04.Community_MassEffect

Angel Rain

Contents

1 Setting up the workspace

1.1 Loading Packages

```
rm(list = ls())
library(cowplot)
library(egg)
library(readxl)
library(RColorBrewer)  #Expando color palette
library(dplyr)
library(stringr)  # FOr editing string
library(reshape2)  #For 'melt' function
library(olsrr)
library(ggtext)
```

2 Loading datasets

2.1 Colorblind palette

```
# Setting the colorblind palette
cbp1 <- c("#999999", "#FFDB6D", "#E69F00", "#56B4E9", "#009E73", "#F0E442",
    "#0072B2", "#D55E00", "#CC79A7", "#293352")
# Expanding the standard pallete Paste1 for downstream analysis
mycolors <- colorRampPalette(brewer.pal(8, "Pastel1"))(22)
```

2.2 OTU table and metadata

```
df1 = data.frame(read.csv("../data/OTU_table_merged200_SILVA_megablast.csv",
    row.names = 1))
tibble(df1)
```

```
## # A tibble: 1,047 x 107
##    Kingdom  Phylum       Class Order Family Genus Specie     C01    C010    C011
##    <chr>    <chr>        <chr> <chr> <chr>  <chr> <chr>    <dbl>   <dbl>   <dbl>
##  1 Bacteria Proteobacte~ Gamm~ Ente~ Alter~ Rhei~ marin~ 0       0       0
##  2 Bacteria Bacteroidota Bact~ Flav~ Flavo~ Flav~ marin~ 0.00893 1.24e-2 0.00206
##  3 Bacteria Chloroflexi  SL56~ mari~ <NA>   o_ma~ <NA>   0       3.44e-4 0
##  4 Bacteria Actinobacte~ Acid~ Micr~ Iluma~ f_Il~ marin~ 0       0       0
##  5 Bacteria Proteobacte~ Gamm~ Burk~ Comam~ f_Co~ marin~ 0       0       0
##  6 Bacteria Proteobacte~ Gamm~ Burk~ Comam~ Pelo~ Pelom~ 0       0       0
##  7 Bacteria Proteobacte~ Gamm~ Burk~ Comam~ f_Co~ <NA>   0       0       0
##  8 Bacteria Proteobacte~ Gamm~ Pseu~ Pseud~ Pseu~ bacte~ 0       0       0
##  9 Bacteria Bacteroidota Bact~ Flav~ Flavo~ Flav~ Flavo~ 0       1.75e-2 0
## 10 Bacteria Proteobacte~ Gamm~ Pseu~ Pseud~ Pseu~ Pseud~ 0       3.44e-4 0
## # i 1,037 more rows
## # i 97 more variables: C012 <dbl>, C013 <dbl>, C014 <dbl>, C015 <dbl>,
## #   C016 <dbl>, C017 <dbl>, C018 <dbl>, C019 <dbl>, C02 <dbl>, C020 <dbl>,
## #   C03 <dbl>, C04 <dbl>, C05 <dbl>, C06 <dbl>, C07 <dbl>, C08 <dbl>,
## #   C09 <dbl>, C11 <dbl>, C110 <dbl>, C111 <dbl>, C112 <dbl>, C113 <dbl>,
## #   C114 <dbl>, C115 <dbl>, C116 <dbl>, C117 <dbl>, C118 <dbl>, C119 <dbl>,
## #   C12 <dbl>, C120 <dbl>, C13 <dbl>, C14 <dbl>, C15 <dbl>, C16 <dbl>, ...
```

```
meta = data.frame(read.csv("../data/metadata_merged200_SILVA_megablast.csv",
    row.names = 1))
tibble(meta)
```

```
## # A tibble: 100 x 3
##    Cycle Replicate M.ID
##    <chr>     <int> <chr>
##  1 C0            1 C01
##  2 C0           10 C010
##  3 C0           11 C011
##  4 C0           12 C012
##  5 C0           13 C013
```

```
##  6 C0           14 C014
##  7 C0           15 C015
##  8 C0           16 C016
##  9 C0           17 C017
## 10 C0           18 C018
## # i 90 more rows
```

2.3 Bulk community properties

```
Comm.properties <- read.csv("../data/CommunityProperties_CycleExp.csv",
    header = T)
tibble(Comm.properties)
```

```
## # A tibble: 160 x 8
##    Date     Sample.ID Cycle Microcosm Abundance_106cellmL Biomass_mgCL
##    <chr>    <chr>     <chr>     <int>               <dbl>        <dbl>
##  1 10.10.19 C01       C0            1                3.01       NA
##  2 10.10.19 C010      C0           10                2.67        0.993
##  3 10.10.19 C011      C0           11                1.34        1.4
##  4 10.10.19 C012      C0           12                6.12        0.125
##  5 10.10.19 C013      C0           13                6.89        0.769
##  6 10.10.19 C014      C0           14                5.12        0.325
##  7 10.10.19 C015      C0           15                2.54        0.323
##  8 10.10.19 C016      C0           16                3.49        1.50
##  9 10.10.19 C017      C0           17                5.76        1.59
## 10 10.10.19 C018      C0           18                3.15        0.385
## # i 150 more rows
## # i 2 more variables: ConsoCellobiose_mgCL <dbl>, CUE <dbl>
```

2.4 Genomic properties of MAGs

```
df.gene.cat <- data.frame(read_xlsx("../data/MAGs_genomic_properties.xlsx",
    sheet = "Pathways_KEGG"))
row.names(df.gene.cat) <- df.gene.cat[, 2]

labels <- data.frame(Functional.category = df.gene.cat[, 1])
row.names(labels) <- df.gene.cat[, 2]
labels$Functional.category <- as.factor(labels$Functional.category)
labels$Functional.category <- factor(labels$Functional.category, c("Secretion systems",
    "Amino acid biosynthesis", "Cellob. degradation"))
```

