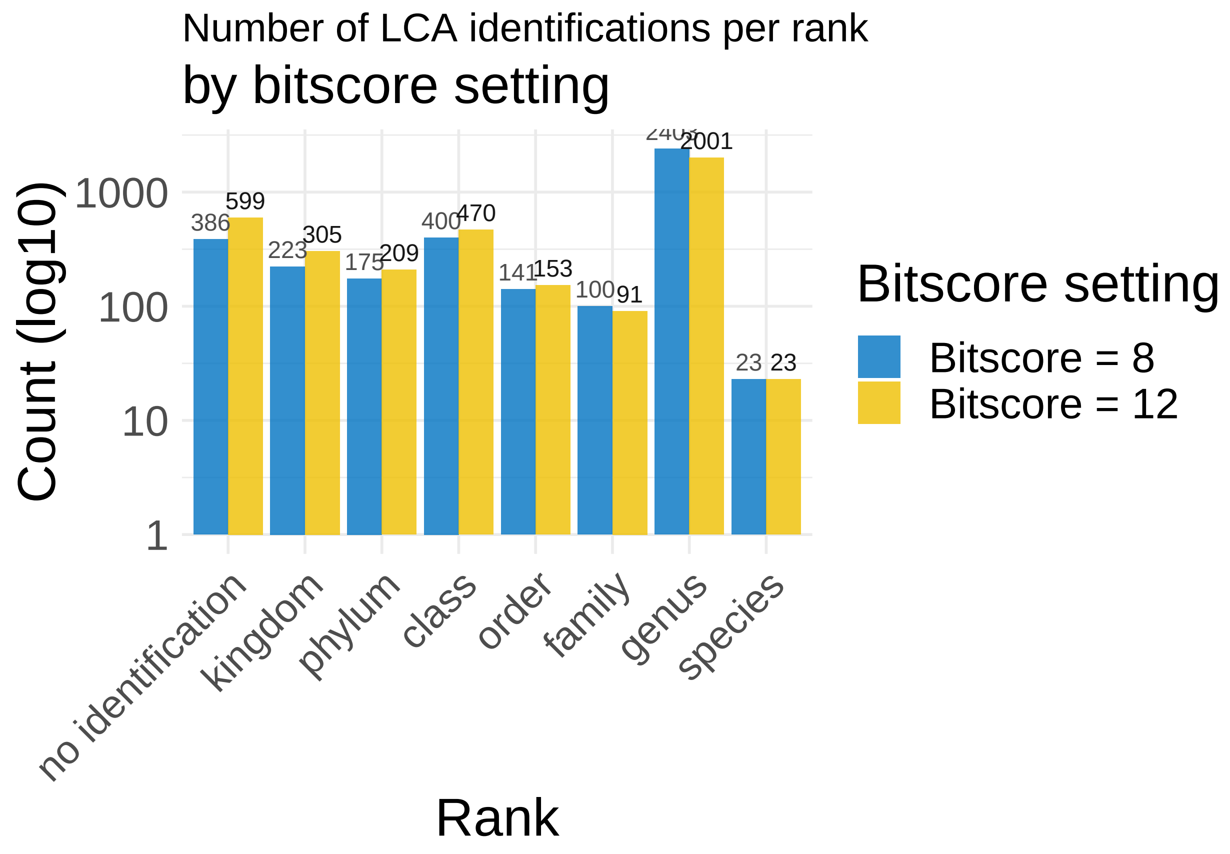
import and prepeare lca data

```r
getLca <- function(){
  df <- read.delim("~/Documents/derep_illum/changedheader/taxadded/bit8range")
  df2 <- read.delim("~/Documents/derep_illum/changedheader/taxadded/bit12range")
  dfs <- list(df, df2)
  # remove X. from cols and name the dfs
  dfs <-
    lapply(dfs, function(x){setNames(x, sub("^X.", "", names(x)))}) %>%
    `names<-`(c("Bitscore = 8", "Bitscore = 12"))}

#execute the functions
lcas <- getLca()
lcas <- lapply(lcas, function(x) {x <- x[!x$Query %in% toremove,]})

# add information on how many reads were captured by the bitscore threshold
merged_dfs <-
  lapply(1:length(lcas), function(x) lcas[[x]] %>%
    data.frame) %>%
  # create extra column with what bitscore was used and bind the dataframes
  Map(cbind, ., Bitscore_setting = names(lcas)) %>%  # info of bitscore for bth dfs
  do.call(rbind, .) %>%  #combined them by row
  data.frame() %>%
  filter(!grepl("sp\\.", species)) # remove hits that contain sp. because theyre not informative.

# get factor levels in right order for nice looking plots
merged_dfs$lca.rank <- factor(merged_dfs$lca.rank, levels = c("no identification", colnames(merged_dfs)
```

