7. Worksheet: Diversity Synthesis

Elaine Hoffman; Z620: Quantitative Biodiversity, Indiana University

18 February, 2025

OVERVIEW

In this worksheet, you will conduct exercises that reinforce fundamental concepts of biodiversity. First, you will construct a site-by-species matrix by sampling confectionery taxa from a source community. Second, you will make a preference-profile matrix, reflecting each student's favorite confectionery taxa. With this primary data structure, you will then answer questions and generate figures using tools from previous weeks, along with wrangling techniques that we learned about in class.

Directions:

1. In the Markdown version of this document in your cloned repo, change "Student Name" on line 3 (above) to your name.
2. Complete as much of the worksheet as possible during class.
3. Refer to previous handouts to help with developing of questions and writing of code.
4. Answer questions in the worksheet. Space for your answer is provided in this document and indicated by the ">" character. If you need a second paragraph be sure to start the first line with ">". You should notice that the answer is highlighted in green by RStudio (color may vary if you changed the editor theme).
5. Before you leave the classroom, **push** this file to your GitHub repo.
6. For the assignment portion of the worksheet, follow the directions at the bottom of this file.
7. When you are done, **Knit** the text and code into a PDF file.
8. After Knitting, submit the completed exercise by creating a **pull request** via GitHub. Your pull request should include this file `7.DiversitySynthesis_Worskheet.Rmd` and the PDF output of `Knitr` (`DiversitySynthesis_Worskheet.pdf`).

QUANTITATIVE CONFECTIONOLOGY

We will construct a site-by-species matrix using confectionery taxa (i.e, jelly beans). The instructors have created a **source community** with known abundance (N) and richness (S). Like a real biological community, the species abundances are unevenly distributed such that a few jelly bean types are common while most are rare. Each student will sample the source community and bin their jelly beans into operational taxonomic units (OTUs).

SAMPLING PROTOCOL: SITE-BY-SPECIES MATRIX

1. From the well-mixed source community, each student should take one Dixie Cup full of individuals.
2. At your desk, sort the jelly beans into different types (i.e., OTUs), and quantify the abundance of each OTU.
3. Working with other students, merge data into a site-by-species matrix with dimensions equal to the number of students (rows) and taxa (columns)
4. Create a worksheet (e.g., Google sheet) and share the site-by-species matrix with the class.

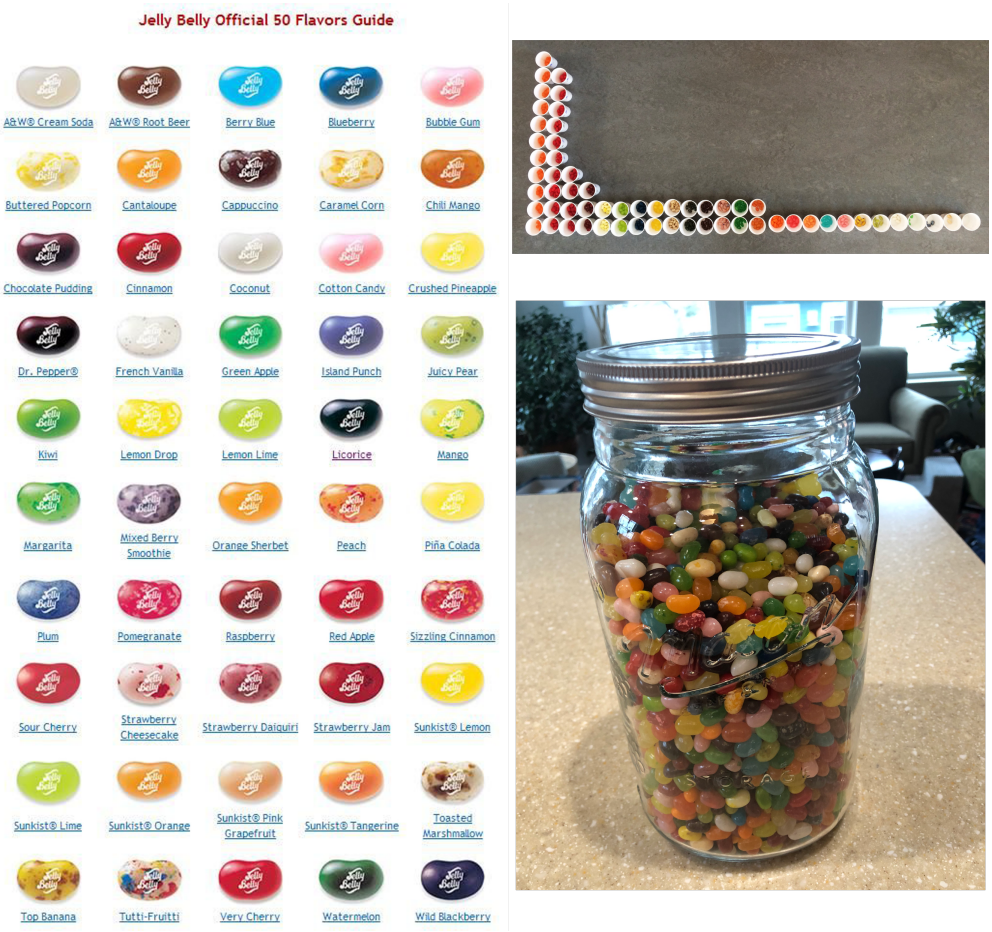


Figure 1: **Left**: taxonomic key, **Top right**: rank abundance distribution, **Bottom right**: source community