2.5 Elo-rating data from Cycle 0 to Cycle 6

```
Elo.0 = read.csv("../data/EloRating_C0_results1000iter.csv", header = T)[,
    -1]
Elo.1 = read.csv("../data/EloRating_C1_results1000iter.csv", header = T)[,
    -1]
```

```
Elo.4 = read.csv("../data/EloRating_C4_results1000iter.csv", header = T)[,
    -1]
Elo.6 = read.csv("../data/EloRating_C6_results1000iter.csv", header = T)[,
    -1]

sum.elo.0 = aggregate(. ~ player_id + n_games, data = Elo.0[, -1], mean)
sum.elo.0$Cycle = "C00"
head(sum.elo.0)
```

```
##            player_id n_games    rating Cycle
## 1     Bradyrhizobium       1 1000.1723   C00
## 2          Emticicia       1  990.3320   C00
## 3      Brevundimonas       2  995.2917   C00
## 4 f_Caulobacteraceae       2 1064.3713   C00
## 5        Caulobacter       3 1065.1167   C00
## 6      Chitinibacter       3  990.1293   C00
```

```
sum.elo.1 = aggregate(. ~ player_id + n_games, data = Elo.1[, -1], mean)
sum.elo.1$Cycle = "C01"
head(sum.elo.1)
```

```
##            player_id n_games   rating Cycle
## 1     Bradyrhizobium       1 1024.644   C01
## 2          Emticicia       1 1015.626   C01
## 3      Brevundimonas       2 1040.649   C01
## 4 f_Caulobacteraceae       2 1062.287   C01
## 5        Caulobacter       3 1102.510   C01
## 6      Chitinibacter       3 1023.327   C01
```

```
sum.elo.4 = aggregate(. ~ player_id + n_games, data = Elo.4[, -1], mean)
sum.elo.4$Cycle = "C04"
head(sum.elo.4)
```

```
##            player_id n_games   rating Cycle
## 1     Bradyrhizobium       1 1050.645   C04
## 2          Emticicia       1 1053.112   C04
## 3      Brevundimonas       2 1037.692   C04
## 4 f_Caulobacteraceae       2 1070.247   C04
## 5        Caulobacter       3 1063.884   C04
## 6      Chitinibacter       3 1006.275   C04
```

```
sum.elo.6 = aggregate(rating ~ player_id + n_games, data = Elo.6[, -1],
    mean)
sum.elo.6$Cycle = "C06"
head(sum.elo.6)
```

```
##            player_id n_games    rating Cycle
## 1     Bradyrhizobium       1 1049.6802   C06
## 2          Emticicia       1 1070.9531   C06
## 3      Brevundimonas       2 1040.2766   C06
## 4 f_Caulobacteraceae       2 1066.6892   C06
## 5        Caulobacter       3 1070.2905   C06
## 6      Chitinibacter       3  984.0774   C06
```

```
sum.elo = rbind(sum.elo.0, sum.elo.1, sum.elo.4, sum.elo.6)
```

2.6 Subsetting community composition dataset

Summarize reads number at Genus levels for downstream analysis

```
# Overview at Genus level aggregate relative counts by Genus
df2 <- aggregate(. ~ Genus, data = df1[, c(6, 8:107)], sum, na.rm = TRUE)
colSums(df2[, 2:101])
```

```
##  C01 C010 C011 C012 C013 C014 C015 C016 C017 C018 C019  C02 C020  C03  C04  C05
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
##  C06  C07  C08  C09  C11 C110 C111 C112 C113 C114 C115 C116 C117 C118 C119  C12
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
## C120  C13  C14  C15  C16  C17  C18  C19  C41 C410 C411 C412 C413 C414 C415 C416
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
## C417 C418 C419  C42 C420  C43  C44  C45  C46  C47  C48  C49  C61 C610 C611 C612
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
## C613 C614 C615 C616 C617 C618 C619  C62 C620  C63  C64  C65  C66  C67  C68  C69
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
##  C71 C710 C711 C712 C713 C714 C715 C716 C717 C718 C719  C72 C720  C73  C74  C75
##    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
##  C76  C77  C78  C79
##    1    1    1    1
```

```
rownames(df2) <- df2[, 1]  #rownames
df2 <- df2[, -1]  #remove column with genera and keep only abundance data
```

