

Chemo.05.GenomicTraits

Angel Rain & Sara Beier

9/9/2022

Contents

0.1	Load packages	1
1	Setting up workspace	2
1.1	Load metadata	2
1.2	Load predicted traits	3
2	Community indexes	5
2.1	Calculation of the Community weighed mean (CWM)	5
2.2	Estimate alpha diversity (Shannon diversity index)	8
2.3	Community weighed means (CWMs)	9
3	Community trait distribution during the experiment	10
3.1	Dataframe and format trait CWM values	10
3.2	Summarizing data replicate mean values	11
3.3	Test for normality and homogeneity of variances	11
3.4	Repeated measurement ANOVA	13
4	Paired-test per Genomic trait	14
4.1	Manuscript Figure 3	15

0.1 Load packages

```
rm(list = ls())
library(phyloseq)
library(reshape2)
library(ggplot2)
library(vegan)
library(rstatix) #Homogeneity of variance test
library(olsrr) # test_normality
library(egg) # Tag figures
```

```
library(rstatix) #Tibble function
library(microViz) #TO use re_order function
library(kableExtra) #Table format

#####
cbbPalette <- c("#000000", "#E69F00", "#56B4E9", "#009E73", "#F0E442", "#0072B2",
               "#D55E00", "#CC79A7")
```

1 Setting up workspace

1.1 Load metadata

```
# Create metadata for experimental setup
schema <- data.frame(sample.ID = paste0("C10-", rep(1:9, each = 12), "-"), sprintf("%02d",
1:12)), T = rep(1:9, each = 12), Chem.ID = sprintf("%02d", 1:12), DOM = rep(c("L",
"H"), each = 6), Sal = c("C", "D"))

schema$T = as.factor(schema$T)
schema$DOM = as.factor(schema$DOM)
schema$Sal = as.factor(schema$Sal)

tibble(schema)
```

```
## # A tibble: 108 x 5
##   sample.ID T      Chem.ID DOM   Sal
##   <chr>     <fct> <chr>  <fct> <fct>
## 1 C10-1-01 1      01     L     C
## 2 C10-1-02 1      02     L     D
## 3 C10-1-03 1      03     L     C
## 4 C10-1-04 1      04     L     D
## 5 C10-1-05 1      05     L     C
## 6 C10-1-06 1      06     L     D
## 7 C10-1-07 1      07     H     C
## 8 C10-1-08 1      08     H     D
## 9 C10-1-09 1      09     H     C
## 10 C10-1-10 1      10     H     D
## # ... with 98 more rows
```

#Loading phyloseq object with ASV count table from #dada2

```
# Loading phyloseq object from #dada2
ps <- readRDS("../data/dada2.output/chem.ps.rds")
# Phyloseq object contain abundance table, sample information, taxonomic
# information and the phylogenetic tree

# Loadgin phylogenetic tree
chem.tree = read_tree("../data/dada2.output/dada-chem.GTR2")
phy_tree(ps) <- chem.tree #Adding phylo-tree to the phyloseq object

# Phyloseq object contain abundance table, sample information, taxonomic
```

```
# information and the phylogenetic tree
ps
```

```
## phyloseq-class experiment-level object
## otu_table() OTU Table: [ 1447 taxa and 110 samples ]
## sample_data() Sample Data: [ 110 samples by 3 sample variables ]
## tax_table() Taxonomy Table: [ 1447 taxa by 7 taxonomic ranks ]
## phy_tree() Phylogenetic Tree: [ 1447 tips and 1445 internal nodes ]
```

```
# Removing initial inoculum samples for downstream analysis
ps = subset_samples(ps, sample.ID != "C10-0-LL" & sample.ID != "C10-0-HH")
ps
```

```
## phyloseq-class experiment-level object
## otu_table() OTU Table: [ 1447 taxa and 108 samples ]
## sample_data() Sample Data: [ 108 samples by 3 sample variables ]
## tax_table() Taxonomy Table: [ 1447 taxa by 7 taxonomic ranks ]
## phy_tree() Phylogenetic Tree: [ 1447 tips and 1445 internal nodes ]
```

