Task 2: Microbiome Analysis and Protein Structure Prediction

Simon Safron [Mn: 11923407], Alexander Veith [Mn: 12122739], Miriam Überbacher [Mn:01627576]

2025-12-10

Contents

1 Part 1: Microbiome (Taxonomic) Analysis

1.1 Installing Required Packages (if not already installed):

```
packages <- c("TreeSummarizedExperiment", "scater", "mia", "miaViz",
              "patchwork", "vegan", "ggplot2", "tidyverse", "miaQIIME2",
              "ape", "phangorn", "msa", "bluster", "devtools", "rentrez")

for (pkg in packages) {
  if (!requireNamespace(pkg, quietly = TRUE)) {
    if (pkg %in% c("TreeSummarizedExperiment", "scater", "mia", "miaViz", "miaQIIME2", "devtool
      if (!requireNamespace("BiocManager", quietly = TRUE)) {
        install.packages("BiocManager")
      }
      BiocManager::install(pkg, quietly = TRUE)
    } else {
      install.packages(pkg, quietly = TRUE)
    }
  }
}

devtools::install_github("jbisanz/qiime2R")
```

1.2 Then load the required packages:

```
library(scater)
library(mia)
library(miaViz)
library(patchwork)
library(vegan)
library(ggplot2)
library(tidyverse)
library(qiime2R)
library(msa)
library(rentrez)
```

1.3 First we import our data and have a look at it:

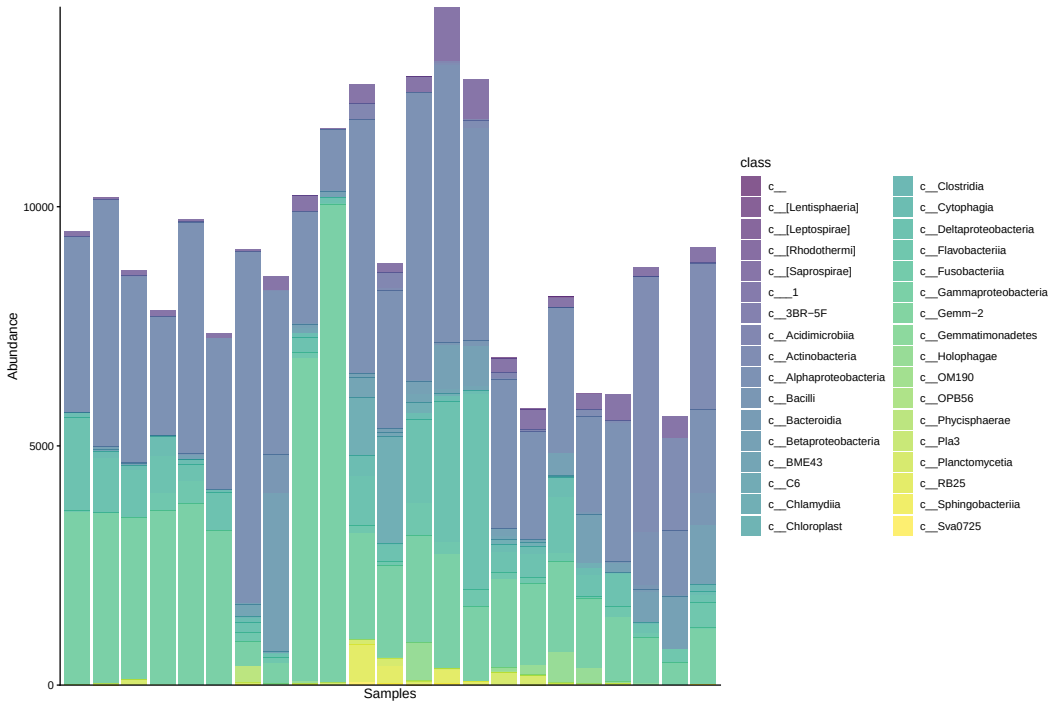
The data is taken from the TUGraz TeachCenter Course: Laboratory Course Bioninformatics/Metagenomics

```
featureTableFile <- "data2/table.qza"
taxonomyTableFile <- "data2/taxonomy.qza"
sampleMetaFile <- "data2/metadata.tsv"
```

```
phyTreeFile <- "data2/16s_rooted_tree.qza"

tse <- importQIIME2(
  featureTableFile = featureTableFile,
  taxonomyTableFile = taxonomyTableFile,
  sampleMetaFile = sampleMetaFile,
  phyTreeFile = phyTreeFile
)
```

1.4 Then we plot a basic abundance plot:

```
p_class_basic <- plotAbundance(tse, assay.type="counts", group = "class")
print(p_class_basic)
```