SAMPLING PROTOCOL: PREFERENCE-PROFILE MATRIX

1. With your individual sample only, each student should choose their top 5-10 preferred taxa based on flavor, color, sheen, etc.
2. Working with other students, merge data into preference-profile incidence matrix where 1 = preferred and 0 = non-preferred taxa.
3. Create a worksheet (e.g., Google sheet) and share the preference-profile matrix with the class.

1) R SETUP

In the R code chunk below, please provide the code to: 1) Clear your R environment, 2) Print your current working directory, 3) Set your working directory to your `Week5-Confection/` folder, and 4) Load the `vegan` R package (be sure to install first if you have not already).

```
rm(list = ls())
getwd()
```

```
## [1] "/cloud/project/Week5-Confection"
```

```
setwd("/cloud/project/Week5-Confection")

library(vegan)
```

```
## Loading required package: permute
```

```
## Loading required package: lattice
```

```
## This is vegan 2.6-8
```

```
library(BiodiversityR)
```

```
## Loading required package: tcltk
```

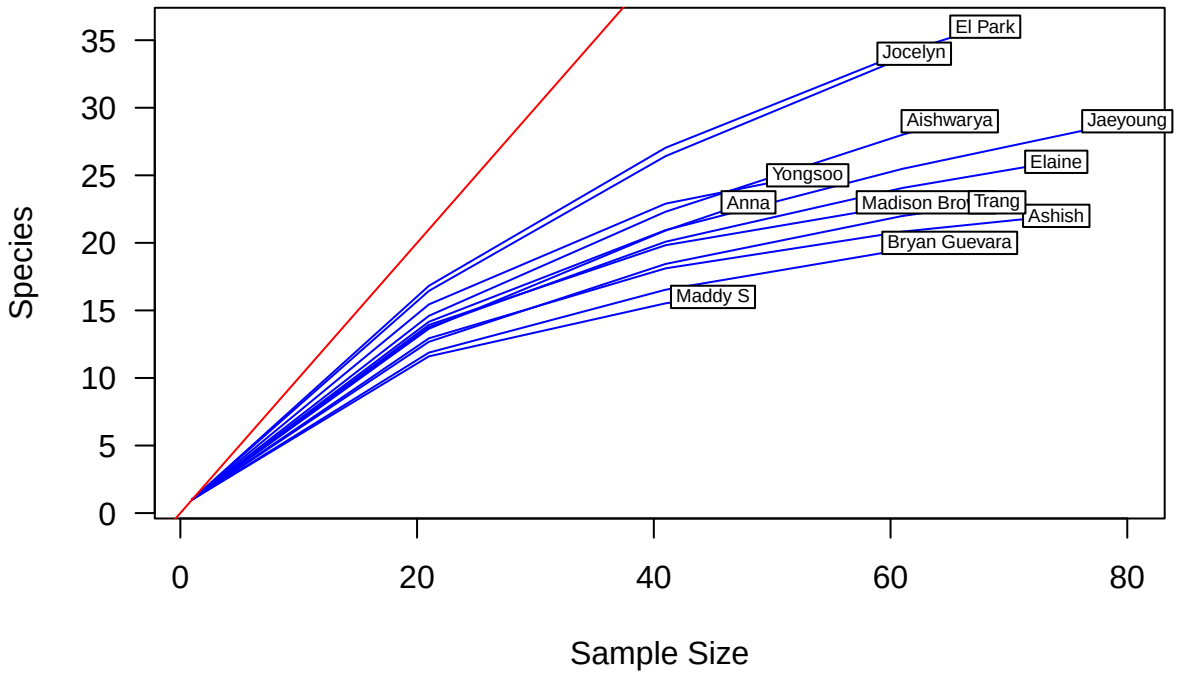
```
## Warning in fun(libname, pkgname): couldn't connect to display ":0"
```

```
## BiodiversityR 2.17-1.1: Use command BiodiversityRGUI() to launch the Graphical User Interface;
## to see changes use BiodiversityRGUI(changeLog=TRUE, backward.compatibility.messages=TRUE)
```