1.2 Load predicted traits

```
# Resilience related genes load RRN predicted from rrnDB tree and trait data
pic.16s.custom <- read.table("../data/picrust2/trait.predicted/pic.chemo10.16S_predicted_custom_tree.txt",
  header = T)
tibble(pic.16s.custom)
```

```
## # A tibble: 1,447 x 3
##   sequence                X16S_rRNA_Count metadata_NSTI
##   <chr>                  <int>          <dbl>
## 1 SV_1000_Sphingomonadales      1      0.0377
## 2 SV_1001_Rhodospirillales      4      0.0525
## 3 SV_1002_Enterobacterales     12      0.283
## 4 SV_1003_NA                    2      0.686
## 5 SV_1004_Enterobacterales      5      0.0286
## 6 SV_1005_Flavobacteriales      3      0.263
## 7 SV_1006_Rhodospirillales      4      0.630
## 8 SV_1007_Flavobacteriales      3      0.0555
## 9 SV_1008_Enterobacterales      5      0.129
## 10 SV_1009_Rhodobacterales      1      0.0286
## # ... with 1,437 more rows
```

```
# load generation time predicted from PICRUST2 default tree and database
pic.d.gRodon.default <- read.table("../data/picrust2/trait.predicted/pic.d.gRodon.retransformed.txt",
  header = T)
tibble(pic.d.gRodon.default)
```

```
## # A tibble: 4,298 x 3
##   sequence                d.gRodon metadata_NSTI
##   <chr>                  <dbl>          <dbl>
```

```
## 1 2228664026      13.3      0.0395
## 2 2236661015      10.9      0.00632
## 3 2264265199      17.8      0.533
## 4 2264813001-cluster 11.7      1.26
## 5 2264867162      14.0      0.504
## 6 2265123003       5.37      0.120
## 7 2500069000       1.19      0.0820
## 8 2501846311       1.43      0.000002
## 9 2504557005      10.2      0.386
## 10 2504756036      3.29      0.00523
## # ... with 4,288 more rows
```

```
# Resistance-related genes load %TF predicted from PICRUST2 default tree and
# database
pic.TFr.default <- read.table("../data/picrust2/trait.predicted/pic.TF_perc.retransformed.txt",
  header = T)
tibble(pic.TFr.default)
```

```
## # A tibble: 4,298 x 3
##   sequence      TF_perc metadata_NSTI
##   <chr>         <dbl>      <dbl>
## 1 2228664026     1.56      0.0395
## 2 2236661015     1.18      0.00632
## 3 2264265199     1.79      0.533
## 4 2264813001-cluster 1.79      1.26
## 5 2264867162     1.22      0.504
## 6 2265123003     1.23      0.120
## 7 2500069000     2.95      0.0820
## 8 2501846311     0.798     0.000002
## 9 2504557005     1.45      0.386
## 10 2504756036     1.43      0.00523
## # ... with 4,288 more rows
```

```
# load genome size predicted from PICRUST2 default tree and database
pic.gs.default <- read.table("../data/picrust2/trait.predicted/pic.genome.size.retransformed.txt",
  header = T)
tibble(pic.gs.default)
```

```
## # A tibble: 3,687 x 3
##   sequence      genome.size metadata_NSTI
##   <chr>         <dbl>      <dbl>
## 1 2228664026     2.45      0.0395
## 2 2236661015     1.41      0.00632
## 3 2264265199     1.68      0.533
## 4 2264813001-cluster 2.14      1.26
## 5 2264867162     2.88      0.504
## 6 2265123003     2.43      0.120
## 7 2500069000     2.05      0.0820
## 8 2501846311     2.39      0.000002
## 9 2504557005     4.87      0.386
## 10 2504756036     2.14      0.00523
## # ... with 3,677 more rows
```

2 Community indexes

2.1 Calculation of the Community weighted mean (CWM)

CWMs were obtained by summing predicted and abundance-weighted trait-values for all ASVs in each community

2.1.1 Relative abundance data

```
# Rarefy by minimum read numbers and transform to relative data
ps = rarefy_even_depth(ps, min(rowSums(otu_table(ps))), rngseed = 1, replace = F,
  trimOTUs = F)
```

```
## 'set.seed(1)' was used to initialize repeatable random subsampling.
```

```
## Please record this for your records so others can reproduce.
```

```
## Try 'set.seed(1); .Random.seed' for the full vector
```

```
## ...
```

```
# Estimating relative abundance
rOTUdf.rar <- prop.table(otu_table(ps), 1)
```

```
# New phyloseq-project with rarefied ASV table
otu_table(ps) <- otu_table(rOTUdf.rar, taxa_are_rows = FALSE)
ps
```

```
## phyloseq-class experiment-level object
## otu_table() OTU Table: [ 1447 taxa and 108 samples ]
## sample_data() Sample Data: [ 108 samples by 3 sample variables ]
## tax_table() Taxonomy Table: [ 1447 taxa by 7 taxonomic ranks ]
## phy_tree() Phylogenetic Tree: [ 1447 tips and 1445 internal nodes ]
```

```
# Keep ASVs with prevalence equivalent to more 0 reads
ps <- prune_taxa(taxa_sums(ps) > 0, ps)
ps
```

```
## phyloseq-class experiment-level object
## otu_table() OTU Table: [ 973 taxa and 108 samples ]
## sample_data() Sample Data: [ 108 samples by 3 sample variables ]
## tax_table() Taxonomy Table: [ 973 taxa by 7 taxonomic ranks ]
## phy_tree() Phylogenetic Tree: [ 973 tips and 971 internal nodes ]
```

```
# Setting up metadata
```

```
# Samples in phyloseq object did not correspond to the metadata (schema), so we
# proceed to reorder ps-data base in the schema$sample.ID ##SARA: ???; samples
# from chem3?
```

```

new_order <- schema$sample.ID
ps = ps %>%
  ps_reorder(new_order) #MicroViz package

# Extract ASV count table
counts = t(otu_table(ps))

```

2.1.2 Remove ASVs without close relatives in the default reference database (NSTI<1)

```

counts.s.default <- counts[row.names(counts) %in% pic.gs.default[pic.gs.default$metadata_NSTI <
  1, 1], ] #extract ASVs with NSTI<1 in default reference database
colSums(counts.s.default) #check which proportion of sequences is left after removing ASVs with NSTI<1