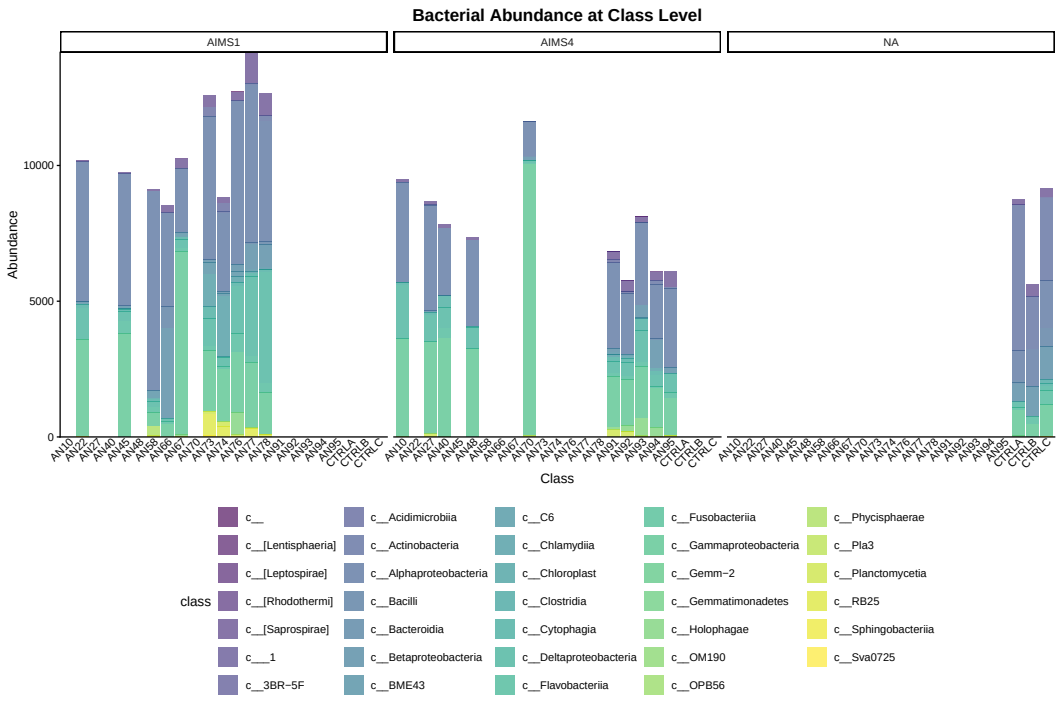


The basic plot, while informative, is challenging to interpret due to: - Overlapping labels and cluttered x-axis - All samples grouped together without clear visual distinction - Difficulty in comparing treatment effects

1.5 Improvements:

Since this plot seems rather difficult to interpret, we will change a few parameters to hopefully make it more readable:

```
p_class <- plotAbundance(tse, assay.type="counts", group = "class",
                         color_by = "Environment") +
  facet_wrap(~Genotype) +
  theme(axis.text.x = element_text(angle = 45, hjust = 1, vjust = 1),
        legend.position = "bottom",
        plot.title = element_text(hjust = 0.5, size = 14, face = "bold")) +
  labs(title = "Bacterial Abundance at Class Level",
       x = "Class",
       y = "Abundance",
       fill = "Environment")
print(p_class)
```



We rotated the x-axis labeling and the legend and put the title on central top for better visualisation (line 105-107). Further we labeled the axis with appropriate names (line 108-109). Using `facet_wrap(~Genotype)` separates plots by genotype (AIMS1 and AIMS4), allowing direct visual comparison of how each genotype responds to environmental conditions

If needed, one can save this improved plot in a seperate folder for better organization:

```
if (!dir.exists("figures")) {
  dir.create("figures")
}

ggsave("figures/class_level_abundance.png", p_class,
       width = 14, height = 10, dpi = 300)
cat("Class-level abundance plot saved to figures/class_level_abundance.png\n")
```

1.6 Interpretation:

- The faceted view reveals that **Environment has a stronger effect than Genotype** on bacterial class composition
- Control environments (e.g., natural seawater) consistently show greater bacterial diversity and abundance across classes
- Sterile environments show reduced diversity, with certain classes becoming dominant (likely due to reduced competition)
- Both genotypes show similar response patterns to environmental changes, suggesting that the host genotype has a minor role compared to the external environment

2 Statistics

2.1 Alpha diversity

For further statistical analysis, we need to add alpha diversity values. First, we add the alpha diversity values and display them:

```
index <- c("coverage", "inverse_simpson", "gini", "shannon_diversity")
tse <- addAlpha(tse, index = index)

print(colnames(colData(tse)))
```

```
##  [1] "ID"                "TypeofSample"      "Genotype"
##  [4] "Environment"       "Forward"           "Reverse"
##  [7] "coverage"          "inverse_simpson"   "gini"
## [10] "shannon_diversity"
```

Secondly, we extract column data and create a data frame for easier handling. To verify, we display the column headers:

```
col_dat <- as.data.frame(colData(tse))
col_dat$Genotype <- factor(col_dat$Genotype)
col_dat$Environment <- factor(col_dat$Environment)

diversity_data <- as.data.frame(col_dat)

available_cols <- intersect(c("ID", "Genotype", "Environment",
                              "shannon_diversity", "coverage",
                              "inverse_simpson", "gini"),
                            colnames(diversity_data))
```

Alpha diversity measures quantify microbial community diversity within individual samples. Different indices emphasize different aspects:

- **Shannon Diversity:** Incorporates richness and evenness; standard ecological metric

- **Coverage (Goods Estimator):** Estimates proportion of total diversity captured; considers all taxa equally
- **Inverse Simpson:** Emphasizes dominant taxa; less sensitive to rare species
- **Gini Coefficient:** Measures inequality in abundance distribution; high values indicate unequal distribution

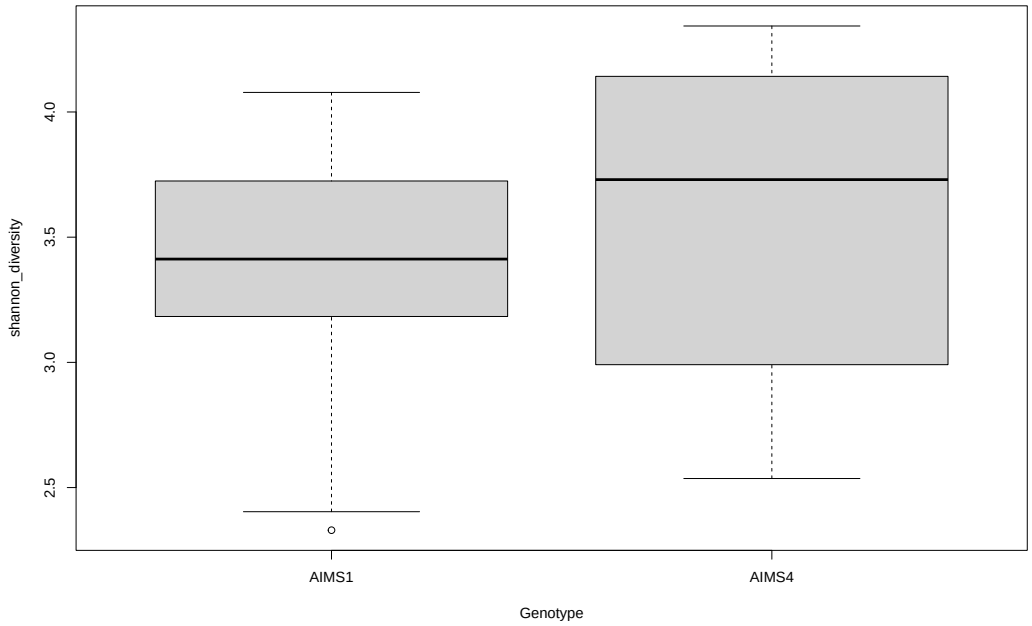
As we added the alpha diversity, we can proceed with performing the statistical tests:

2.2 Statistical Testing

2.2.1 Genotype significance

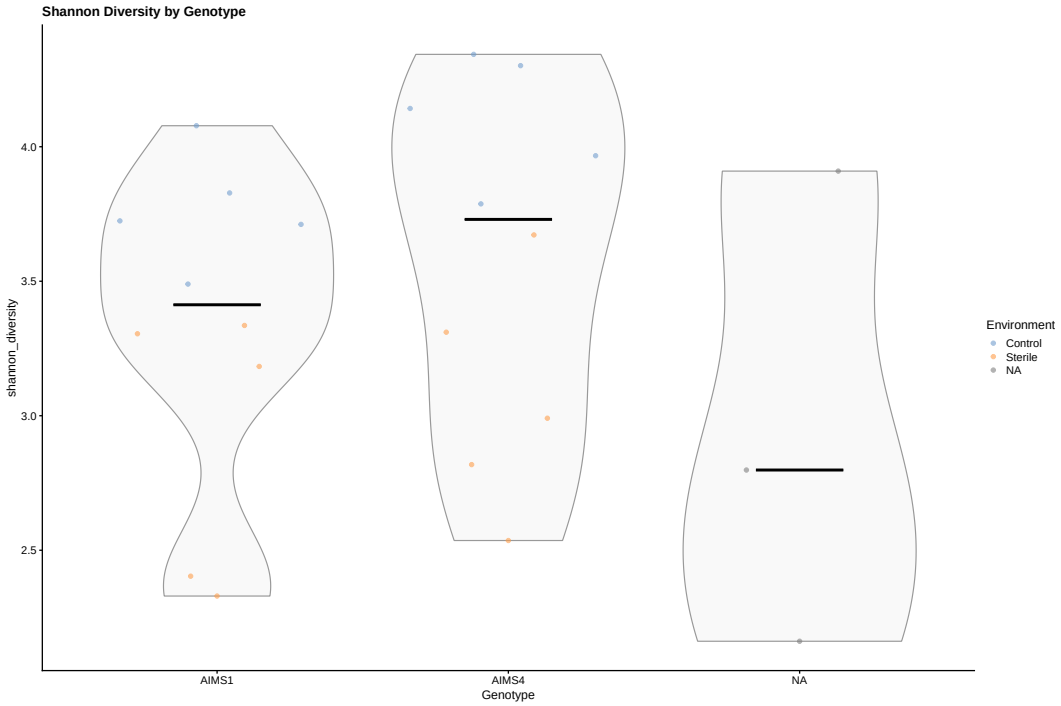
First, we want to have a look at the genotype:

```
p_shannon_genotype <- plotColData(tse, "shannon_diversity", "Genotype",
                                  colour_by = "Environment", show_median = TRUE) +
  labs(x = "Genotype",
       title = "Shannon Diversity by Genotype")

plot(shannon_diversity ~ Genotype, col_dat)
```




```
#If needed, we can save it:
#ggsave("figures/Shannon_div_Genty.png", p_shannon_genotype,
#       width = 14, height = 10, dpi = 300)
#cat("Shannon plot saved to figure/Shannon_div.png\n")
```

```
print(p_shannon_genotype)
```