plot number of taxa found per rank by bitscore setting

Bitscore 8 has a stronger bias towards genus level identifications because the it takes fewer reads into account for the lca deterimination so higher chance that theres only 1 read to do taxa determination with,

```r
myPlot <- function(){
  merged_dfs %>%
    ggplot(aes(x= lca.rank, y = ..count.., group = Bitscore_setting)) +
    geom_bar(aes(fill=`Bitscore_setting`),
             position = "dodge", alpha = 0.8) +
    geom_text(stat = "count",
               aes(label = ..count.., colour = Bitscore_setting),
               position = position_dodge(0.9),
               vjust = -0.5, size = 3) +
    scale_fill_jco() +
    #scale_fill_brewer(type = "qual", palette = "Pastel2") +
    #scale_fill_manual(values = alpha(c("#00AFBB", "#FC4E07"), 0.8)) +
    theme_minimal() +
    scale_color_manual(values = c("gray30", "gray8"),
                       guide = F) +
    scale_y_log10() +
    labs(title = "Number of LCA identifications per rank",
         subtitle = "by bitscore setting",
         x = "Rank",
         y = "Count (log10)",
         fill = "Bitscore setting") +
    theme(plot.title = element_text(size = 15, vjust = 1),
          text = element_text(size = 20),
          axis.text.x = element_text(angle = 45, hjust =1, size = 15))}
p <- myPlot()
p
```



```
ggsave(filename = "Bitscore plot", p, device = "png", width = 10, height = 7.5)
```

Combine OTU and sampling location data

Compare the number of centroids supplied to the blast file to the number of blast hits found that had at least one blast hit of 70% identity and 70% coverage. get LCA: numbers

```
merge_otu_lca  <- function(){
  otu <- rownames_to_column(otu, var = "Query")
  df_otu_lca <- merge(otu, lcas[[2]], by='Query', all.x = TRUE)
  total_otu <- nrow(otu)
  OTUs_hit <- length(unique(lcas[[2]]$Query))
  lca_hit <- length(lcas[[2]]$lca.rank[lcas[[2]]$lca.rank != "no identification"])
  cat(paste("\tnumber of rows in OTU file (number of OTUs found):\t", total_otu,
            "\n\n",
            "\tnumber of rows in LCA file (OTUs that had at least one blast hit):\t", OTUs_hit,
            "\n\nPercentage of dark taxa is: ", round((1-lca_hit/total_otu)*100,2), "%"))

  return(df_otu_lca)}

otu_lca <- merge_otu_lca()
```

```
##  number of rows in OTU file (number of OTUs found):   17948
##
##      number of rows in LCA file (OTUs that had at least one blast hit):   3848
##
## Percentage of dark taxa is:  81.87 %
```

combine sampling location data with the OTU table

```
get_bin_tags <- function(df){
    mdf %>%
    filter(sample %in% colnames(df)) %>%
    mutate(habitat = as.character(habitat)) %>%
    mutate(habitat = replace_na(habitat, "Mixed")) %>%
    mutate(bin = paste(tag, habitat)) %>%
    t() %>%
    as.data.frame() %>%
    row_to_names(1) %>%
    rownames_to_column(var = "Query") %>%
    filter(Query == "bin")
  }

bins_instead <- function(df){
  df <- df %>% rownames_to_column(var = "Query")
  with_bins <- rbind.fill(tags, df) %>%
  row_to_names(row_number = 1)
  colnames(with_bins)[1] <- "Query"
  return(with_bins)
}
```

```r
# make a new df with one column calles Query (for merging), bind by col and remove rownames
#otu <- rownames_to_column(otu, var = "Query")
```