DATA ANALYSIS

Question 1: In the space below, generate a rarefaction plot for all samples of the source community. Based on these results, discuss how individual vs. collective sampling efforts capture the diversity of the source community.

```
beans <- as.matrix(read.csv(file = "./data/Jelly belly data! - SbyS.csv", header = TRUE, row.names = 1))

S.obs <- function(x = ""){
  rowSums(x > 0) * 1
}

obs.beans <- S.obs(beans)
min.N <- min(rowSums(beans))
S.rarefy <- rarefy(x = beans, sample = min.N, se = TRUE)
rarecurve(x = beans, step = 20, col = "blue", cex = 0.6, las = 1)
abline(0, 1, col = 'red')
text(1500, 1500, "1:1", pos = 2, col = 'red')
```

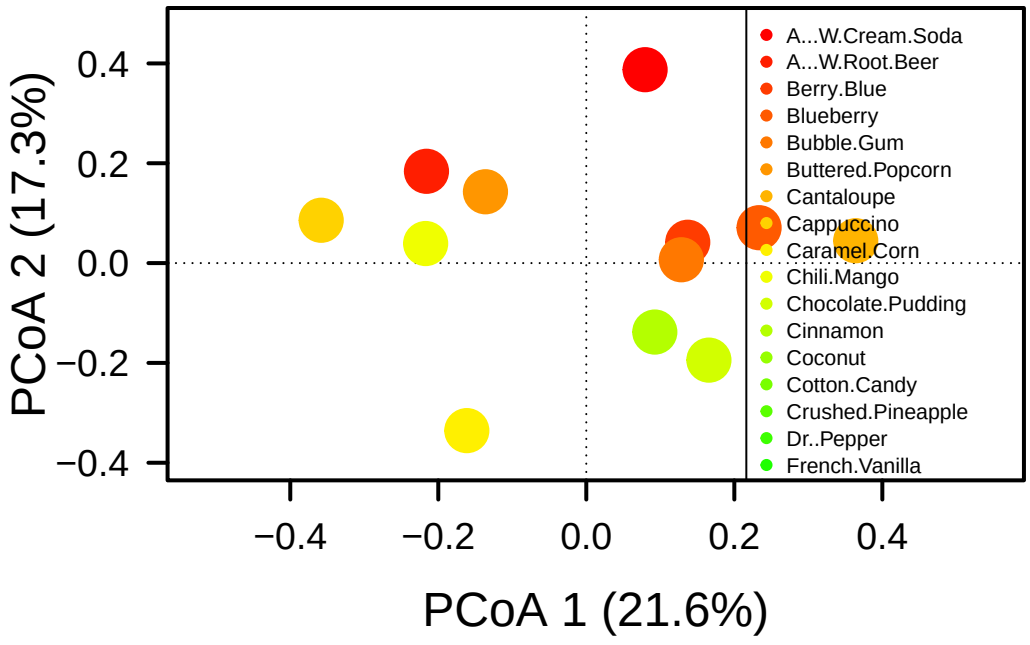


Answer 1:By looking at the variation in the lines for each of our samples, it is clear that collective sampling efforts do a better job at capturing the diversity of the jelly bean population. The number of species observed ranges from ~12 to ~35 by estimating on the graph. This is a huge amount of variation and highlights the importance of multiple sampling events to achieve accuracy.