3 Use Genus level for downstream analysis

```
# Setup dataframe
df2$Genus = row.names(df2)
tax.genus = unique(df1[, 1:6])  # Get full taxonomy

# Combine genus-level relative read numbers and full taxonomy
df3 = merge(tax.genus, df2, by = "Genus", all.x = TRUE)

row.names(df3) = df3$Genus
tibble(df3)
```

```
## # A tibble: 118 x 106
##    Genus       Kingdom Phylum Class Order Family     C01    C010    C011    C012
##    <chr>       <chr>   <chr>  <chr> <chr> <chr>    <dbl>   <dbl>   <dbl>   <dbl>
##  1 [Polyangiu~ Bacter~ Prote~ Gamm~ Burk~ Comam~ 0       0       0       7.56e-3
##  2 Acidovorax  Bacter~ Prote~ Gamm~ Burk~ Comam~ 4.05e-2 1.06e-2 0.0842  2.28e-1
##  3 Acinetobac~ Bacter~ Prote~ Gamm~ Pseu~ Morax~ 0       0       0       0
##  4 Aeromonas   Bacter~ Prote~ Gamm~ Ente~ Aerom~ 6.95e-1 6.87e-4 0.00103 3.44e-4
##  5 Algoriphag~ Bacter~ Bacte~ Bact~ Cyto~ Cyclo~ 3.44e-4 3.44e-4 0       0
```

```
##  6 Alicycliph~ Bacter~ Prote~ Gamm~ Burk~ Comam~ 3.78e-3 0       0       0
##  7 Allorhizob~ Bacter~ Prote~ Alph~ Rhiz~ Rhizo~ 0       0       0       0
##  8 Aquabacter~ Bacter~ Prote~ Gamm~ Burk~ Comam~ 8.59e-3 4.47e-1 0.00824 1.25e-1
##  9 Aquamicrob~ Bacter~ Prote~ Alph~ Rhiz~ Rhizo~ 0       0       0       0
## 10 Aquincola   Bacter~ Prote~ Gamm~ Burk~ Comam~ 0       2.40e-3 0       0
## # i 108 more rows
## # i 96 more variables: C013 <dbl>, C014 <dbl>, C015 <dbl>, C016 <dbl>,
## #   C017 <dbl>, C018 <dbl>, C019 <dbl>, C02 <dbl>, C020 <dbl>, C03 <dbl>,
## #   C04 <dbl>, C05 <dbl>, C06 <dbl>, C07 <dbl>, C08 <dbl>, C09 <dbl>,
## #   C11 <dbl>, C110 <dbl>, C111 <dbl>, C112 <dbl>, C113 <dbl>, C114 <dbl>,
## #   C115 <dbl>, C116 <dbl>, C117 <dbl>, C118 <dbl>, C119 <dbl>, C12 <dbl>,
## #   C120 <dbl>, C13 <dbl>, C14 <dbl>, C15 <dbl>, C16 <dbl>, C17 <dbl>, ...
```

#Estimating absolute number of cells

```
# Subset community cell numbers data to match Genus relative reads
# number data
Comm.properties = Comm.properties[Comm.properties$Cycle %in% c("C0", "C1",
    "C4", "C6", "C7"), ]
df3.sub = df3[, c(7:106)]

# Temporal file cell abundance per microcosm to calculate absolute
# abundances
Tmp.cells = matrix(rep(Comm.properties$Abundance_106cellmL, dim(df3.sub)[1]),
    nrow = dim(df3.sub)[1], byrow = T) * 1e+06

# Multiple the relative read numbers by the cell numbers to obtain
# absolute cells number per Genus
df3.sub.aa = df3.sub * Tmp.cells
```

3.0.1 Pooled community analysis

We implemented here an Index based in the Elo-rating from final cycle of biological interactions (Cycle6) to predict homogenizing dispersal event (Cycle 7).

```
# Step 1 Extract genus representing >0.1% (or 0.001 in proportions)
# of read numbers and data from Cycle 1-6 and Cycle 7
df.sub.aa <- df3.sub.aa[which(rowMeans(df3.sub) > 0.001), ]  #41 Genus
df.aa.C6_C7 = df.sub.aa[, c(61:100)]

#### Extract genus representing read numbers and data between Cycle 6
#### and Cycle 7
df.aa.C6_C7 = df.aa.C6_C7[rowSums(df.aa.C6_C7) > 0, ]  #41 Genus did make to Ccyle6
dim(df.aa.C6_C7)
```

```
## [1] 41 40
```

```
## Get only Cycle6 microcosms
df.aa.pool = df.aa.C6_C7[, 1:20]

## Recalculate relative abundance for the total community
df.aa.pool.C6 = as.data.frame(rowSums(df.aa.pool))
```

```
df.aa.pool.C6.norm = df.aa.pool.C6/20  #sum(df.aa.pool.C6) #<- HERE WE HAVE THE BASE FOR THE PREDICTION

# Dataframe with all data rename
names(df.aa.pool.C6.norm) = "Neutral.prediction"
df.aa.pool.C6.norm$Genus = row.names(df.aa.pool.C6.norm)
```

```
# For microcosm
df.C0.C6 = df3[, c(7:106)]
df.C0.C6 = df.C0.C6[rowMeans(df.C0.C6) > 0.001, ]  #Only those that represent 0.1% (41 Genera)
# row.names(df.C0.C6)=df.C0.C6$Genus
df.C0.C6 = df.C0.C6[, c(1:80)]
meta.C0.C6 = meta[c(1:80), ]
```

#Plot mean abundance Elo-rating

```
sum.elo.all=rbind(sum.elo.0,sum.elo.1,sum.elo.4,sum.elo.6)
df.C0<-data.frame(Abundance=rowMeans(df.C0.C6[,1:20]),Cycle="C00",player_id=names(rowMeans(df.C0.C6[,1:
df.C1<-data.frame(Abundance=rowMeans(df.C0.C6[,21:40]),Cycle="C01",player_id=names(rowMeans(df.C0.C6[,1
df.C4<-data.frame(Abundance=rowMeans(df.C0.C6[,41:60]),Cycle="C04",player_id=names(rowMeans(df.C0.C6[,1
df.C6<-data.frame(Abundance=rowMeans(df.C0.C6[,61:80]),Cycle="C06",player_id=names(rowMeans(df.C0.C6[,1

df.mean<-rbind(df.C0,df.C1,df.C4,df.C6)
df.mean$index<-paste0(df.mean$player_id,".",df.mean$Cycle)
sum.elo.all$index<-paste0(sum.elo.all$player_id,".",sum.elo.all$Cycle)
sum.elo.all<-merge(df.mean,sum.elo.all,by="index")

jpeg("../Figures/Elo_abundance_vs_rating.jpg",width = 10,height = 8,units = "cm",res=300)

plot.Elo.abundance<-sum.elo.all%>%
ggplot(aes(x=(Abundance+0.001),y=(rating),fill=Cycle.x,colour=Cycle.x))+
  geom_point(alpha=0.5,shape=21,size=1.5,color="black")+
  theme_bw()+labs(y="Elo-rating",x="Average  % read counts")+
  scale_color_brewer(palette = "Dark2")+scale_fill_brewer(palette = "Dark2")+#ylim(0,11)+
    theme(legend.position="right", panel.grid.minor = element_blank(),panel.grid.major = element_blank()
  geom_smooth(method='lm', formula = y~x,se=F,size=0.5)
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```