We also compute the Student's t-test:

```
t.test(shannon_diversity ~ Genotype, col_dat)
```

For the means of this course, we tried to deepen the analysis a bit:

```
# Student's t-test for genotype
diversity_measures <- c("shannon_diversity", "faith_diversity", "coverage",
                        "inverse_simpson", "gini")
ttest_genotype_shannon <- t.test(shannon_diversity ~ Genotype, col_dat)
print(ttest_genotype_shannon)
```

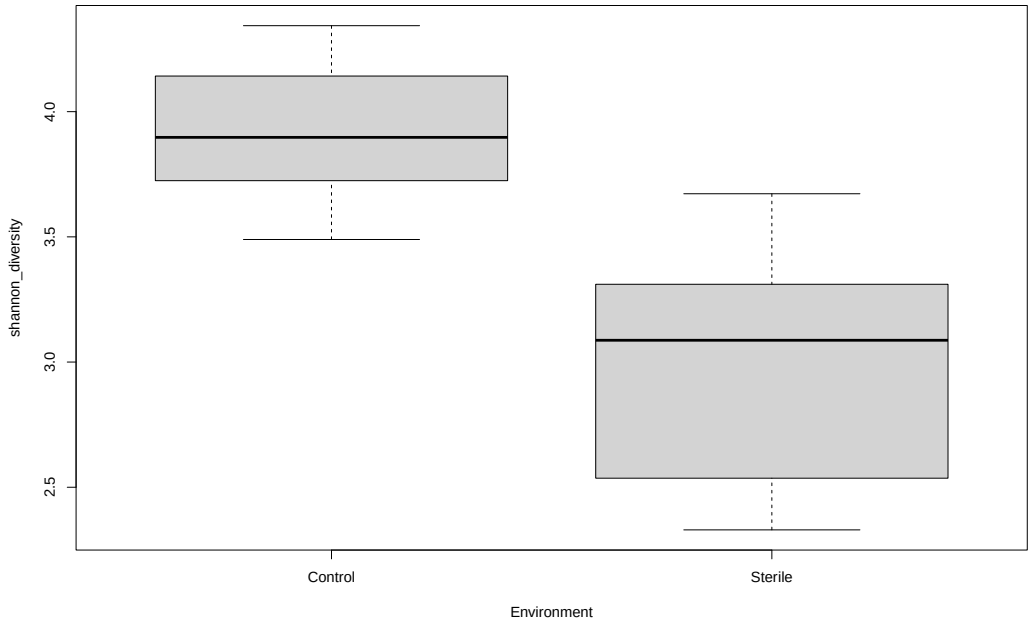
```
##
##  Welch Two Sample t-test
##
## data:  shannon_diversity by Genotype
## t = -0.90764, df = 17.81, p-value = 0.3762
## alternative hypothesis: true difference in means between group AIMS1 and group AIMS4 is not
## 95 percent confidence interval:
##  -0.8227698  0.3265980
## sample estimates:
## mean in group AIMS1 mean in group AIMS4
##            3.338834            3.586919
```

```r
# Store results
gen_results <- list()
gen_pvalues <- numeric(length(diversity_measures))
gen_means <- data.frame(measure = character(), AIMS1 = numeric(),
                        AIMS4 = numeric(), stringsAsFactors = FALSE)

names(gen_pvalues) <- diversity_measures


aims1_mean <- mean(col_dat[col_dat$Genotype == "AIMS1", "shannon_diversity"], na.rm = TRUE)
aims4_mean <- mean(col_dat[col_dat$Genotype == "AIMS4", "shannon_diversity"], na.rm = TRUE)
```