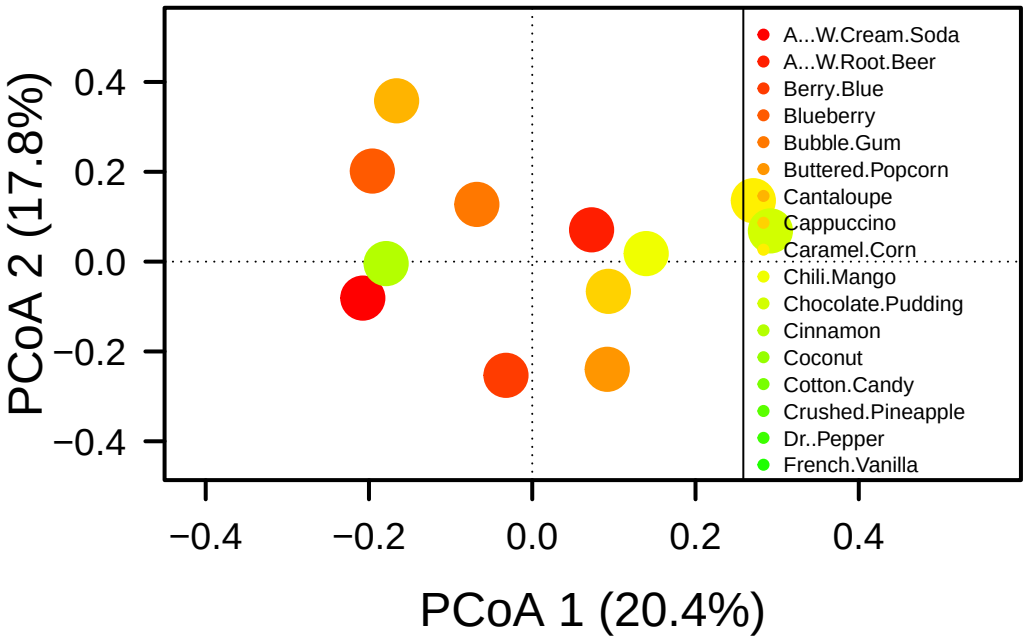
Question 2: Starting with the site-by-species matrix, visualize beta diversity. In the code chunk below, conduct principal coordinates analyses (PCoA) using both an abundance- and incidence-based resemblance matrix. Plot the sample scores in species space using different colors, symbols, or labels. Which "species" are contributing the patterns in the ordinations? How does the choice of resemblance matrix affect your interpretation?

```
beans.dj <- vegdist(beans, method = "jaccard", binary = TRUE)
beans.db <- vegdist(beans, method = "bray")

beans.pcoa.dj <- cmdscale(beans.dj, eig = TRUE, k = 3)
beans.pcoa.db <- cmdscale(beans.db, eig = TRUE, k = 3)

explainvar1.dj <- round(beans.pcoa.dj$eig[1] / sum(beans.pcoa.dj$eig), 3) * 100
explainvar2.dj <- round(beans.pcoa.dj$eig[2] / sum(beans.pcoa.dj$eig), 3) * 100
explainvar3.dj <- round(beans.pcoa.dj$eig[3] / sum(beans.pcoa.dj$eig), 3) * 100
sum.eig.dj <- sum(explainvar1.dj, explainvar2.dj, explainvar3.dj)

explainvar1.db <- round(beans.pcoa.db$eig[1] / sum(beans.pcoa.db$eig), 3) * 100
explainvar2.db <- round(beans.pcoa.db$eig[2] / sum(beans.pcoa.db$eig), 3) * 100
explainvar3.db <- round(beans.pcoa.db$eig[3] / sum(beans.pcoa.db$eig), 3) * 100
sum.eig.db <- sum(explainvar1.db, explainvar2.db, explainvar3.db)

# Adjust margins to ensure all points and labels are visible
par(mar = c(6, 6, 4, 4) + 0.1)

# Calculate axis limits based on point positions
x_range <- range(jitter(beans.pcoa.dj$points[,1], amount = 0.1)) + c(-0.2, 0.2)
y_range <- range(jitter(beans.pcoa.dj$points[,2], amount = 0.1)) + c(-0.2, 0.2)
```

```r
# Generate distinct colors for each species
species_colors <- rainbow(ncol(beans))

# Plot the PCoA points without points initially
plot(beans.pcoa.dj$points[,1], beans.pcoa.dj$points[,2],
     xlim = x_range, ylim = y_range,
     xlab = paste("PCoA 1 (", explainvar1.dj, "%)", sep = ""),
     ylab = paste("PCoA 2 (", explainvar2.dj, "%)", sep = ""),
     pch = 16, cex = 2.0, type = "n", cex.lab = 1.5,
     cex.axis = 1.2, axes = FALSE)

# Add axes and reference lines
axis(side = 1, labels = TRUE, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = TRUE, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

# Add jittered points with different colors for each species
set.seed(123)  # Ensure reproducible jitter
for (i in 1:nrow(beans.pcoa.dj$points)) {
  points(jitter(beans.pcoa.dj$points[i, 1], amount = 0.1),
         jitter(beans.pcoa.dj$points[i, 2], amount = 0.1),
         pch = 19, cex = 3, col = species_colors[i])
}

# Add jittered species names as labels
#text(jitter(beans.pcoa.dj$points[,1], amount = 0.1),
     #jitter(beans.pcoa.dj$points[,2], amount = 0.1),
     #labels = colnames(beans), col = species_colors, cex = 1)

# Add a legend to explain the colors
legend("topright", legend = colnames(beans), col = species_colors, pch = 19, cex = 0.7)
```