```
plot.Elo.abundance

plot.Elo.ocurrence<-sum.elo.all%>%
ggplot(aes(x=(n_games),y=(rating),fill=Cycle.x,colour=Cycle.x))+
  geom_point(alpha=0.5,shape=21,size=1.5,color="black")+
  theme_bw()+labs(y="Elo-rating",x="Average  % read counts")+
  scale_color_brewer(palette = "Dark2")+scale_fill_brewer(palette = "Dark2")+#ylim(0,11)+
    theme(legend.position="right", panel.grid.minor = element_blank(),panel.grid.major = element_blank()
  geom_smooth(method='lm', formula = y~poly(x,3),se=F,size=0.5)
plot.Elo.ocurrence
```

#Subset data according Elo genus

```
### Adding rownames and sorting genus ranting to match relative
### abundance vector
rownames(sum.elo.6) = sum.elo.6$player_id
sum.elo.6 = sum.elo.6[order(rownames(sum.elo.6)), ]

# Subset Elo by the Genus
sum.elo.6 <- sum.elo.6[sum.elo.6$player_id %in% row.names(df.aa.pool.C6.norm),
    ][, c("player_id", "rating")]
```

3.0.2 Calculating Normalized Elo (competitive index)

```
tmp.elo <- sum.elo.6  # To use Cycle 6 Elo rating
# tmp.elo<-sum.elo # To use the addition of of the differences
tmp.elo$rating = tmp.elo$rating - abs(min(tmp.elo$rating))

# Recalculating the relative abundance
tmp.elo$rating = tmp.elo$rating/sum(tmp.elo$rating)

tmp.elo$rating <- tmp.elo$rating * median(Tmp.cells[1, 61:80])  #df1.aa.pool.C6.norm$Neutral.prediction
names(tmp.elo)[1:2] = c("Genus", "Elo.prediction")  #option 2
```

3.0.3 Neutral prediction

```
# Extracting the Pooled community (n=20)
df.aa.pool.C7 = df.aa.C6_C7[, 21:40]

df.Pooled.Cycle = cbind(row.names(df.aa.pool.C7), df.aa.pool.C7)
# Rename first column
colnames(df.Pooled.Cycle)[1] = "Genus"

df.Pooled.Cycle$Genus = as.factor(df.Pooled.Cycle$Genus)

# Order OTU levels by abundance in C6 pooled community
df.Pooled.Cycle$Genus = factor(df.Pooled.Cycle$Genus, df.Pooled.Cycle$Genus)

df.Pooled.Cycle.long <- melt(data = df.Pooled.Cycle, id = "Genus")
df.Pooled.Cycle.long$Genus = as.factor(df.Pooled.Cycle.long$Genus)

df.Pooled.Cycle.long = merge(df.Pooled.Cycle.long, df3[, 1:6], by = "Genus")
df.Pooled.Cycle.long = merge(df.Pooled.Cycle.long, df.aa.pool.C6.norm,
    by = "Genus")
df.Pooled.Cycle.long = merge(df.Pooled.Cycle.long, tmp.elo, by = "Genus")
```

4 Preparing the data for the t.test for Neutral dispersal

```r
# Normality test (Shapiro) Empty matrix
tmp.norm = data.frame(Genus = rep("NA", length(levels(df.Pooled.Cycle.long$Genus))),
    P_normality = NA, Mean = NA)

for (i in 1:length(levels(df.Pooled.Cycle.long$Genus))) {
    # Retrieving data by genus
    index.norm = df.Pooled.Cycle.long[df.Pooled.Cycle.long$Genus == levels(df.Pooled.Cycle.long$Genus)[i
        ]
    # Normality estimation
    tmp.norm$Genus[i] = levels(df.Pooled.Cycle.long$Genus)[i]
    try(tmp.norm$P_normality[i] <- as.numeric(ols_test_normality((index.norm$value))[[1]][2]),
        silent = T)
    tmp.norm$Mean[i] <- mean(index.norm$value)
}
```

```
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
```

```
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
## Warning in ks.test.default(y, "pnorm", mean(y), sd(y)): ties should not be
## present for the Kolmogorov-Smirnov test
```

```
tmp.norm$Normality = ifelse(tmp.norm$P_normality > 0.05, "TRUE", "FALSE")
print(tmp.norm)
```