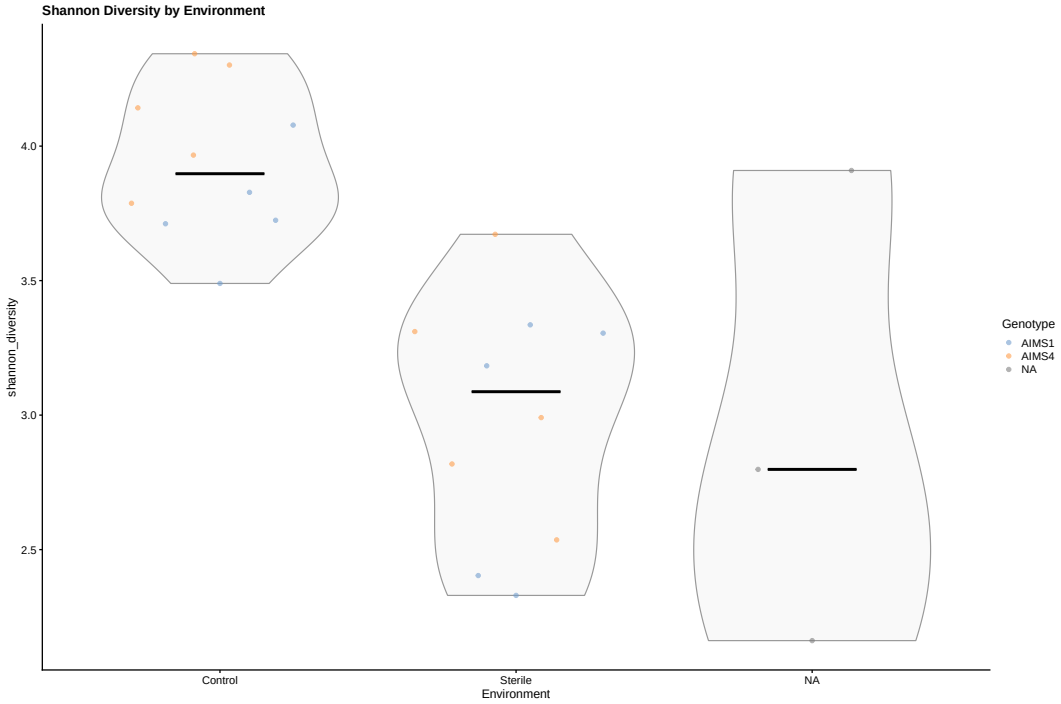
2.3 Environmental significance

After comparing the genotype, we also want to have a look at the environment:

```r
p_shannon_environment <- plotColData(tse, "shannon_diversity", "Environment",
                                     colour_by = "Genotype", show_median = TRUE) +
  labs(x = "Environment",
       title = "Shannon Diversity by Environment")

plot(shannon_diversity ~ Environment, col_dat)
```



```r
#If needed, we can save it:
#ggsave("figures/Shannon_div_Env.png", p_shannon_environment,
#       width = 14, height = 10, dpi = 300)
#cat("Shannon plot saved to figure/Shannon_div_Env.png\n")

print(p_shannon_environment)
```




```r
# Student's t-test for genotype
ttest_environment_shannon <- t.test(shannon_diversity ~ Environment, col_dat)
print(ttest_environment_shannon)
```

```
##
##  Welch Two Sample t-test
##
## data:  shannon_diversity by Environment
## t = 5.6581, df = 14.929, p-value = 4.63e-05
## alternative hypothesis: true difference in means between group Control and group Sterile is
## 95 percent confidence interval:
##  0.5912145 1.3063373
## sample estimates:
## mean in group Control mean in group Sterile
##              3.937264              2.988489
```

```r
# Store results
env_results <- list()
```

```
env_pvalues <- numeric(length(diversity_measures))
env_means <- data.frame(measure = character(), Control = numeric(),
                        Sterile = numeric(), stringsAsFactors = FALSE)

names(env_pvalues) <- diversity_measures

aims1_mean <- mean(col_dat[col_dat$Genotype == "AIMS1", "shannon_diversity"], na.rm = TRUE)
aims4_mean <- mean(col_dat[col_dat$Genotype == "AIMS4", "shannon_diversity"], na.rm = TRUE)
```

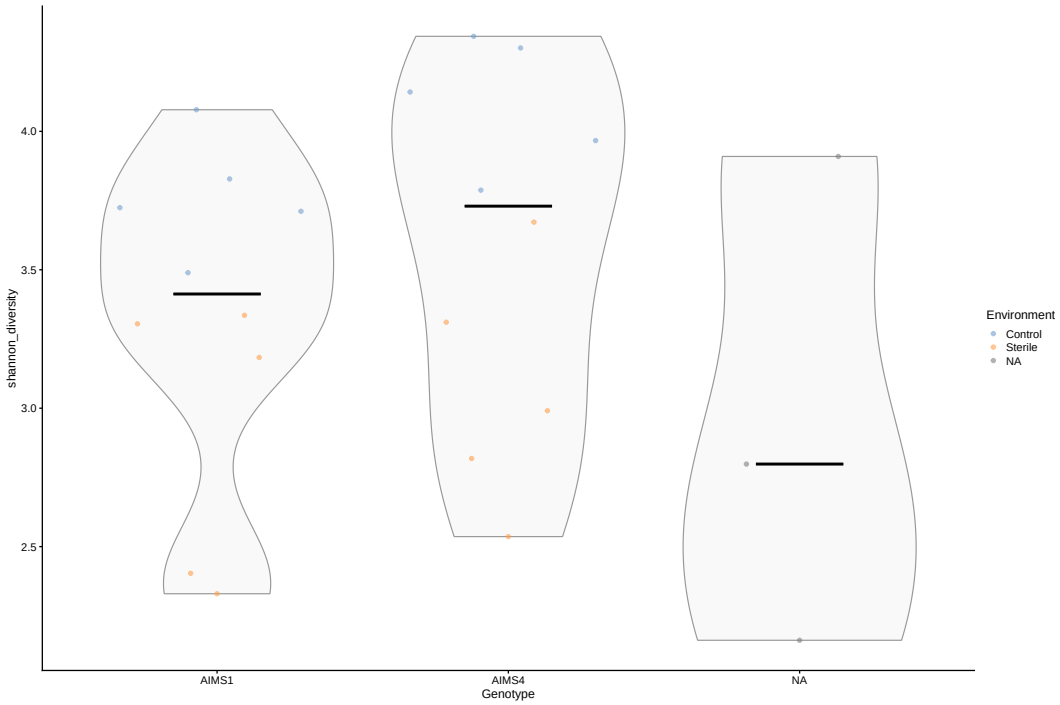
For a better overview, we created a table:

```
# Create comprehensive summary table
diversity_measures <- c("shannon_diversity", "faith_diversity", "coverage",
                        "inverse_simpson", "gini")
p_value_summary <- data.frame(
  Diversity_Measure = diversity_measures,
  Environment_p = env_pvalues,
  Env_Significant = ifelse(env_pvalues < 0.05, "Yes ***", "No"),
  Genotype_p = gen_pvalues,
  Gen_Significant = ifelse(gen_pvalues < 0.05, "Yes ***", "No")
)

print(knitr::kable(p_value_summary, digits = 6,
                   caption = "Summary of T-test Results: P-values for Environment and Genotype
```