```

```

## C10-1-01 C10-1-02 C10-1-03 C10-1-04 C10-1-05 C10-1-06 C10-1-07 C10-1-08
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-1-09 C10-1-10 C10-1-11 C10-1-12 C10-2-01 C10-2-02 C10-2-03 C10-2-04
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-2-05 C10-2-06 C10-2-07 C10-2-08 C10-2-09 C10-2-10 C10-2-11 C10-2-12
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.9998938
## C10-3-01 C10-3-02 C10-3-03 C10-3-04 C10-3-05 C10-3-06 C10-3-07 C10-3-08
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-3-09 C10-3-10 C10-3-11 C10-3-12 C10-4-01 C10-4-02 C10-4-03 C10-4-04
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-4-05 C10-4-06 C10-4-07 C10-4-08 C10-4-09 C10-4-10 C10-4-11 C10-4-12
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-5-01 C10-5-02 C10-5-03 C10-5-04 C10-5-05 C10-5-06 C10-5-07 C10-5-08
## 1.0000000 1.0000000 1.0000000 0.9998938 1.0000000 1.0000000 1.0000000 1.0000000
## C10-5-09 C10-5-10 C10-5-11 C10-5-12 C10-6-01 C10-6-02 C10-6-03 C10-6-04
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-6-05 C10-6-06 C10-6-07 C10-6-08 C10-6-09 C10-6-10 C10-6-11 C10-6-12
## 0.9998938 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-7-01 C10-7-02 C10-7-03 C10-7-04 C10-7-05 C10-7-06 C10-7-07 C10-7-08
## 0.9997875 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-7-09 C10-7-10 C10-7-11 C10-7-12 C10-8-01 C10-8-02 C10-8-03 C10-8-04
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-8-05 C10-8-06 C10-8-07 C10-8-08 C10-8-09 C10-8-10 C10-8-11 C10-8-12
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-9-01 C10-9-02 C10-9-03 C10-9-04 C10-9-05 C10-9-06 C10-9-07 C10-9-08
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
## C10-9-09 C10-9-10 C10-9-11 C10-9-12
## 1.0000000 1.0000000 1.0000000 1.0000000

```

```

min(colSums(counts.s.default))

```

```

## [1] 0.9997875

```

```

counts.s.rel.default <- as.data.frame.matrix(prop.table(t(t(counts.s.default)), 2)) #re-normalize remain
colSums(counts.s.rel.default) #should sum up again to 1

```

```

## C10-1-01 C10-1-02 C10-1-03 C10-1-04 C10-1-05 C10-1-06 C10-1-07 C10-1-08

```

```
##      1      1      1      1      1      1      1      1
## C10-1-09 C10-1-10 C10-1-11 C10-1-12 C10-2-01 C10-2-02 C10-2-03 C10-2-04
##      1      1      1      1      1      1      1      1
## C10-2-05 C10-2-06 C10-2-07 C10-2-08 C10-2-09 C10-2-10 C10-2-11 C10-2-12
##      1      1      1      1      1      1      1      1
## C10-3-01 C10-3-02 C10-3-03 C10-3-04 C10-3-05 C10-3-06 C10-3-07 C10-3-08
##      1      1      1      1      1      1      1      1
## C10-3-09 C10-3-10 C10-3-11 C10-3-12 C10-4-01 C10-4-02 C10-4-03 C10-4-04
##      1      1      1      1      1      1      1      1
## C10-4-05 C10-4-06 C10-4-07 C10-4-08 C10-4-09 C10-4-10 C10-4-11 C10-4-12
##      1      1      1      1      1      1      1      1
## C10-5-01 C10-5-02 C10-5-03 C10-5-04 C10-5-05 C10-5-06 C10-5-07 C10-5-08
##      1      1      1      1      1      1      1      1
## C10-5-09 C10-5-10 C10-5-11 C10-5-12 C10-6-01 C10-6-02 C10-6-03 C10-6-04
##      1      1      1      1      1      1      1      1
## C10-6-05 C10-6-06 C10-6-07 C10-6-08 C10-6-09 C10-6-10 C10-6-11 C10-6-12
##      1      1      1      1      1      1      1      1
## C10-7-01 C10-7-02 C10-7-03 C10-7-04 C10-7-05 C10-7-06 C10-7-07 C10-7-08
##      1      1      1      1      1      1      1      1
## C10-7-09 C10-7-10 C10-7-11 C10-7-12 C10-8-01 C10-8-02 C10-8-03 C10-8-04
##      1      1      1      1      1      1      1      1
## C10-8-05 C10-8-06 C10-8-07 C10-8-08 C10-8-09 C10-8-10 C10-8-11 C10-8-12
##      1      1      1      1      1      1      1      1
## C10-9-01 C10-9-02 C10-9-03 C10-9-04 C10-9-05 C10-9-06 C10-9-07 C10-9-08
##      1      1      1      1      1      1      1      1
## C10-9-09 C10-9-10 C10-9-11 C10-9-12
##      1      1      1      1
```