Execution:

```r
tags <- get_bin_tags(otu)
for_network <- bins_instead(otu)
```

```
## Warning in row_to_names(., row_number = 1): Row 1 does not provide unique names.
## Consider running clean_names() after row_to_names().
```

prepare data for summarising per habitat

```r
prep <- function(df){
  df %>%
  `rownames<-`(NULL) %>%
  column_to_rownames(var = "Query") %>%
  data.matrix() %>%
  t() %>%
  as.data.frame() %>%
  rownames_to_column(var = "habitat") %>%
  filter(!habitat == "Saba.North.100.m.above.bottom") %>%
  mutate(habitat = sub("\\.\\d+$", "", habitat)) %>%
  filter(!habitat == "Saba.North.100.m.above.bottom")
}

aggr <- function(df){
  aggregate(df[,-1], list(habitat = df$habitat), mean) %>%
    mutate(habitat = habitat %>%
             sub("Saba.South", "SS", .) %>%
             sub("Saba.North", "SN", .) %>%
             sub(".above.bottom", "ab", .) %>%
             sub(".layer", "", .) %>%
             gsub("\\.", " ", .) %>%
             sub("0 05", "0.05", ., fixed = TRUE)) %>%
    column_to_rownames(var = "habitat") %>%
    t() %>%
    as.data.frame %>%
    rownames_to_column(var = "habitat")
}
```