```
##                                                  Genus  P_normality         Mean
## 1                                           Acidovorax 3.823713e-01 9.225802e+05
## 2                                            Aeromonas 8.470380e-01 4.039098e+05
## 3   Allorhizobium-Neorhizobium-Pararhizobium-Rhizobium 5.115567e-01 1.485744e+03
## 4                                         Aquabacterium 5.451532e-01 7.174167e+03
## 5                                                 Bosea 9.979464e-01 1.607920e+05
## 6                                        Bradyrhizobium           NA 0.000000e+00
## 7                                         Brevundimonas           NA 0.000000e+00
## 8                                           Caenimonas 2.850905e-02 1.006012e+03
## 9                               Candidatus Symbiobacter 2.950332e-05 1.344899e+02
## 10                                         Caulobacter           NA 0.000000e+00
## 11                                          Cellvibrio 6.024457e-01 1.113995e+05
## 12                                       Chitinibacter 3.800870e-04 3.390587e+02
## 13                                          Comamonas 3.592098e-01 2.548780e+03
## 14                                         Curvibacter 2.176139e-01 5.609584e+03
## 15                                       Dechloromonas 1.837022e-05 7.471659e+01
## 16                                             Deefgea 5.980961e-02 2.789419e+03
## 17                                             Delftia 4.322032e-01 4.118688e+03
## 18                                          Duganella 5.261038e-02 3.796994e+04
## 19                                          Emticicia 1.837022e-05 7.179663e+01
## 20                                  f_Caulobacteraceae           NA 0.000000e+00
## 21                                   f_Comamonadaceae 5.088813e-01 3.342369e+05
## 22                                    f_Rhodocyclaceae           NA 0.000000e+00
## 23                                      Ferribacterium           NA 0.000000e+00
## 24                                      Flavobacterium           NA 0.000000e+00
## 25                                         Leptothrix 9.928193e-01 9.903641e+03
## 26                                       Limnohabitans 5.565629e-01 2.006819e+04
## 27                                            Massilia 1.210505e-03 2.630866e+04
## 28                                        Methylibium 4.066147e-05 2.049124e+02
## 29                                             Ottowia 9.947686e-01 9.801408e+04
## 30                                         Paucibacter 8.745727e-01 3.945792e+04
## 31                                           Pelomonas 9.291570e-01 1.740252e+05
## 32                                         Polaromonas 3.923850e-01 4.199244e+03
## 33                                      Pseudarcicella           NA 0.000000e+00
## 34                                         Pseudomonas 6.552916e-01 1.415274e+06
## 35                                         Ramlibacter 9.777333e-01 5.654947e+04
## 36                                        Rheinheimera 2.958126e-05 1.389557e+02
## 37                                         Rhizobacter 7.909311e-01 4.154930e+03
```

```
## 38                                        Rhodoferax 8.389690e-01 6.760873e+04
## 39                                         Rivibacter 1.837022e-05 6.561319e+01
## 40                                      Sphaerotilus 9.228008e-01 2.635520e+03
## 41                                        Variovorax 5.042834e-01 1.040278e+04
##    Normality
## 1       TRUE
## 2       TRUE
## 3       TRUE
## 4       TRUE
## 5       TRUE
## 6       <NA>
## 7       <NA>
## 8      FALSE
## 9      FALSE
## 10      <NA>
## 11      TRUE
## 12     FALSE
## 13      TRUE
## 14      TRUE
## 15     FALSE
## 16      TRUE
## 17      TRUE
## 18      TRUE
## 19     FALSE
## 20      <NA>
## 21      TRUE
## 22      <NA>
## 23      <NA>
## 24      <NA>
## 25      TRUE
## 26      TRUE
## 27     FALSE
## 28     FALSE
## 29      TRUE
## 30      TRUE
## 31      TRUE
## 32      TRUE
## 33      <NA>
## 34      TRUE
## 35      TRUE
## 36     FALSE
## 37      TRUE
## 38      TRUE
## 39     FALSE
## 40      TRUE
## 41      TRUE
```

```
### -> Ok until here we have identified the Genera normally
### distributes and those that are only Os.

# Run One sample t-test or Wilcoxon for normal and not normal,
# respectively.
res.t.test = data.frame(test.name = NA, Neutral.p.value = matrix(NA, nrow = length(levels(df.Pooled.Cycl
    Elo.p.value = NA)
```

```r
rownames(res.t.test) = levels(df.Pooled.Cycle.long$Genus)

for (i in 1:length(levels(df.Pooled.Cycle.long$Genus))) {
    tmp.data = df.Pooled.Cycle.long[df.Pooled.Cycle.long$Genus == levels(df.Pooled.Cycle.long$Genus)[i]
        ]
    if (is.na(tmp.norm$Normality[tmp.norm$Genus == levels(df.Pooled.Cycle.long$Genus)[i]]) ==
        T) {
        res.t.test$test.name[i] = "None"
    } else if (tmp.norm$Normality[tmp.norm$Genus == levels(df.Pooled.Cycle.long$Genus)[i]] ==
        FALSE) {
        res.tmp <- wilcox.test((tmp.data$value), mu = unique((tmp.data$Neutral.prediction)),
            alternative = "two.sided")
        res.t.test$Neutral.p.value[i] = res.tmp$p.value
        res.t.test$test.name[i] = "Wilcoxon.test"
    } else if (tmp.norm$Normality[tmp.norm$Genus == levels(df.Pooled.Cycle.long$Genus)[i]] ==
        TRUE) {
        res.tmp <- t.test((tmp.data$value), mu = unique((tmp.data$Neutral.prediction)),
            alternative = "two.sided")
        res.t.test$Neutral.p.value[i] = res.tmp$p.value
        res.t.test$test.name[i] = "t.test"
    }
}
```

```
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Neutral.prediction)), : cannot compute exact p-value with ties

## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Neutral.prediction)), : cannot compute exact p-value with ties
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Neutral.prediction)), : cannot compute exact p-value with ties
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Neutral.prediction)), : cannot compute exact p-value with ties
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Neutral.prediction)), : cannot compute exact p-value with ties
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Neutral.prediction)), : cannot compute exact p-value with ties
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Neutral.prediction)), : cannot compute exact p-value with ties
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Neutral.prediction)), : cannot compute exact p-value with ties
```