2.1.3 Remove ASVs without close relatives in the custom reference database (NSTI<1)

```
counts.s.custom <- counts[row.names(counts) %in% pic.16s.custom[pic.16s.custom$metadata_NSTI <
  1, 1], ] #extract ASVs with NSTI<1 (= ASVs with no close relative in the picrust2 reference databa.
colSums(counts.s.custom) #check which proportion of sequences is left after removing ASVs with NSTI<1
```

```
## C10-1-01 C10-1-02 C10-1-03 C10-1-04 C10-1-05 C10-1-06 C10-1-07 C10-1-08
## 0.9989375 0.9993625 0.9993625 0.9993625 0.9998938 0.9997875 1.0000000 0.9998938
## C10-1-09 C10-1-10 C10-1-11 C10-1-12 C10-2-01 C10-2-02 C10-2-03 C10-2-04
## 1.0000000 0.9997875 0.9996813 0.9997875 1.0000000 0.9996813 1.0000000 0.9998938
## C10-2-05 C10-2-06 C10-2-07 C10-2-08 C10-2-09 C10-2-10 C10-2-11 C10-2-12
## 1.0000000 0.9998938 1.0000000 0.9998938 0.9997875 1.0000000 1.0000000 1.0000000
## C10-3-01 C10-3-02 C10-3-03 C10-3-04 C10-3-05 C10-3-06 C10-3-07 C10-3-08
## 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.9992563 0.9997875
## C10-3-09 C10-3-10 C10-3-11 C10-3-12 C10-4-01 C10-4-02 C10-4-03 C10-4-04
## 0.9998938 0.9998938 0.9998938 0.9996813 1.0000000 1.0000000 1.0000000 1.0000000
## C10-4-05 C10-4-06 C10-4-07 C10-4-08 C10-4-09 C10-4-10 C10-4-11 C10-4-12
## 1.0000000 1.0000000 0.9989375 0.9997875 0.9995750 0.9994688 0.9997875 0.9995750
## C10-5-01 C10-5-02 C10-5-03 C10-5-04 C10-5-05 C10-5-06 C10-5-07 C10-5-08
## 1.0000000 1.0000000 0.9998938 1.0000000 1.0000000 1.0000000 0.9993625 0.9996813
## C10-5-09 C10-5-10 C10-5-11 C10-5-12 C10-6-01 C10-6-02 C10-6-03 C10-6-04
## 0.9998938 0.9997875 0.9997875 0.9994688 1.0000000 1.0000000 1.0000000 1.0000000
## C10-6-05 C10-6-06 C10-6-07 C10-6-08 C10-6-09 C10-6-10 C10-6-11 C10-6-12
## 0.9998938 1.0000000 0.9991500 0.9994688 0.9997875 1.0000000 0.9994688 0.9997875
```

```
## C10-7-01 C10-7-02 C10-7-03 C10-7-04 C10-7-05 C10-7-06 C10-7-07 C10-7-08
## 1.0000000 1.0000000 1.0000000 1.0000000 0.9998938 1.0000000 1.0000000 0.9986188
## C10-7-09 C10-7-10 C10-7-11 C10-7-12 C10-8-01 C10-8-02 C10-8-03 C10-8-04
## 1.0000000 0.9996813 1.0000000 1.0000000 0.9994688 0.9998938 1.0000000 1.0000000
## C10-8-05 C10-8-06 C10-8-07 C10-8-08 C10-8-09 C10-8-10 C10-8-11 C10-8-12
## 1.0000000 1.0000000 0.9991500 0.9993625 0.9997875 0.9997875 1.0000000 1.0000000
## C10-9-01 C10-9-02 C10-9-03 C10-9-04 C10-9-05 C10-9-06 C10-9-07 C10-9-08
## 0.9997875 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000 0.9997875 0.9987250
## C10-9-09 C10-9-10 C10-9-11 C10-9-12
## 1.0000000 0.9996813 1.0000000 1.0000000
```

```
min(colSums(counts.s.custom))
```

```
## [1] 0.9986188
```

```
counts.s.rel.custom <- as.data.frame.matrix(prop.table(t(t(counts.s.custom)), 2)) #re-normalize remain
colSums(counts.s.rel.custom) #should sum up again to 1
```

```
## C10-1-01 C10-1-02 C10-1-03 C10-1-04 C10-1-05 C10-1-06 C10-1-07 C10-1-08
## 1 1 1 1 1 1 1 1
## C10-1-09 C10-1-10 C10-1-11 C10-1-12 C10-2-01 C10-2-02 C10-2-03 C10-2-04
## 1 1 1 1 1 1 1 1
## C10-2-05 C10-2-06 C10-2-07 C10-2-08 C10-2-09 C10-2-10 C10-2-11 C10-2-12
## 1 1 1 1 1 1 1 1
## C10-3-01 C10-3-02 C10-3-03 C10-3-04 C10-3-05 C10-3-06 C10-3-07 C10-3-08
## 1 1 1 1 1 1 1 1
## C10-3-09 C10-3-10 C10-3-11 C10-3-12 C10-4-01 C10-4-02 C10-4-03 C10-4-04
## 1 1 1 1 1 1 1 1
## C10-4-05 C10-4-06 C10-4-07 C10-4-08 C10-4-09 C10-4-10 C10-4-11 C10-4-12
## 1 1 1 1 1 1 1 1
## C10-5-01 C10-5-02 C10-5-03 C10-5-04 C10-5-05 C10-5-06 C10-5-07 C10-5-08
## 1 1 1 1 1 1 1 1
## C10-5-09 C10-5-10 C10-5-11 C10-5-12 C10-6-01 C10-6-02 C10-6-03 C10-6-04
## 1 1 1 1 1 1 1 1
## C10-6-05 C10-6-06 C10-6-07 C10-6-08 C10-6-09 C10-6-10 C10-6-11 C10-6-12
## 1 1 1 1 1 1 1 1
## C10-7-01 C10-7-02 C10-7-03 C10-7-04 C10-7-05 C10-7-06 C10-7-07 C10-7-08
## 1 1 1 1 1 1 1 1
## C10-7-09 C10-7-10 C10-7-11 C10-7-12 C10-8-01 C10-8-02 C10-8-03 C10-8-04
## 1 1 1 1 1 1 1 1
## C10-8-05 C10-8-06 C10-8-07 C10-8-08 C10-8-09 C10-8-10 C10-8-11 C10-8-12
## 1 1 1 1 1 1 1 1
## C10-9-01 C10-9-02 C10-9-03 C10-9-04 C10-9-05 C10-9-06 C10-9-07 C10-9-08
## 1 1 1 1 1 1 1 1
## C10-9-09 C10-9-10 C10-9-11 C10-9-12
## 1 1 1 1
```