Execution:

```r
try <- prep(for_network)

try[,-1][try[,-1]>0] <- 1

agr <- try %>% aggr()
```

```r
filtered_animalia <- merge(agr, lcas[[2]],
                           all.x = TRUE,
                           by.x = "habitat",
                           by.y = "Query") %>%
  filter(kingdom == "Animalia") %>%
  mutate(otu_tax = paste(kingdom, phylum, class, order, family, genus, species, sep = "/") %>%
           gsub("no identification", "NA", .)) %>%
  mutate(habitat = paste("OTU", seq_along(habitat), " ", otu_tax, sep = "")) %>%
  select(!c(colnames(lcas[[2]])[-1], otu_tax))
```

```r
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

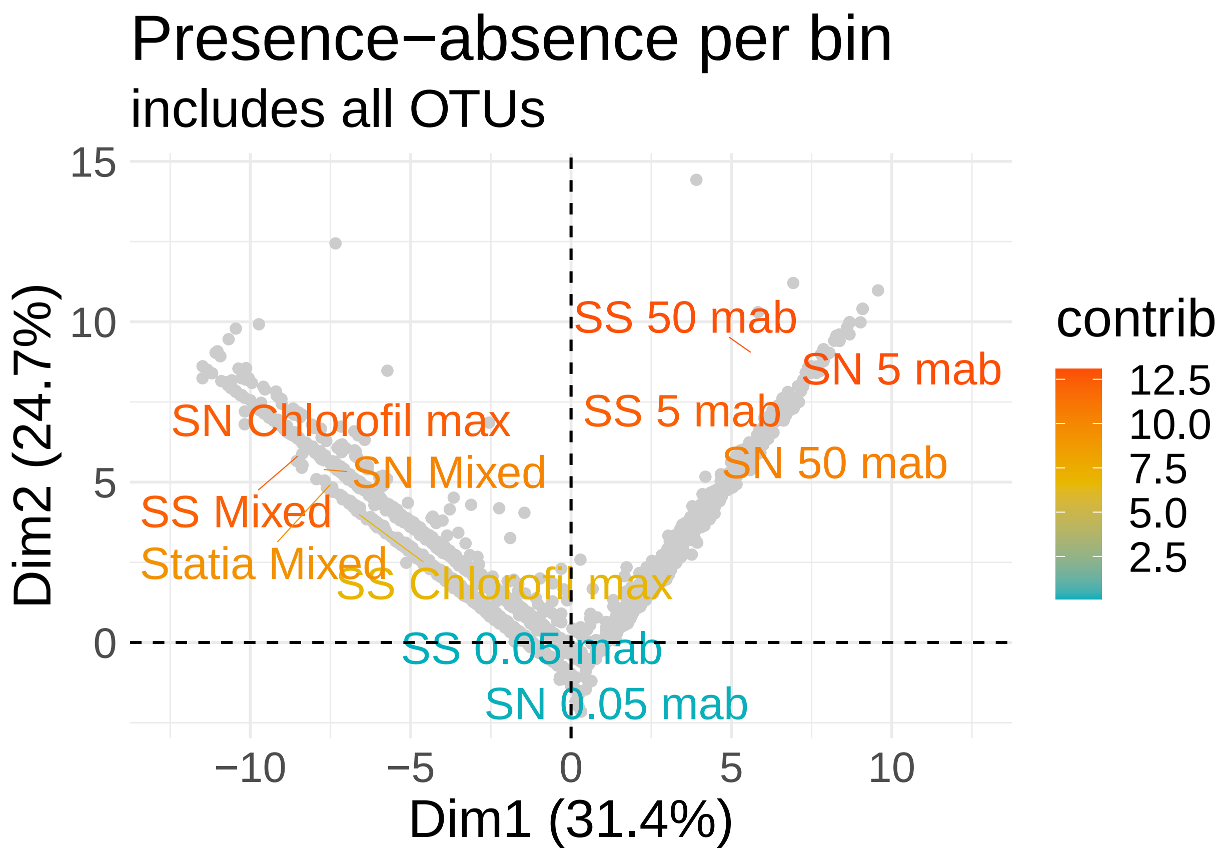
```r
myPca <- function(df){
  res.pca <- prcomp(column_to_rownames(df, var = "habitat"),
                    center = TRUE, scale. = TRUE)
  print(get_eig(res.pca))
  fviz_eig(res.pca)
  fviz_pca_biplot(res.pca,
                  label = "var",
                  col.var = "contrib",
                  gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
                  col.ind = "gray80",
                  repel = TRUE,
                  geom.ind = "point",
                  geom.var = "text",
                  labelsize = 6) + theme_minimal() + theme(text = element_text(size = 20))
}

p_agr <- myPca(agr) +
  labs(title = "Presence-absence per bin",
                           subtitle = "includes all OTUs") + xlim(-12.5, 12.5)
```