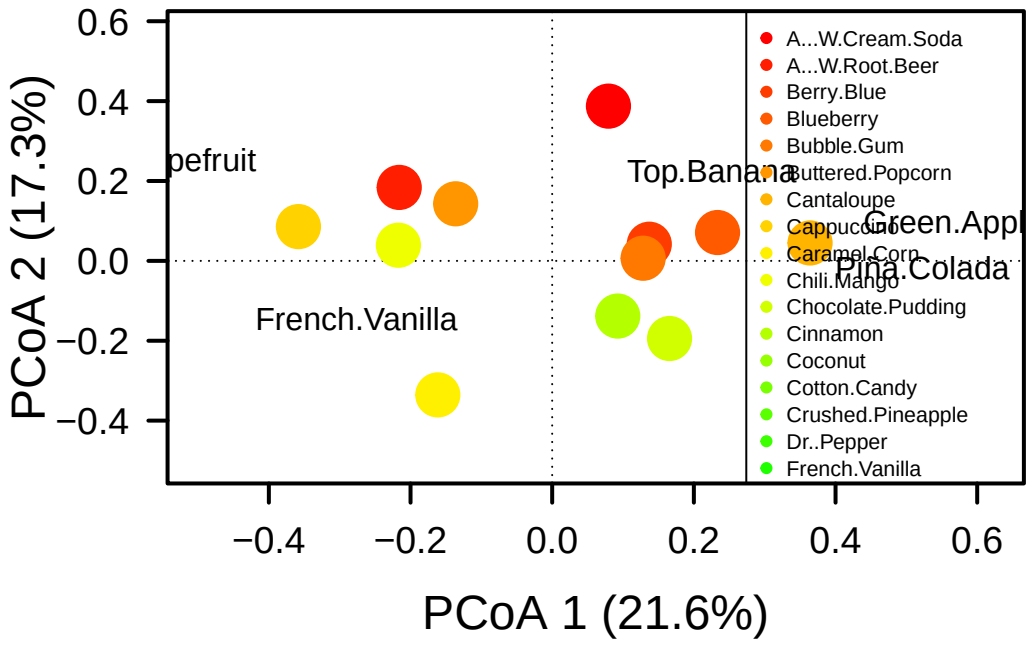


```r
# Adjust margins to ensure all points and labels are visible
par(mar = c(6, 6, 4, 4) + 0.1)

# Calculate axis limits based on point positions
x_range <- range(jitter(beans.pcoa.db$points[,1], amount = 0.1)) + c(-0.2, 0.2)
y_range <- range(jitter(beans.pcoa.db$points[,2], amount = 0.1)) + c(-0.2, 0.2)

# Generate distinct colors for each species
species_colors <- rainbow(ncol(beans))

# Plot the PCoA points without points initially
plot(beans.pcoa.db$points[,1], beans.pcoa.db$points[,2],
     xlim = x_range, ylim = y_range,
     xlab = paste("PCoA 1 (", explainvar1.db, "%)", sep = ""),
     ylab = paste("PCoA 2 (", explainvar2.db, "%)", sep = ""),
     pch = 16, cex = 2.0, type = "n", cex.lab = 1.5,
     cex.axis = 1.2, axes = FALSE)

# Add axes and reference lines
axis(side = 1, labels = TRUE, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = TRUE, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

# Add jittered points with different colors for each species
set.seed(123)  # Ensure reproducible jitter
for (i in 1:nrow(beans.pcoa.db$points)) {
  points(jitter(beans.pcoa.db$points[i, 1], amount = 0.1),
         jitter(beans.pcoa.db$points[i, 2], amount = 0.1),
         pch = 19, cex = 3, col = species_colors[i])
}

# Add a legend to explain the colors
legend("topright", legend = colnames(beans), col = species_colors, pch = 19, cex = 0.7)
```