2.2 Estimate alpha diversity (Shannon diversity index)

```
# Shannon diversity
H <- diversity(counts, index = "shannon", MARGIN = 2, base = exp(1))
tibble(H)
```



```
## # A tibble: 108 x 1
##       H
##   <dbl>
## 1  1.41
## 2  1.56
## 3  1.26
## 4  1.42
## 5  1.20
## 6  1.43
## 7  1.75
## 8  1.86
## 9  1.28
## 10 1.67
## # ... with 98 more rows
```

2.3 Community weighed means (CWMs)

For each sample and genomic trait (16S rRNA gene copy number, generation time, %transcription factors, and generation time), the community weighted mean (CWM) was used for downstream statistical analyses.

```
## 16s rRNA gene copy number
counts.16s <- merge(pic.16s.custom, counts.s.rel.custom, by.x = "sequence", by.y = 0)
row.names(counts.16s) <- counts.16s[, 1]
counts.16s <- counts.16s[, c(2, 4:dim(counts.16s)[2])]
# CWM 16S rRNA gene copy per sample
av.16s <- colSums(counts.16s[, 1] * counts.16s[, 2:dim(counts.16s)[2]])
av.16s[1:10]
```

```
## C10-1-01 C10-1-02 C10-1-03 C10-1-04 C10-1-05 C10-1-06 C10-1-07 C10-1-08
## 3.571474 3.757602 3.548480 3.706464 3.472957 3.653666 3.274543 3.507916
## C10-1-09 C10-1-10
## 3.206120 3.527205
```

```
## Generation time gRodon (from codon usage bias using the gRodon R package)
counts.generationtime.gR <- merge(pic.d.gRodon.default, counts.s.rel.default, by.x = "sequence",
  by.y = 0)
row.names(counts.generationtime.gR) <- counts.generationtime.gR[, 1]
# Select sample columns
counts.generationtime.gR <- counts.generationtime.gR[, c(2, 4:dim(counts.generationtime.gR)[2])]
# CWM generation time
av.dgR <- colSums(counts.generationtime.gR[, 1] * counts.generationtime.gR[, 2:dim(counts.generationtime.gR)[2]])
av.dgR[1:10]
```

```
## C10-1-01 C10-1-02 C10-1-03 C10-1-04 C10-1-05 C10-1-06 C10-1-07 C10-1-08
## 2.920043 2.719025 2.880545 2.737744 2.948559 2.804541 3.311073 3.206684
## C10-1-09 C10-1-10
## 3.334865 3.161886
```

```
# Percent transcription factors (%TF)
counts.TFr <- merge(pic.TFr.default, counts.s.rel.default, by.x = "sequence", by.y = 0) #create a column
row.names(counts.TFr) <- counts.TFr[, 1]
# Select sample columns
```

```
counts.TFr <- counts.TFr[, c(2, 4:dim(counts.TFr)[2])]
# CWM generation time
av.TFr <- colSums(counts.TFr[, 1] * counts.TFr[, 2:dim(counts.TFr)[2]])
av.TFr[1:10]
```

```
## C10-1-01 C10-1-02 C10-1-03 C10-1-04 C10-1-05 C10-1-06 C10-1-07 C10-1-08
## 2.576288 2.643172 2.562466 2.607937 2.501393 2.602768 2.604296 2.641633
## C10-1-09 C10-1-10
## 2.465664 2.571326
```

```
## Genome size (in Mbp)
counts.gs <- merge(pic.gs.default, counts.s.rel.default, by.x = "sequence", by.y = 0)
row.names(counts.gs) <- counts.gs[, 1]
# Select sample columns
counts.gs <- counts.gs[, c(2, 4:dim(counts.gs)[2])]
# CWM Genome size
av.gs <- colSums(counts.gs[, 1] * counts.gs[, 2:dim(counts.gs)[2]])
av.gs[1:10]
```

```
## C10-1-01 C10-1-02 C10-1-03 C10-1-04 C10-1-05 C10-1-06 C10-1-07 C10-1-08
## 4.045784 4.108859 4.045568 4.085424 3.996843 4.069437 3.989177 4.035152
## C10-1-09 C10-1-10
## 3.924678 4.010054
```

```
# NSTI custom
counts.NSTIs <- merge(pic.16s.custom, counts.s.rel.custom, by.x = "sequence", by.y = 0) #create a column
row.names(counts.NSTIs) <- counts.NSTIs[, 1]
counts.NSTIs <- counts.NSTIs[, c(3, 4:dim(counts.NSTIs)[2])] #select releavnt samples
av.NSTI <- colSums(counts.NSTIs[, 1] * counts.NSTIs[, 2:dim(counts.NSTIs)[2]]) #average number Of 16s
summary(av.NSTI)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.02578 0.06757 0.10371 0.09982 0.12602 0.17881
```

```
# NSTI default
counts.NSTIs <- merge(pic.gs.default, counts.s.rel.default, by.x = "sequence", by.y = 0) #create a column
row.names(counts.NSTIs) <- counts.NSTIs[, 1]
counts.NSTIs <- counts.NSTIs[, c(3, 4:dim(counts.NSTIs)[2])] #select releavnt samples
av.NSTI <- colSums(counts.NSTIs[, 1] * counts.NSTIs[, 2:dim(counts.NSTIs)[2]]) #average number Of
summary(av.NSTI)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01122 0.02949 0.05102 0.05674 0.08241 0.12767
```

3 Community trait distribution during the experiment

3.1 Dataframe and format trait CWM values

```

# Data frame with CWM trait data and sample schema
traits <- cbind(schema, av.16s, av.gs, av.dgR, av.TFr, H)

# Formatting data set from wide to long format
traits.w <- melt(traits[, 2:10], id.vars = c("Sal", "DOM", "T"), measure.vars = c("av.16s",
  "av.dgR", "av.TFr", "av.gs", "H"))

# Add column with Replicate ID
traits.w$Rep = rep(c("1", "2", "3"), each = 2)
traits.w$Rep = as.factor(traits.w$Rep)

# Add Column with sample time (day)
traits.w$Time = as.numeric(rep(c(4, 8, 15, 18, 22, 29, 36, 39, 41), each = 12))