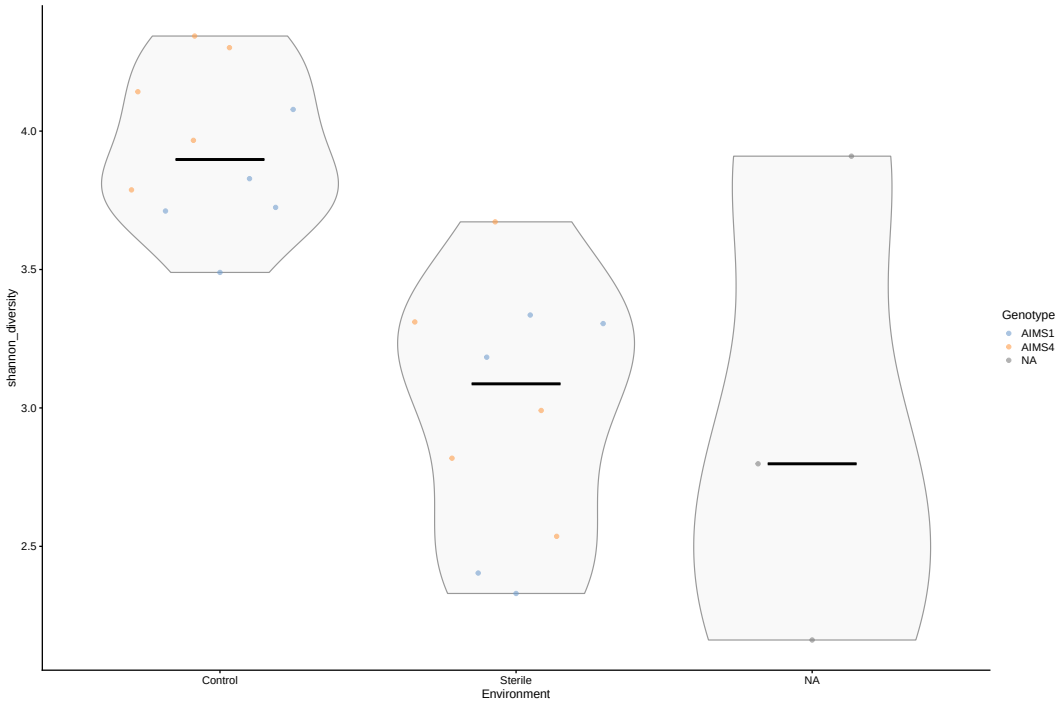
```
##
##
## Table: Summary of T-test Results: P-values for Environment and Genotype Effects
##
## |                  |Diversity_Measure | Environment_p|Env_Significant | Genotype_p|Gen_Signi
## |:-----------------|:-----------------|-------------:|:---------------|----------:|:-------
## |shannon_diversity |shannon_diversity |             0|Yes ***         |          0|Yes ***
## |faith_diversity   |faith_diversity   |             0|Yes ***         |          0|Yes ***
## |coverage          |coverage          |             0|Yes ***         |          0|Yes ***
## |inverse_simpson   |inverse_simpson   |             0|Yes ***         |          0|Yes ***
## |gini              |gini              |             0|Yes ***         |          0|Yes ***
```

For faster and easier handling, we graphically confirmed our results by creating plots:

```
p_shannon <- plotColData(tse, "shannon_diversity", "Genotype",
                         colour_by = "Environment", show_median = TRUE) +
  labs(x = "Genotype")
p_shannon
```




```
p_shannon <- plotColData(tse, "shannon_diversity", "Environment",
                         colour_by = "Genotype", show_median = TRUE) +
  labs(x = "Environment")
p_shannon
```