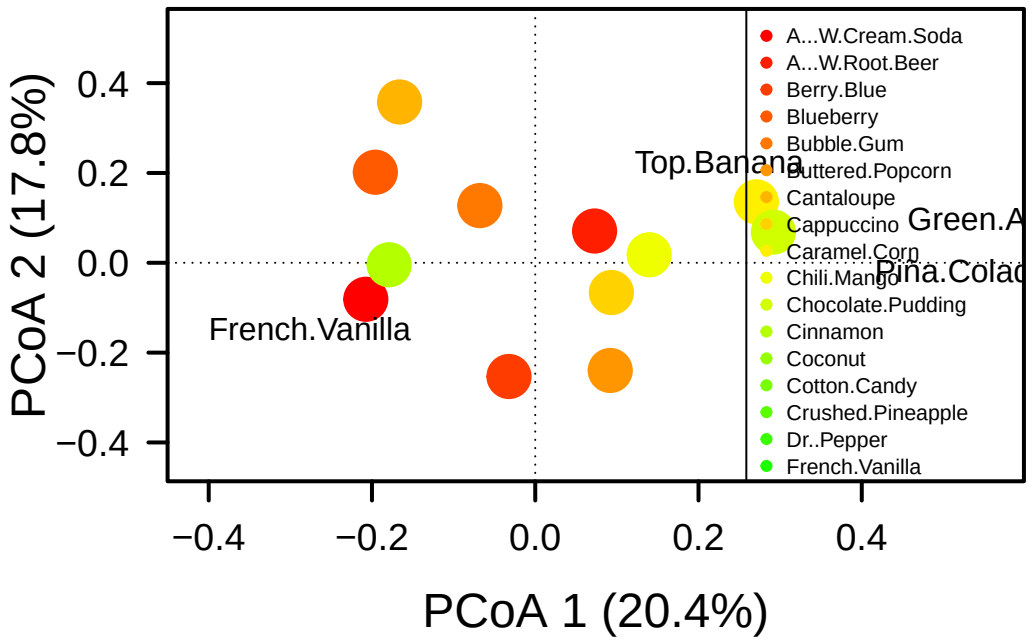

```r
beansREL <- beans
for(i in 1:nrow(beans)){
  beansREL[i, ] = beans[i, ] / sum(beans[i, ])
}

# Identify columns with zero variance
zero_var_species <- apply(beans, 2, function(x) var(x) == 0)

# Remove columns with zero variance
beans_clean <- beans[, !zero_var_species]

# Now run the function again
beans.pcoa.dj <- add.spec.scores(beans.pcoa.dj, beans_clean, method = "pcoa.scores")


# Adjust margins to ensure all points and labels are visible
par(mar = c(6, 6, 4, 4) + 0.1)

# Calculate axis limits based on point positions
x_range <- range(jitter(beans.pcoa.dj$points[,1], amount = 0.1)) + c(-0.2, 0.2)
y_range <- range(jitter(beans.pcoa.dj$points[,2], amount = 0.1)) + c(-0.2, 0.2)

# Generate distinct colors for each species
species_colors <- rainbow(ncol(beans))

# Plot the PCoA points without points initially
plot(beans.pcoa.dj$points[,1], beans.pcoa.dj$points[,2],
     xlim = x_range, ylim = y_range,
     xlab = paste("PCoA 1 (", explainvar1.dj, "%)", sep = ""),
     ylab = paste("PCoA 2 (", explainvar2.dj, "%)", sep = ""),
     pch = 16, cex = 2.0, type = "n", cex.lab = 1.5,
     cex.axis = 1.2, axes = FALSE)
```

```r
# Add axes and reference lines
axis(side = 1, labels = TRUE, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = TRUE, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

# Add jittered points with different colors for each species
set.seed(123)  # Ensure reproducible jitter
for (i in 1:nrow(beans.pcoa.dj$points)) {
  points(jitter(beans.pcoa.dj$points[i, 1], amount = 0.1),
         jitter(beans.pcoa.dj$points[i, 2], amount = 0.1),
         pch = 19, cex = 3, col = species_colors[i])
}

# Add jittered species names as labels
text(beans.pcoa.dj$cproj[ ,1], beans.pcoa.dj$cproj[ ,2],
     labels = colnames(beans), col = "black")

# Add a legend to explain the colors
legend("topright", legend = colnames(beans), col = species_colors, pch = 19, cex = 0.7)
```




```r
# Now run the function again
beans.pcoa.db <- add.spec.scores(beans.pcoa.db, beans_clean, method = "pcoa.scores")


# Adjust margins to ensure all points and labels are visible
par(mar = c(6, 6, 4, 4) + 0.1)

# Calculate axis limits based on point positions
x_range <- range(jitter(beans.pcoa.db$points[,1], amount = 0.1)) + c(-0.2, 0.2)
y_range <- range(jitter(beans.pcoa.db$points[,2], amount = 0.1)) + c(-0.2, 0.2)

# Generate distinct colors for each species
```

```r
species_colors <- rainbow(ncol(beans))

# Plot the PCoA points without points initially
plot(beans.pcoa.db$points[,1], beans.pcoa.db$points[,2],
     xlim = x_range, ylim = y_range,
     xlab = paste("PCoA 1 (", explainvar1.db, "%)", sep = ""),
     ylab = paste("PCoA 2 (", explainvar2.db, "%)", sep = ""),
     pch = 16, cex = 2.0, type = "n", cex.lab = 1.5,
     cex.axis = 1.2, axes = FALSE)

# Add axes and reference lines
axis(side = 1, labels = TRUE, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = TRUE, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

# Add jittered points with different colors for each species
set.seed(123)  # Ensure reproducible jitter
for (i in 1:nrow(beans.pcoa.db$points)) {
  points(jitter(beans.pcoa.db$points[i, 1], amount = 0.1),
         jitter(beans.pcoa.db$points[i, 2], amount = 0.1),
         pch = 19, cex = 3, col = species_colors[i])
}
# Add jittered species names as labels
text(beans.pcoa.dj$cproj[ ,1], beans.pcoa.dj$cproj[ ,2],
     labels = colnames(beans), col = "black")

# Add a legend to explain the colors
legend("topright", legend = colnames(beans), col = species_colors, pch = 19, cex = 0.7)
```



Answer 2: In the Jaccard PCoA, the species contributing to the patterns are grapefruit, french vanilla, and top. banana. In the Bray-Curtis PCoA, the species contributing to the patterns are french vanilla and top. banana. The choice of the resemblance matrix revealed an extra influential

species with the Jaccard matrix. However, there were not significant differences in the variation explained by the axes between the two matrices (Jaccard: 17.3% & 21.6%, Bray: 17.8% & 20.4%).

Question 3 Using the preference-profile matrix, determine the most popular jelly bean in the class using a control structure (e.g., for loop, if statement, function, etc).

```
# Load the preference matrix
favorite <- as.matrix(read.csv(file = "./data/Jelly belly data! - Preference.csv",
                               header = TRUE, row.names = 1))

# Initialize variables to track the most popular jelly bean
max_count <- 0
most_popular <- ""

# Calculate the total preference count for each jelly bean
for (i in 1:ncol(favorite)) {
  # Calculate sum while ignoring NA values
  current_sum <- sum(favorite[, i], na.rm = TRUE)

  # Check if this flavor has the highest count so far
  if (current_sum > max_count) {
    max_count <- current_sum
    most_popular <- colnames(favorite)[i]
  }
}

# Print the result
cat("The most popular jelly bean is:", most_popular, "with", max_count, "votes.\n")
```

```
## The most popular jelly bean is: Berry.Blue with 7 votes.
```

Answer 3:The most popular jelly bean is: Berry.Blue with 7 votes.