```

3.2 Summarizing data replicate mean values

```
tibble(aggregate(value ~ Sal + DOM + variable, traits.w, mean))
```

```

## # A tibble: 20 x 4
##   Sal   DOM variable value
##   <fct> <fct> <fct>    <dbl>
## 1 C     H     av.16s    2.64
## 2 D     H     av.16s    2.84
## 3 C     L     av.16s    2.86
## 4 D     L     av.16s    2.93
## 5 C     H     av.dgR    4.28
## 6 D     H     av.dgR    4.03
## 7 C     L     av.dgR    4.72
## 8 D     L     av.dgR    4.39
## 9 C     H     av.TFr    2.72
## 10 D    H     av.TFr    2.79
## 11 C    L     av.TFr    2.76
## 12 D    L     av.TFr    2.78
## 13 C    H     av.gs     3.93
## 14 D    H     av.gs     3.98
## 15 C    L     av.gs     4.02
## 16 D    L     av.gs     3.99
## 17 C    H     H         2.39
## 18 D    H     H         2.67
## 19 C    L     H         1.81
## 20 D    L     H         1.90

```

3.3 Test for normality and homogeneity of variances

```

# Normality Kolmogorov smirnov test
l = length(levels(traits.w$variable))
traits.w$T = factor(traits.w$T)
sum.normality = data.frame(variable = rep(NA, l), L_C = rep(NA, l), L_D = rep(NA,

```

```

1), H_C = rep(NA, 1), H_D = rep(NA, 1))

for (i in 1:length(levels(traits.w$variable))) {
  tmp = traits.w[traits.w$variable == levels(traits.w$variable)[i], ]
  sum.normality$variable[i] = levels(traits.w$variable)[i]
  sum.normality$L_C[i] = ols_test_normality((tmp$value[tmp$DOM == "L" & tmp$Sal ==
    "C"]))[[1]][[2]]
  sum.normality$L_D[i] = ols_test_normality((tmp$value[tmp$DOM == "L" & tmp$Sal ==
    "D"]))[[1]][[2]]
  sum.normality$H_C[i] = ols_test_normality((tmp$value[tmp$DOM == "H" & tmp$Sal ==
    "C"]))[[1]][[2]]
  sum.normality$H_D[i] = ols_test_normality((tmp$value[tmp$DOM == "H" & tmp$Sal ==
    "D"]))[[1]][[2]]
}
sum.normality[, 2:5] = round(sum.normality[, 2:5], 3)
tibble(sum.normality)

```

```

## # A tibble: 5 x 5
##   variable  L_C  L_D  H_C  H_D
##   <chr>    <dbl> <dbl> <dbl> <dbl>
## 1 av.16s  0.682 0.757 0.127 0.753
## 2 av.dgR  0.503 0.385 0.871 0.897
## 3 av.TFr  0.79  0.724 0.512 0.987
## 4 av.gs   0.565 0.536 0.415 0.762
## 5 H       0.504 0.342 0.252 0.115

```

```

# Homogeneity of variances
HV = traits.w %>%
  group_by(variable, DOM, Sal) %>%
  levene_test(value ~ T)
tibble(HV)

```

```

## # A tibble: 20 x 7
##   Sal  DOM  variable  df1  df2 statistic    p
##   <fct> <fct> <fct>    <int> <int>    <dbl> <dbl>
## 1 C    H    av.16s      8    18     0.695 0.691
## 2 D    H    av.16s      8    18     0.606 0.761
## 3 C    L    av.16s      8    18     0.433 0.886
## 4 D    L    av.16s      8    18     0.994 0.473
## 5 C    H    av.dgR      8    18     0.493 0.846
## 6 D    H    av.dgR      8    18     1.00  0.467
## 7 C    L    av.dgR      8    18     0.541 0.811
## 8 D    L    av.dgR      8    18     0.594 0.770
## 9 C    H    av.TFr      8    18     0.871 0.558
## 10 D   H    av.TFr      8    18     0.741 0.656
## 11 C   L    av.TFr      8    18     0.944 0.507
## 12 D   L    av.TFr      8    18     1.31  0.299
## 13 C   H    av.gs       8    18     1.49  0.229
## 14 D   H    av.gs       8    18     0.727 0.667
## 15 C   L    av.gs       8    18     0.515 0.830
## 16 D   L    av.gs       8    18     1.05  0.440
## 17 C   H    H           8    18     1.10  0.409
## 18 D   H    H           8    18     0.373 0.921

```

```
## 19 C      L      H           8    18    0.842 0.579
## 20 D      L      H           8    18    0.500 0.840
```