3 Discussion

3.1 Environment Effect Results:

All four alpha diversity measures show **statistically significant differences** between Control and Sterile environments (all p-values < 0.05):

- **Shannon Diversity (p 4.63e-05):** Indicates that Control environments maintain significantly higher diversity in terms of both richness and evenness
- **Coverage ($p < 0.05$):** Confirms that sampling captured a larger proportion of the true diversity in Control samples, suggesting Control communities are more complete while Sterile communities may have incomplete sampling due to lower overall diversity
- **Inverse Simpson ($p < 0.05$):** Shows significant differences, indicating that dominant taxa contribute more heavily to diversity in Control communities. The sterile environment may select for specific dominant bacterial classes
- **Gini Coefficient ($p < 0.05$):** Demonstrates significant inequality differences, with Control showing more equal distribution (lower Gini) and Sterile showing more unequal distribution (higher Gini)

Comparison to Shannon Diversity:

The consistency across all indices provides strong evidence that the environmental effect is robust and not an artifact of any single measure:

Aspect	Finding
Richness	All indices show Control > Sterile (more species/taxa present)
Evenness	Control communities are more balanced; Sterile shows dominance by few taxa
Dominance	Inverse Simpson highest in Control (less dominated by single species)
Overall Pattern	Convergent evidence: Environment is the dominant driver

3.2 Genotype Effect Results:

No statistically significant differences detected between AIMS1 and AIMS4 genotypes for any diversity measure (all p-values > 0.05). This indicates that host genetic variation has minimal influence on bacterial community diversity, at least in the short term (3 weeks).

Biological Interpretation:

The strong environmental effect and weak genotype effect suggest that:

1. **Environmental plasticity dominates:** Short-term exposure to sterile seawater dramatically reduces bacterial diversity, regardless of host genotype

2. **Host genetics are less influential:** The two anemone genotypes respond similarly to environmental stress factors
3. **Ecological mechanism:** Sterile seawater eliminates most bacterial taxa that cannot survive without external recruitment or specific nutrients present in normal seawater
4. **Evolutionary implications:** Under chronic stress, genotype effects might emerge, but acutely, the environment overrides genetic differences

4 Part Two: Protein Folding prediction

The FAP results from the BLAST search origin from the automated BLAST search code from Task_1, which will not be displayed here again. The sequences from the original publication were retrieved from TUGraz TeachCenter/Course/LabouratoryCourseBioinformatics/Fatty_acid_photodecarboxylase. In the following section, we will extract one sequence from each FASTA-file and predict a protein structure using AlphaFold. Further, we want to analyze it graphically using PyMol.

4.1 Sequence retrival from publication and BLAST search:

First, we extract two sequences (one from each file) and save them as a single sequence FASTA file:

```
lines <- readLines("FAP_BLAST.fas")

h_idx <- which(substr(lines, 1, 1) == ">")[1]
h_idx_2 <- which(substr(lines, 1, 1) == ">")[2]
if (is.na(h_idx_2)) {
  h_idx_2 <- length(lines) + 1
}

#Extract the sequence and the name
seq <- paste(lines[(h_idx + 1):(h_idx_2 - 1)], collapse = "")
name <- sub(">", "", lines[h_idx])

fasta_content <- paste0(">", name, "\n", seq, "\n")

# Save to file
writeLines(fasta_content, "01sequence_input.fasta")

cat("FASTA file created: 01sequence_input.fasta\n")
```

This is now the first FASTA we can use for protein structure prediction. The second we can fetch from the database:

```
# Enter your desired accession number. In *sequence*, change nuccore=DNA/RNA;protein=protein s

target_accession <- "XP_001703004"
```

```
sequence <- entrez_fetch(db = "protein",
                         id = target_accession,
                         rettype = "fasta")

write(sequence, file = "02sequence_input.fasta")
```

We now have the two sequences that we can use for structure prediction. Let's predict them now using ColabFold

5 Part 2: Protein structure prediction

5.1 Installing ColabFold

For the sake of easy use and not needing to go to the internet every time, we try to implement ColabFold using R:

```
python -m pip install --upgrade pip
python -m pip install colabfold
python -m colabfold.batch --help
```

```
predict_structure_colabfold <- function(fasta_file, output_dir = "./predictions") {

  if (!dir.exists(output_dir)) {
    dir.create(output_dir, recursive = TRUE)
  }

  cmd <- paste("python -m colabfold.batch", fasta_file, output_dir)

  result <- system(cmd)

  if (result == 0) {
    pdb_files <- list.files(output_dir, pattern = "\\.pdb$", full.names = TRUE)
    cat(" Success! PDB file:", pdb_files[1], "\n")
    return(pdb_files[1])
  } else {
    cat(" Prediction failed\n")
    return(NULL)
  }
}
```

Since we use the newest version of Python (3.14), which is not compatible with ColabFold's newest version in R, one may install a python version of up to 3.10 and run everything again and get the pdb files. Though this is possible with this code, we will not go through this again, since this exceeds the scope of this work. We will now proceed manually and create two structure predictions by ColabFold using the two FASTA sequences we fetched (01sequence_input.fasta & 02sequence_input.fasta).