```r
for (i in 1:length(levels(df.Pooled.Cycle.long$Genus))) {
    tmp.data = df.Pooled.Cycle.long[df.Pooled.Cycle.long$Genus == levels(df.Pooled.Cycle.long$Genus)[i]
        ]
    if (is.na(tmp.norm$Normality[tmp.norm$Genus == levels(df.Pooled.Cycle.long$Genus)[i]]) ==
        T) {
        res.t.test$test.name[i] = "None"
    } else if (tmp.norm$Normality[tmp.norm$Genus == levels(df.Pooled.Cycle.long$Genus)[i]] ==
        FALSE) {
        res.tmp <- wilcox.test((tmp.data$value), mu = unique((tmp.data$Elo.prediction)),
            alternative = "two.sided")
        res.t.test$Elo.p.value[i] = res.tmp$p.value
        res.t.test$test.name[i] = "Wilcoxon.test"
```

```
    } else if (tmp.norm$Normality[tmp.norm$Genus == levels(df.Pooled.Cycle.long$Genus)[i]] ==
        TRUE) {
        res.tmp <- t.test((tmp.data$value), mu = unique((tmp.data$Elo.prediction)),
            alternative = "two.sided")
        res.t.test$Elo.p.value[i] = res.tmp$p.value
        res.t.test$test.name[i] = "t.test"
    }
}
```

```
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Elo.prediction)), : cannot compute exact p-value with ties
```

```
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Elo.prediction)), : cannot compute exact p-value with ties
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Elo.prediction)), : cannot compute exact p-value with ties
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Elo.prediction)), : cannot compute exact p-value with ties
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Elo.prediction)), : cannot compute exact p-value with ties
```

```
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Elo.prediction)), : cannot compute exact p-value with zeroes
```

```
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Elo.prediction)), : cannot compute exact p-value with ties
## Warning in wilcox.test.default((tmp.data$value), mu =
## unique((tmp.data$Elo.prediction)), : cannot compute exact p-value with ties
```

#Bonferroni

```
res.t.test$Neutral.p.value.adj = p.adjust(res.t.test$Neutral.p.value, method = "bonferroni")
res.t.test$Elo.p.value.adj = p.adjust(res.t.test$Elo.p.value, method = "bonferroni")
res.t.test$Genus <- row.names(res.t.test)
```

#Step2 determine the best predictor

```
res.t.test = merge(res.t.test, unique(df.Pooled.Cycle.long[, c(-2, -3)]),
    by = "Genus", all.x = T)
# Get mean values of C7
tmp.mean.C7 <- aggregate(value ~ Genus, data = df.Pooled.Cycle.long, mean)
names(tmp.mean.C7) <- c("Genus", "MeanValue")
# combine dfs
res.t.test <- merge(res.t.test, tmp.mean.C7, by = "Genus")

res.t.test$best.predictor = ifelse(res.t.test$Neutral.p.value > res.t.test$Elo.p.value,
    "Neutral", "Competition Effect")

res.t.test$best.predictor[res.t.test$Neutral.p.value.adj < 0.0499 & res.t.test$Elo.p.value.adj <
    0.0499 & res.t.test$MeanValue > res.t.test$Neutral.prediction] = "Outperformed"
```

```
res.t.test$best.predictor[res.t.test$Neutral.p.value.adj < 0.0499 & res.t.test$Elo.p.value.adj <
    0.0499 & res.t.test$MeanValue < res.t.test$Elo.prediction] = "Underperformed"

res.t.test$best.predictor[is.na(res.t.test$best.predictor)] = "Excluded"
```

5 Final figure combined results

```
df.Pooled.Cycle.long = merge(df.Pooled.Cycle.long, res.t.test, by = "Genus")
# Edit Genus names
df.Pooled.Cycle.long$Genus <- as.character(df.Pooled.Cycle.long$Genus)
df.Pooled.Cycle.long$Genus[which(df.Pooled.Cycle.long$Genus == "Allorhizobium-Neorhizobium-Pararhizobiu
df.Pooled.Cycle.long$Genus <- str_replace(df.Pooled.Cycle.long$Genus, "f_",
    "uncl_")
df.Pooled.Cycle.long$Genus <- as.factor(df.Pooled.Cycle.long$Genus)

df.Pooled.Cycle.long$Genus = factor(df.Pooled.Cycle.long$Genus, unique(df.Pooled.Cycle.long$Genus[order

cbp2 <- c("#999999", "#E69F00", "#56B4E9", "#009E73", "#293352")
df.Pooled.Cycle.long$best.predictor = as.factor(df.Pooled.Cycle.long$best.predictor)
df.Pooled.Cycle.long$best.predictor = factor(df.Pooled.Cycle.long$best.predictor,
    c("Neutral", "Competition Effect", "Outperformed", "Underperformed",
        "Excluded"))
```

```
constant=0+1
plot.coale=ggplot(df.Pooled.Cycle.long, aes(x=as.numeric(Genus),y =(value+constant)/1000000)) +
  geom_point(aes(fill=best.predictor),size=1,shape=21,alpha=0.5,stroke=0)+
  geom_line(aes(as.numeric(Genus),(Elo.prediction.x+constant)/1000000),color=cbp2[2],size=.6,alpha=1)+
  geom_line(aes(as.numeric(Genus),(Neutral.prediction.x+constant)/1000000),color=cbp2[1],size=.6,alpha=
   scale_colour_manual(values=c(cbp2),name="")+
   scale_fill_manual(values=c(cbp2),name="Predictions models")+
  theme_bw()+labs(y="Cell numbers (x10<sup>6</sup> Cells mL<sup>-1</sup>)",x=NULL)+
  theme(panel.grid.major = element_blank(),
    panel.grid.minor = element_blank())+
  coord_flip()+
  scale_x_reverse(breaks=seq(1,41,1),labels=levels(df.Pooled.Cycle.long$Genus),expand = c(0.01, 0.01))+
  guides(color = guide_legend(override.aes = list(size=4)))+
  theme(plot.margin = unit(c(0, 0, 0, 0), "cm"))+
  theme(text= element_text(size=7),legend.text = element_text(size = 5.5), legend.spacing.x =unit(0.1,
        legend.position=c(.7,.2),legend.key.size = unit(0.3, "cm"))+
  guides(fill = guide_legend(override.aes = list(alpha = 1,size=2) ) )+
  theme(plot.margin = margin(t = 0.5, r = 0.5, b =0, l = 0, "cm"))+#+scale_y_continuous(trans="log10")+
  theme(axis.title.x = element_markdown())+
  theme(#legend.title=element_blank(),
      legend.margin = margin(0, 0, 0, 0),
      legend.spacing.x = unit(0, "mm"),
      legend.spacing.y = unit(0, "mm"))
```