3.4 Repeated measurement ANOVA

A repeated measurement anova was applied separately for the two DOM regimes to test the effect of the disturbance regime on the distribution of the resilience- and resistance-related genomic traits.

```
# Repeated measurements ANOVA for LDOM

list.rm_anova = list()
m.rm_anova = data.frame(variable = rep(NA, 1), F_Time = rep(NA, 1), P_Time = rep(NA,
  1), F_Sal = rep(NA, 1), P_Sal = rep(NA, 1))
for (i in 1:length(levels(traits.w$variable))) {
  list.rm_anova[[i]] <- with(traits.w[traits.w$DOM == "L" & traits.w$variable ==
    levels(traits.w$variable)[i], ], aov(value ~ T * Sal + Error(Rep)))
  m.rm_anova$variable[i] = levels(traits.w$variable)[i]
  m.rm_anova$F_Time[i] = unlist(summary(list.rm_anova[[i]]))["Error: Within.F value1"]
  m.rm_anova$P_Time[i] = unlist(summary(list.rm_anova[[i]]))["Error: Within.Pr(>F)1"]
  m.rm_anova$F_Sal[i] = unlist(summary(list.rm_anova[[i]]))["Error: Within.F value2"]
  m.rm_anova$P_Sal[i] = unlist(summary(list.rm_anova[[i]]))["Error: Within.Pr(>F)2"]
}

m.rm_anova[, 2:5] = round(m.rm_anova[, 2:5], 3)
a.rm_anova <- m.rm_anova
tibble(a.rm_anova)
```

```
## # A tibble: 5 x 5
##   variable F_Time P_Time F_Sal P_Sal
##   <chr>     <dbl> <dbl> <dbl> <dbl>
## 1 av.16s    3.57   0.004 0.268 0.608
## 2 av.dgR   24.5    0     6.19 0.018
## 3 av.TFr    5.25    0     0.115 0.737
## 4 av.gs     0.342   0.943 0.167 0.685
## 5 H         8.45    0     0.699 0.409
```

```
# Repeated measurements ANOVA for HDOM

list.rm_anova = list()
m.rm_anova = data.frame(variable = rep(NA, 1), F_Time = rep(NA, 1), P_Time = rep(NA,
  1), F_Sal = rep(NA, 1), P_Sal = rep(NA, 1))
for (i in 1:length(levels(traits.w$variable))) {
  list.rm_anova[[i]] <- with(traits.w[traits.w$DOM == "H" & traits.w$variable ==
    levels(traits.w$variable)[i], ], aov(value ~ T * Sal + Error(Rep)))
  m.rm_anova$variable[i] = levels(traits.w$variable)[i]
  m.rm_anova$F_Time[i] = unlist(summary(list.rm_anova[[i]]))["Error: Within.F value1"]
  m.rm_anova$P_Time[i] = unlist(summary(list.rm_anova[[i]]))["Error: Within.Pr(>F)1"]
  m.rm_anova$F_Sal[i] = unlist(summary(list.rm_anova[[i]]))["Error: Within.F value2"]
  m.rm_anova$P_Sal[i] = unlist(summary(list.rm_anova[[i]]))["Error: Within.Pr(>F)2"]
}

m.rm_anova[, 2:5] = round(m.rm_anova[, 2:5], 3)
a.rm_anova <- m.rm_anova
tibble(a.rm_anova)
```

```
## # A tibble: 5 x 5
##   variable F_Time P_Time F_Sal P_Sal
##   <chr>      <dbl> <dbl> <dbl> <dbl>
## 1 av.16s      7.26  0      5.98 0.02
## 2 av.dgR      8.99  0      5.19 0.029
## 3 av.TFr      4.73 0.001   4.02 0.053
## 4 av.gs       3.37 0.006   3.80 0.06
## 5 H          16.1  0     14.6 0.001
```

4 Paired-test per Genomic trait

```
traits.w.mean = aggregate(value ~ Sal + DOM + T + variable, data = traits.w, mean)
res.ttest = list()
res.ttest.df = data.frame(variable = levels(traits.w.mean$variable), direction = c("greater",
"less", "greater", "greater", "greater"), LDOM.pvalue = NA, HDOM.pvalue = NA)

for (i in 1:length(levels(traits.w.mean$variable))) {
  tmp = traits.w.mean[traits.w.mean$variable == levels(traits.w$variable)[i], ]
  value.control = tmp[tmp$DOM == "L" & tmp$Sal == "C", ]
  value.disturbance = tmp[tmp$DOM == "L" & tmp$Sal == "D", ]
  res.ttest[[i]] = t.test(value.disturbance$value, value.control$value, alternative = res.ttest.df$direction[i],
var.equal = T, paired = T)
  res.ttest.df$LDOM.pvalue[i] = res.ttest[[i]]$p.value
}

res.ttest = list()
for (i in 1:length(levels(traits.w.mean$variable))) {
  tmp = traits.w.mean[traits.w.mean$variable == levels(traits.w$variable)[i], ]
  value.control = tmp[tmp$DOM == "H" & tmp$Sal == "C", ]
  value.disturbance = tmp[tmp$DOM == "H" & tmp$Sal == "D", ]
  res.ttest[[i]] = t.test(value.disturbance$value, value.control$value, alternative = res.ttest.df$direction[i],
var.equal = T, paired = T)
  res.ttest.df$HDOM.pvalue[i] = res.ttest[[i]]$p.value
}

tibble(res.ttest.df)
```

```
## # A tibble: 5 x 4
##   variable direction LDOM.pvalue HDOM.pvalue
##   <chr>      <chr>      <dbl>      <dbl>
## 1 av.16s    greater      0.138      0.00739
## 2 av.dgR    less        0.0154     0.00334
## 3 av.TFr    greater      0.334      0.00112
## 4 av.gs     greater      0.753      0.00781
## 5 H        greater      0.160      0.000358
```