```
##        eigenvalue variance.percent cumulative.variance.percent
## Dim.1   3.4579667        31.436061                    31.43606
## Dim.2   2.7160396        24.691269                    56.12733
## Dim.3   0.9972303         9.065730                    65.19306
## Dim.4   0.9955104         9.050094                    74.24315
## Dim.5   0.7590921         6.900837                    81.14399
## Dim.6   0.5116063         4.650966                    85.79496
## Dim.7   0.4595809         4.178008                    89.97297
## Dim.8   0.3532729         3.211572                    93.18454
## Dim.9   0.2794858         2.540780                    95.72532
## Dim.10  0.2540913         2.309921                    98.03524
## Dim.11  0.2161238         1.964762                   100.00000
```

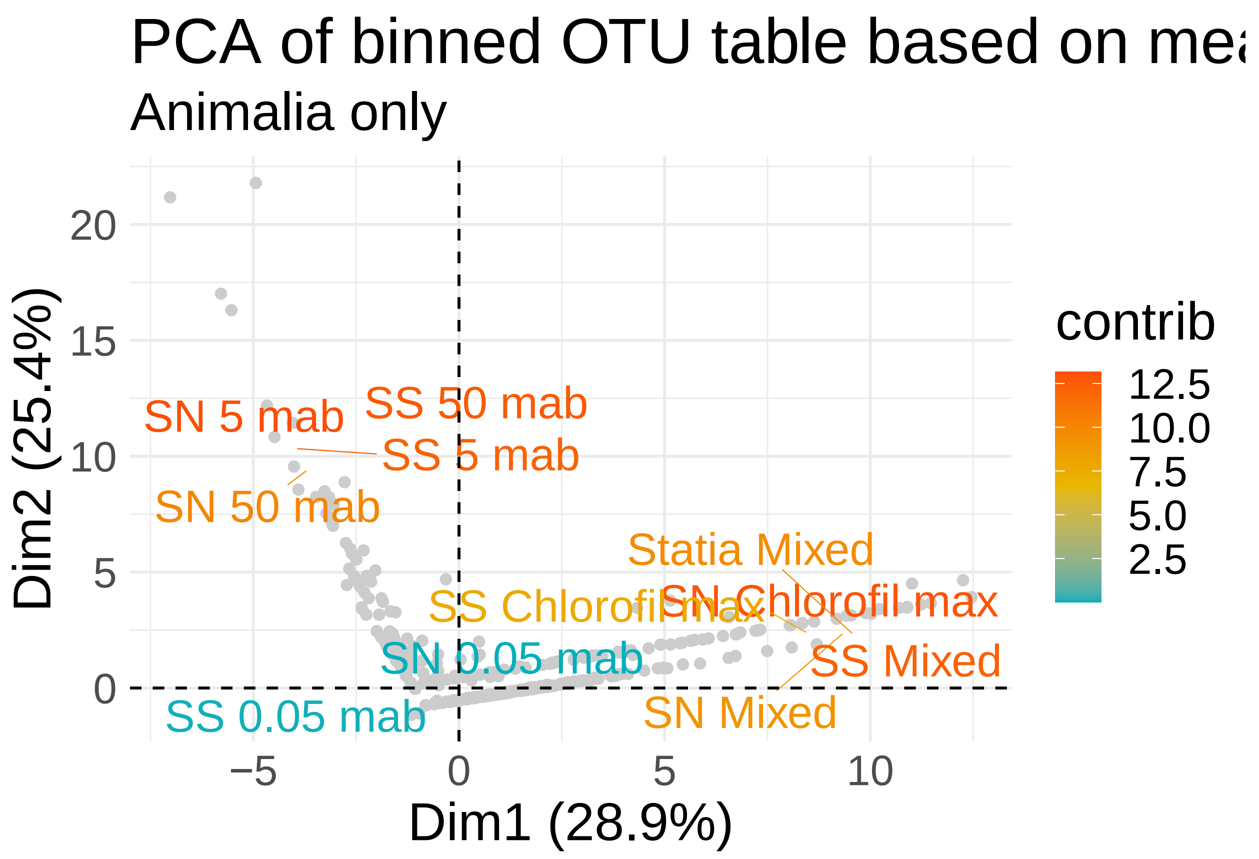
```r
p_animalia <- myPca(filtered_animalia) +
  labs(title = "PCA of binned OTU table based on mean presence per bin",
       subtitle = "Animalia only")
```

```
##        eigenvalue variance.percent cumulative.variance.percent
## Dim.1   3.1815469        28.923153                    28.92315
## Dim.2   2.7962682        25.420620                    54.34377
## Dim.3   1.0124769         9.204335                    63.54811
## Dim.4   0.9975567         9.068697                    72.61681
## Dim.5   0.6680307         6.073006                    78.68981
## Dim.6   0.5966768         5.424334                    84.11415
## Dim.7   0.5577649         5.070590                    89.18474
## Dim.8   0.3659879         3.327162                    92.51190
## Dim.9   0.2999111         2.726465                    95.23836
## Dim.10  0.2859852         2.599866                    97.83823
## Dim.11  0.2377949         2.161772                   100.00000
```