```
## Warning: A numeric ‘legend.position‘ argument in ‘theme()‘ was deprecated in ggplot2
## 3.5.0.
## i Please use the ‘legend.position.inside‘ argument of ‘theme()‘ instead.
```

```
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_lifecycle_warnings()' to see where this warning was
## generated.
```

5.1 Estimate contribution of each one

```
tmp.contr = aggregate(value ~ best.predictor + variable, data = df.Pooled.Cycle.long,
    sum)

# Printing figure
plot.contribution <- tmp.contr %>%
    ggplot(aes(best.predictor, value/3967300, colour = best.predictor)) +
    geom_boxplot(outlier.shape = NA, alpha = 0.5, size = 0.25) + scale_colour_manual(values = c(cbp2),
    name = "") + xlab("") + ylab("% Abundance") + geom_jitter(width = 0.2,
    shape = 21, size = 0.7, stroke = 0.25) + theme_bw() + theme(panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()) + theme(axis.text.y = element_text(hjust = 1),
    legend.position = "none", axis.text.x = element_text(angle = 45, vjust = 1,
        hjust = 1), text = element_text(size = 7)) + theme(plot.margin = margin(t = 0,
    r = 0, b = 0, l = 0, "cm"))

tmp.contr <- aggregate(value ~ best.predictor + variable, data = df.Pooled.Cycle.long,
    sum)
summary.contr <- aggregate(value ~ best.predictor, data = tmp.contr, mean)
summary.contr$value <- summary.contr$value/3967300
tibble(summary.contr)
```

```
## # A tibble: 5 x 2
##   best.predictor      value
##   <fct>               <dbl>
## 1 Neutral            0.448
## 2 Competition Effect 0.113
## 3 Outperformed       0.357
## 4 Underperformed     0.0720
## 5 Excluded           0
```

6 Barplot with genus count per factor

```
tmp.contr=aggregate(value~best.predictor+variable,data=df.Pooled.Cycle.long,length)
plot.contribution.numbers<-tmp.contr%>%
  ggplot(aes(value/20,best.predictor))+
  geom_col(aes(fill=best.predictor),color=NA)+#ylim(0,1)+
  scale_fill_manual(values=c(cbp2),name="")+
  labs(x="Number of Genera",y="")+theme_bw()+
 theme(legend.position="none",panel.grid.major = element_blank(),
    panel.grid.minor = element_blank())+
  coord_flip()+theme(text =element_text(size=7),
  axis.text.x = element_blank())+xlim(0,22)+
    theme(plot.margin = margin(t = 0, r = 0, b =0, l = 0, "cm"))
```

```
library(ComplexHeatmap)
```

```
## Loading required package: grid
```

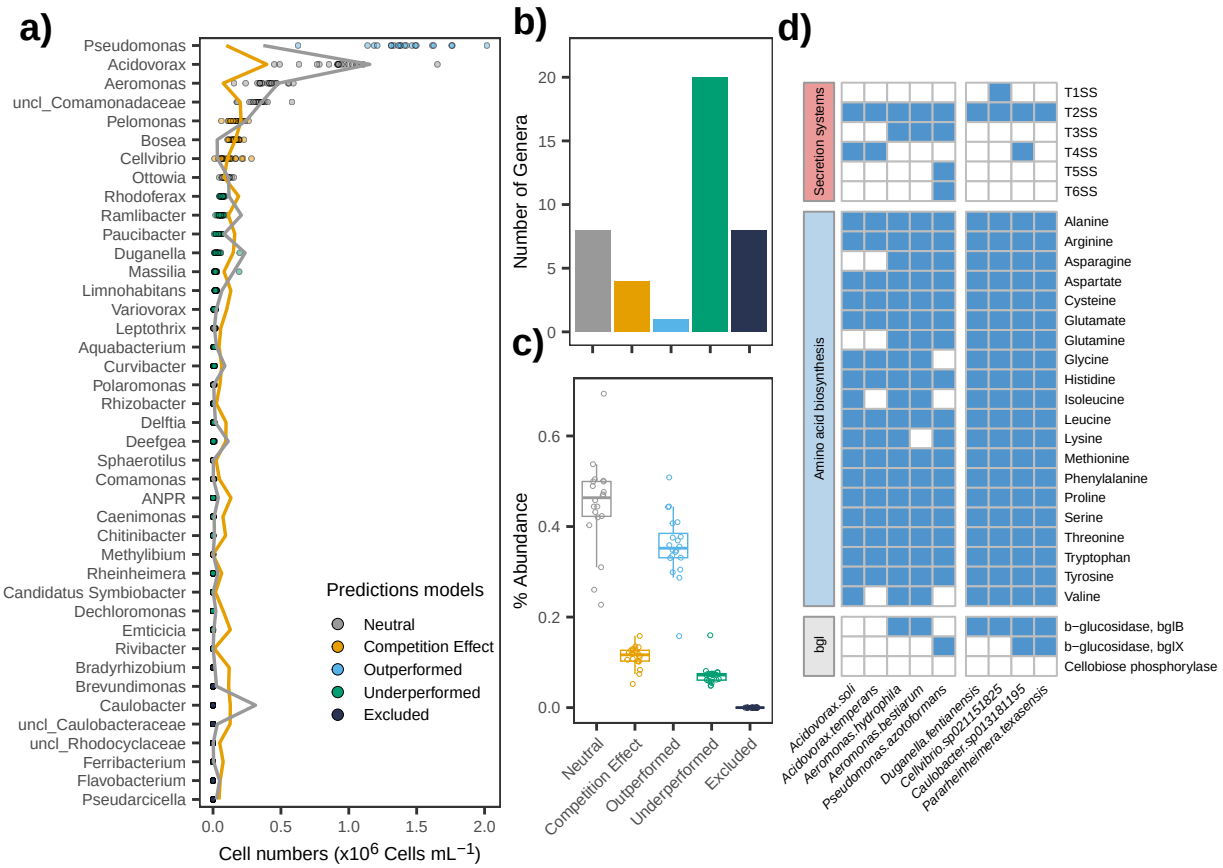
```
## ========================================
## ComplexHeatmap version 2.15.4
## Bioconductor page: http://bioconductor.org/packages/ComplexHeatmap/
## Github page: https://github.com/jokergoo/ComplexHeatmap
## Documentation: http://jokergoo.github.io/ComplexHeatmap-reference
##
## If you use it in published research, please cite either one:
## - Gu, Z. Complex Heatmap Visualization. iMeta 2022.
## - Gu, Z. Complex heatmaps reveal patterns and correlations in multidimensional
##     genomic data. Bioinformatics 2016.
##
##
## The new InteractiveComplexHeatmap package can directly export static
## complex heatmaps into an interactive Shiny app with zero effort. Have a try!
##
## This message can be suppressed by:
##   suppressPackageStartupMessages(library(ComplexHeatmap))
## ========================================
```