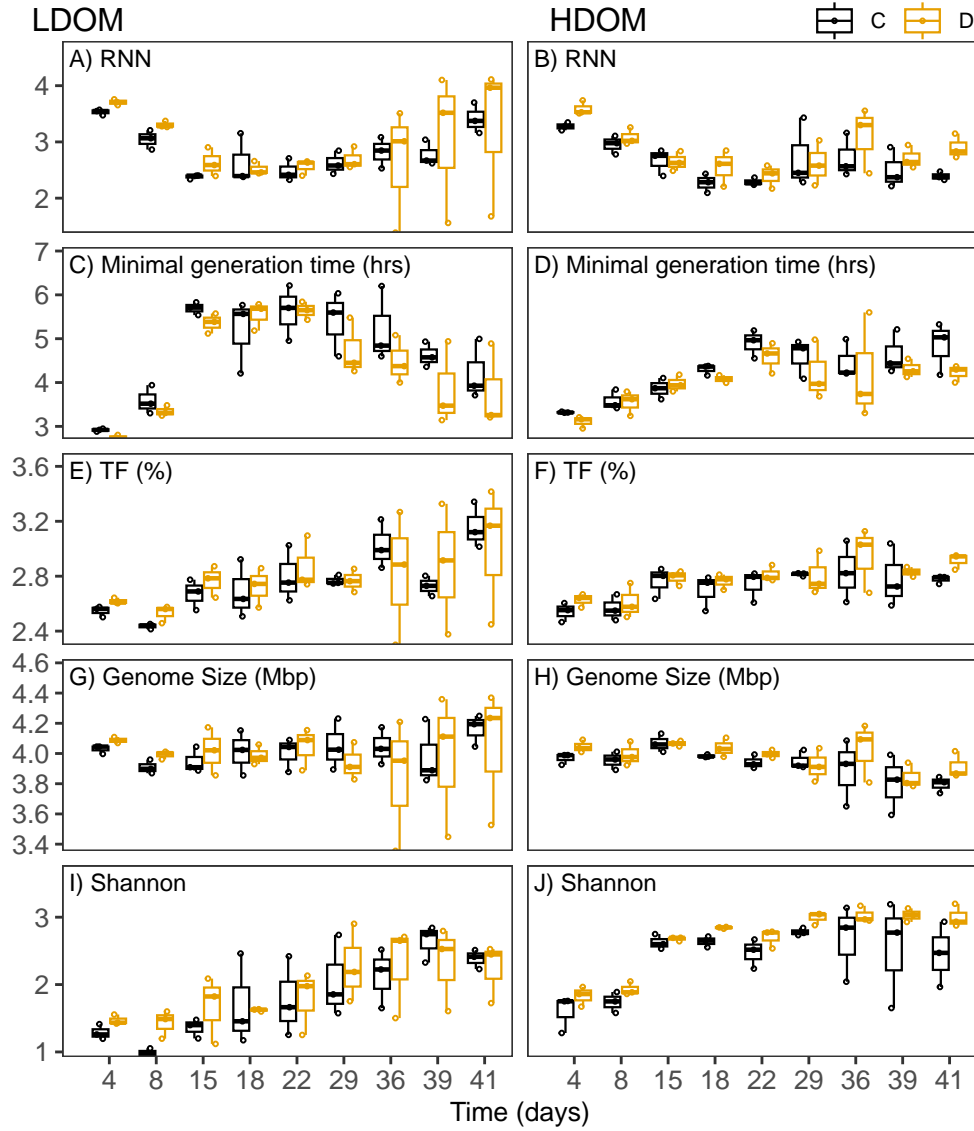
4.1 Manuscript Figure 3

```
# New facet label names for dose variable
bxp_labs <- c("", "", "", " ", "")
names(bxp_labs) <- levels(traits.w$variable)
traits.w$DOM = factor(traits.w$DOM, levels = c("L", "H"))

levels(traits.w$T) = c("4", "8", "15", "18", "22", "29", "36", "39", "41")

bxp = traits.w %>%
  ggplot(aes(x = T, y = value, colour = Sal)) + geom_boxplot(aes(colour = (Sal)),
    outlier.shape = NA, alpha = 0.3, size = 0.4) + geom_jitter(aes(colour = Sal),
    shape = 21, size = 0.5, position = position_jitterdodge()) + scale_colour_manual(values = cbbPalett
    name = "") + theme_bw() + ylab("") + scale_y_continuous(expand = expansion(mult = c(0,
    0.25))) + theme(panel.grid.minor = element_blank(), panel.grid.major = element_blank(),
    axis.text.x = element_text(size = 10), axis.text.y = element_text(size = 10)) +
    theme(legend.position = c(0.9, 1.02), legend.direction = "horizontal", legend.key = element_blank()
    legend.background = element_blank()) + theme(text = element_text(size = 10,
    family = "ArialMT")) + facet_grid(variable ~ DOM, scale = "free_y", switch = "y",
    labeller = labeller(variable = bxp_labs)) + xlab("Time (days)") + theme(strip.placement.y = "outsid
    strip.text.y = element_text(angle = 270), strip.background = element_blank()) +
    labs(tag = "LDOM HDOM") + theme(plot.tag.position = c
    1.02))

# Labels using facet_tag
bxp = tag_facet(bxp, open = "", close = "", tag_pool = c(" A) RNN", " B) RNN ", " C) Minimal generation
  " D) Minimal generation time (hrs)", " E) TF (%)", " F) TF (%)", " G) Genome Size (Mbp)",
  " H) Genome Size (Mbp)", " I) Shannon", " J) Shannon"), x = 0, fontface = 1,
  size = 3, hjust = 0)
bxp = bxp + theme(plot.margin = margin(t = 20, r = 5, b = 5, l = 5, unit = "pt"))
bxp
```



Boxplots displaying CWMs of genomic traits. LDOM and HDOM in the left and right panels restively for A, B), RRN, C, D) generation time (d), E, F), %TF G, H) genome size and I, J) Shannon diversity index.