Question 4 In the code chunk below, identify the student in QB who has a preference-profile that is most like yours. Quantitatively, how similar are you to your "jelly buddy"? Visualize the preference profiles of the class by creating a cluster dendogram. Label each terminal node (a.k.a., tip or "leaf") with the student's name or initials. Make some observations about the preference-profiles of the class.

```
# Load necessary libraries
library(vegan)       # For distance calculations
library(dendextend)  # For enhanced dendrogram visualization
```

```
## Registered S3 method overwritten by 'dendextend':
##   method     from
##   rev.hclust vegan
```

```
##
## ---------------------
## Welcome to dendextend version 1.19.0
## Type citation('dendextend') for how to cite the package.
##
## Type browseVignettes(package = 'dendextend') for the package vignette.
## The github page is: https://github.com/talgalili/dendextend/
##
## Suggestions and bug-reports can be submitted at: https://github.com/talgalili/dendextend/issues
## You may ask questions at stackoverflow, use the r and dendextend tags:
##   https://stackoverflow.com/questions/tagged/dendextend
##
```

```
##  To suppress this message use:  suppressPackageStartupMessages(library(dendextend))
## ----------------------
```

```
##
## Attaching package: 'dendextend'
```

```
## The following object is masked from 'package:permute':
##
##     shuffle
```

```
## The following object is masked from 'package:stats':
##
##     cutree
```

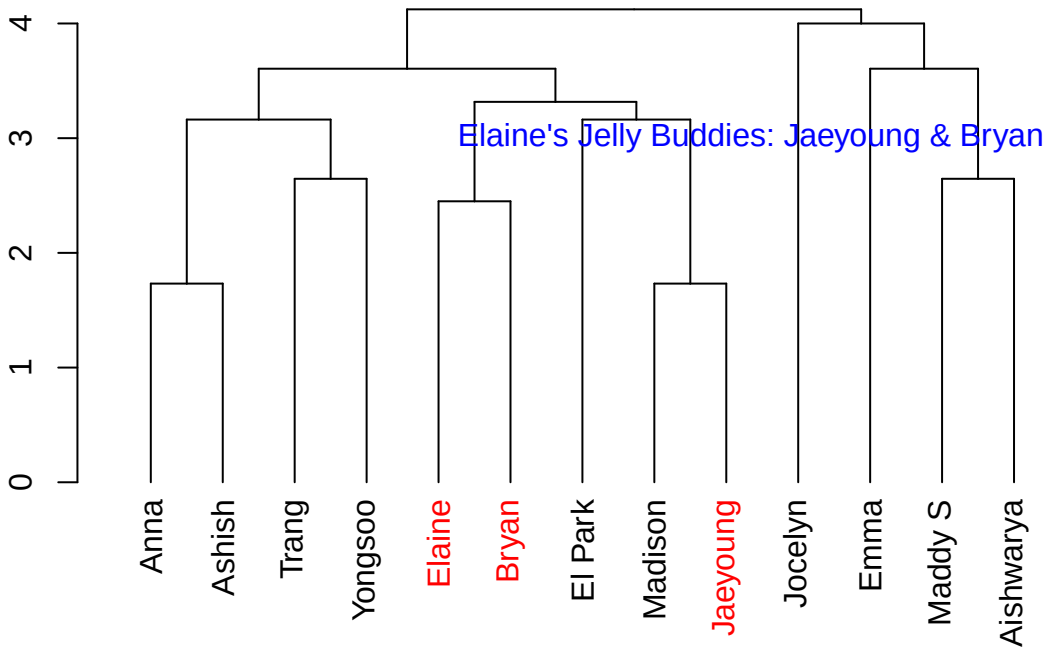
```
library(ggplot2)     # Optional for additional plot customization

# Load the preference matrix
favorite <- as.matrix(read.csv(file = "./data/Jelly belly data! - Preference.csv",
                               header = TRUE, row.names = 1))

# Replace missing values with zero
favorite[is.na(favorite)] <- 0

# Calculate Euclidean distance matrix
dist_matrix <- as.matrix(dist(favorite, method = "euclidean"))

# Extract distances for Elaine
elaine_distances <- dist_matrix["Elaine", ]
elaine_distances["Elaine"] <- Inf  # Exclude Elaine herself

# Identify the closest two students
sorted_distances <- sort(elaine_distances)
closest_students <- names(sorted_distances[1:2])
closest_distance <- sorted_distances[1]

# Print the results
cat("Elaine's jelly buddies are:", closest_students[1], "and", closest_students[2], "\n")
```

```
## Elaine's jelly buddies are: Jaeyoung and Bryan
```