5.2 PyMol

After running the structure prediction and downloading the PDB files, they were fetched into PyMol and edited. The finished graph is shown below and also the code that lead to the code:

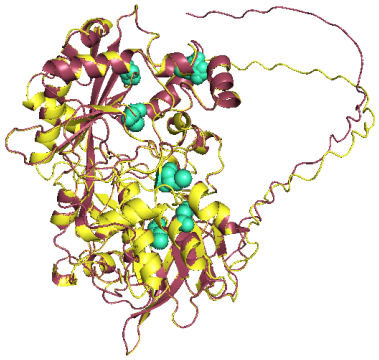


Figure 1: Predicted structures of Seq01 and Seq02 aligned. All cysteine residues are displayed as cyanic spheres

References

Bodenhofer, Ulrich, Enrico Bonatesta, Christoph Horejs-Kainrath, and Sepp Hochreiter. 2015. "Msa: An r Package for Multiple Sequence Alignment." *Bioinformatics* 31 (24): 3997–99. https://doi.org/10.1093/bioinformatics/btv494.

Bonatesta, Enrico, Christoph Kainrath, and Ulrich Bodenhofer. 2025. *Msa: Multiple Sequence Alignment*. https://doi.org/10.18129/B9.bioc.msa.

Borman, Tuomas, Felix G. M. Ernst, and Leo Lahti. 2025. *miaViz: Microbiome Analysis Plotting and Visualization*. https://github.com/microbiome/miaViz.

Borman, Tuomas, Felix G. M. Ernst, Sudarshan A. Shetty, and Leo Lahti. 2025. *Mia: Microbiome Analysis*. https://microbiome.github.io/mia/.

Huang, Ruizhu. 2025. *TreeSummarizedExperiment: A S4 Class for Data with Tree Structures*.

Huang, Ruizhu, Charlotte Soneson, Felix G. M. Ernst, Kevin C. Rue-Albrecht, Guangchuang Yu, Stephanie C. Hicks, and Mark D. Robinson. 2021. "TreeSummarizedExperiment: A S4 Class for Data with Hierarchical Structure." *F1000Research* 9: 1246. https://f1000research.com/articles/9-1246.

McCarthy, Davis J., Kieran R. Campbell, Aaron T. L. Lun, and Quin F. Willis. 2017. "Scater: Pre-Processing, Quality Control, Normalisation and Visualisation of Single-Cell RNA-Seq Data in R." *Bioinformatics* 33: 1179–86. https://doi.org/10.1093/bioinformatics/btw777.

McCarthy, Davis, Kieran Campbell, Aaron Lun, and Quin Wills. 2025. *Scater: Single-Cell Analysis Toolkit for Gene Expression Data in r*. http://bioconductor.org/packages/scater/.

Oksanen, Jari, Gavin L. Simpson, F. Guillaume Blanchet, Roeland Kindt, Pierre Legendre, Peter R. Minchin, R. B. O'Hara, et al. 2025. *Vegan: Community Ecology Package*. https://vegandevs.github.io/vegan/.

Paradis, Emmanuel, Simon Blomberg, Ben Bolker, Joseph Brown, Santiago Claramunt, Julien Claude, Hoa Sien Cuong, et al. 2024. *Ape: Analyses of Phylogenetics and Evolution*. https://github.com/emmanuelparadis/ape.

Paradis, Emmanuel, and Klaus Schliep. 2019. "Ape 5.0: An Environment for Modern Phylogenetics and Evolutionary Analyses in R." *Bioinformatics* 35: 526–28. https://doi.org/10.1093/bioinformatics/bty633.

Pedersen, Thomas Lin. 2025. *Patchwork: The Composer of Plots*. https://patchwork.data-imaginist.com.

Schliep, Klaus. 2011. "Phangorn: Phylogenetic Analysis in r." *Bioinformatics* 27 (4): 592–93. https://doi.org/10.1093/bioinformatics/btq706.

Schliep, Klaus, Emmanuel Paradis, Leonardo de Oliveira Martins, Alastair Potts, and Iris Bardel-Kahr. 2025. *Phangorn: Phylogenetic Reconstruction and Analysis*. https://github.com/KlausVigo/phangorn.

Schliep, Klaus, Alastair J. Potts, David A. Morrison, and Guido W. Grimm. 2017. "Intertwining Phylogenetic Trees and Networks." *Methods in Ecology and Evolution* 8 (10): 1212–20.

Wickham, Hadley. 2016. *Ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. https://ggplot2.tidyverse.org.

———. 2023. *Tidyverse: Easily Install and Load the Tidyverse*. https://tidyverse.tidyverse.org.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, Winston Chang, Lionel Henry, Thomas Lin Pedersen, Kohske Takahashi, Claus Wilke, Kara Woo, Hiroaki Yutani, Dewey Dunnington, and Teun van den Brand. 2025. *Ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. https://ggplot2.tidyverse.

org.