```
bk2 <- c(0, 1)
colors2 <- c("white", "steelblue3")
my_heatmap = ComplexHeatmap::pheatmap(as.matrix(df.gene.cat[, 3:11]), color = colors2,
    breaks = bk2, cellwidth = 9, cellheight = 7.7, border_color = "grey",
    left_annotation = rowAnnotation(foo = anno_block(gp = gpar(fontsize_number = 4.8,
        fontface = "plain", fill = c("red3", "steelblue3", "grey"), alpha = 0.4),
        labels = c("Secretion systems", "Amino acid biosynthesis", " bgl"),
        labels_gp = gpar(col = "black", fontsize = 4.5, fontface = "plain"),
        width = unit(0.4, "cm"))), fontface_col = "italic", cluster_rows = F,
    cluster_cols = F, angle_col = "45", fontsize_col = 5, fontsize = 5,
    legend = F, annotation_legend = T, gaps_col = c(5), gaps_row = c(6,
        26))

test_plot <- my_heatmap %>%
    draw(padding = unit(c(0, 0, 0, 0), "mm")) %>%
    grid.grabExpr()
```

```
## pdf
##   2
```

```
plot_grid(plot.coale, NULL, plot_grid(plot.contribution.numbers, plot.contribution,
    ncol = 1, rel_heights = c(0.7, 1), labels = c("b)", "c)"), label_x = -0.07,
    label_y = 1.13), NULL, plot_grid(NULL, test_plot, rel_heights = c(0.5,
    0.5)), ncol = 5, align = "hv", axis = "rigth", rel_widths = c(0.45,
    -0.021, 0.25, -0.33, 0.7), labels = c("a)", "", "", "d)"), label_x = c(0,
    0, 0, 0.13))
```


a)
Pseudomonas
Acidovorax
Aeromonas
uncl_Comamonadaceae
Pelomonas
Bosea
Cellvibrio
Ottowia
Rhodoferax
Ramlibacter
Paucibacter
Duganella
Massilia
Limnohabitans
Variovorax
Leptothrix
Aquabacterium
Curvibacter
Polaromonas
Rhizobacter
Delftia
Deefgea
Sphaerotilus
Comamonas
ANPR
Caenimonas
Chitinibacter
Methylibium
Rheinheimera
Candidatus Symbiobacter
Dechloromonas
Emticicia
Rivibacter
Bradyrhizobium
Brevundimonas
Caulobacter
uncl_Caulobacteraceae
uncl_Rhodocyclaceae
Ferribacterium
Flavobacterium
Pseudarcicella
0.0
0.5
1.0
1.5
2.0
Cell numbers (x10^6 Cells mL^-1)
Predictions models
Neutral
Competition Effect
Outperformed
Underperformed
Excluded
b)
Number of Genera
0
5
10
15
20
c)
% Abundance
0.0
0.2
0.4
0.6
Neutral
Competition Effect
Outperformed
Underperformed
Excluded
d)
Secretion systems
T1SS
T2SS
T3SS
T4SS
T5SS
T6SS
Amino acid biosynthesis
Alanine
Arginine
Asparagine
Aspartate
Cysteine
Glutamate
Glutamine
Glycine
Histidine
Isoleucine
Leucine
Lysine
Methionine
Phenylalanine
Proline
Serine
Threonine
Tryptophan
Tyrosine
Valine
bgl
b−glucosidase, bglB
b−glucosidase, bglX
Cellobiose phosphorylase
Acidovorax.soli
Acidovorax.temperans
Aeromonas.hydrophila
Aeromonas.bestiarum
Pseudomonas.azotoformans
Duganella.fentianensis
Cellvibrio.sp0211151825
Caulobacter.sp0131811195
Paraheinheimera.texasensis