```
p_agr
```



```
p_animalia
```



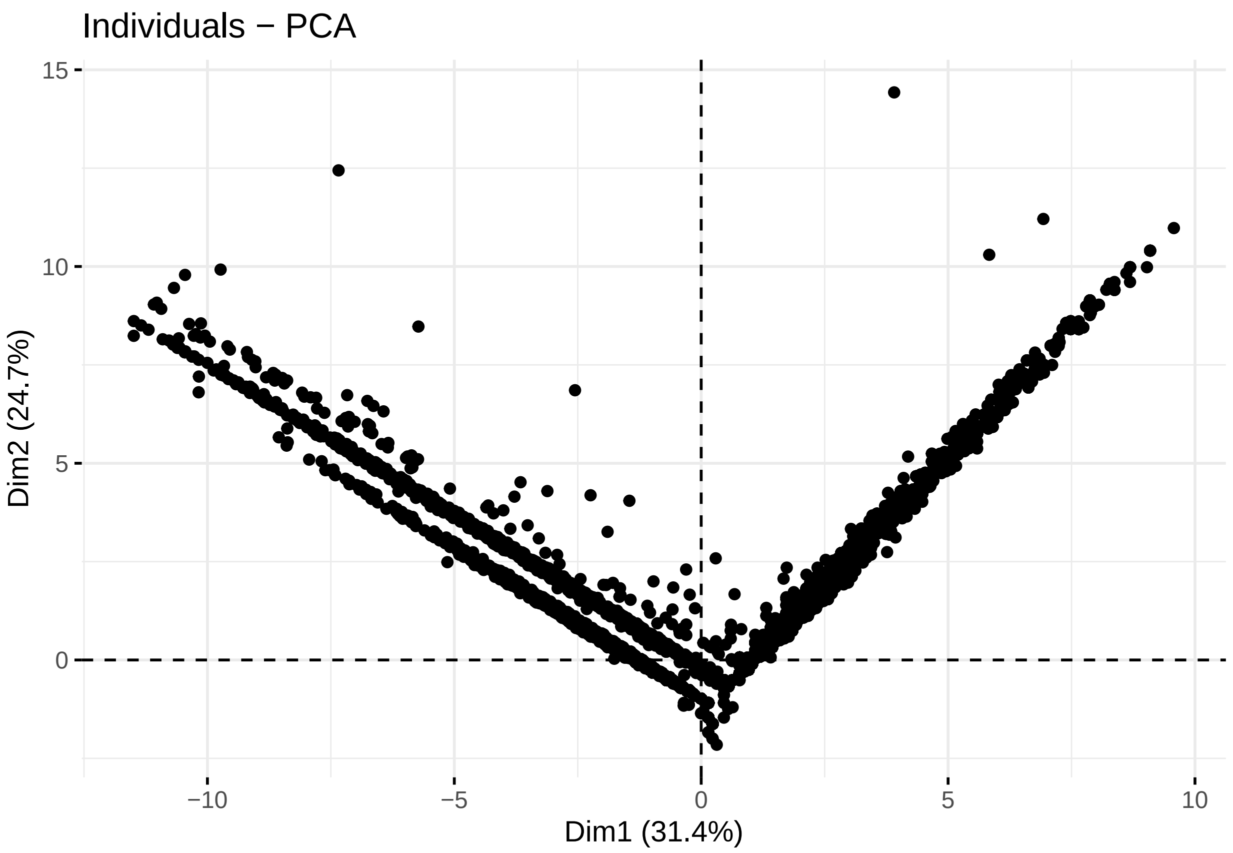
```
ggsave("finalpcapresence", p_agr, device = "png")
```

```
## Saving 6.5 x 4.5 in image
```

```
ggsave("try<-1mean_animalia", p_animalia, device = "png")
```

```
## Saving 6.5 x 4.5 in image
```

```
res.pca <- prcomp(agr[,-1], center = TRUE, scale. = TRUE)
fviz_pca_ind(res.pca, label = "none")
```


Individuals − PCA
Dim2 (24.7%)
Dim1 (31.4%)
15
10
5
0
−10
−5
0
5
10