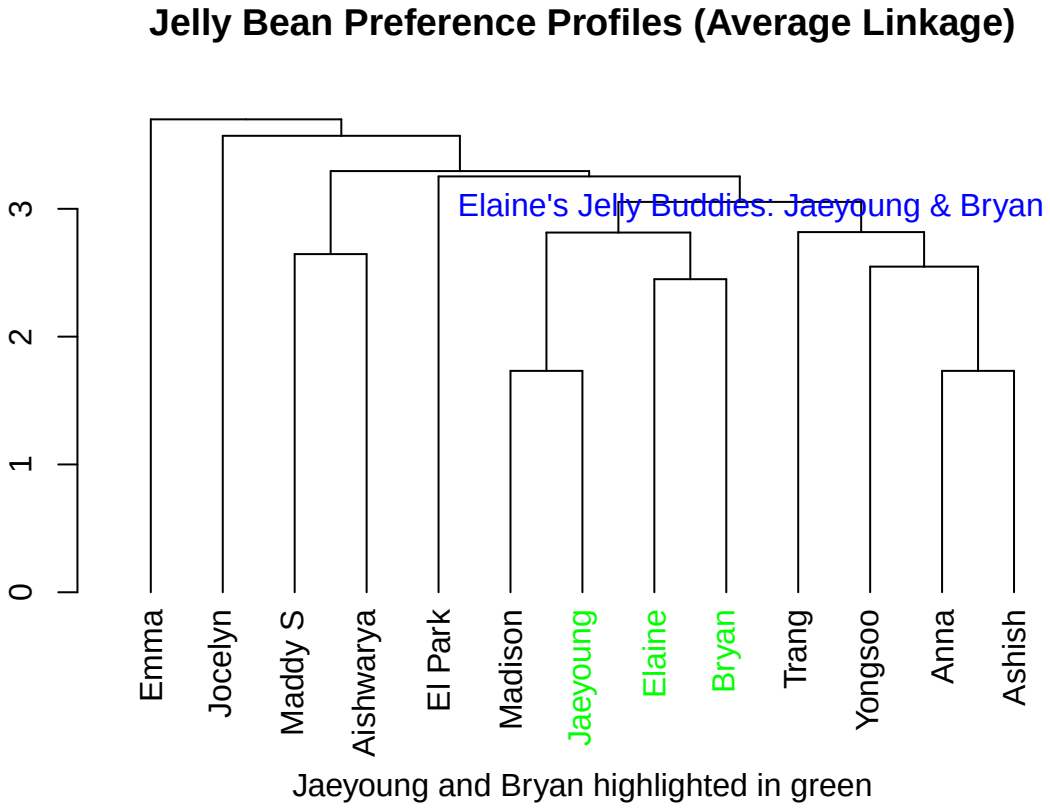
```
cat("Minimum Euclidean distance:", round(closest_distance, 3), "\n")
```

```
## Minimum Euclidean distance: 2.449
```

```
# Perform hierarchical clustering using complete linkage
cluster_complete <- hclust(dist(favorite, method = "euclidean"), method = "complete")

# Convert to dendrogram and color Elaine's buddies
dend_complete <- as.dendrogram(cluster_complete)
dend_complete <- color_labels(dend_complete, labels = c("Elaine", closest_students), col = "red")

# Plot the complete linkage dendrogram
plot(dend_complete, main = "Jelly Bean Preference Profiles (Complete Linkage)",
     sub = "Jaeyoung and Bryan highlighted in red", cex = 0.9)

# Add annotation
text(x = 5, y = 3, "Elaine's Jelly Buddies: Jaeyoung & Bryan", col = "blue", pos = 4)
```




```r
# Perform hierarchical clustering using average linkage
cluster_avg <- hclust(dist(favorite, method = "euclidean"), method = "average")

# Convert to dendrogram and color Elaine's buddies
dend_avg <- as.dendrogram(cluster_avg)
dend_avg <- color_labels(dend_avg, labels = c("Elaine", closest_students), col = "green")

# Plot the average linkage dendrogram
plot(dend_avg, main = "Jelly Bean Preference Profiles (Average Linkage)",
     sub = "Jaeyoung and Bryan highlighted in green", cex = 0.9)

# Add annotation
text(x = 5, y = 3, "Elaine's Jelly Buddies: Jaeyoung & Bryan", col = "blue", pos = 4)
```




```
# Observations:
# - Both Jaeyoung and Bryan have equal distances (2.475) to Elaine.
# - The difference in leaf position results from the clustering method.
# - Complete linkage may prioritize cluster-level distances, causing Bryan to appear closer.
```

Answer 4: It turns out that I have two jelly buddies. I am just as similar to Jaeyoung as I am to Bryan with a euclidean distance of 2.475. When looking at the dendrogram, it appears that the class sorts into a few clusters of similar preferences. I changed the structure of the dendrogram because it was showing Jaeyoung really far away from me just based on how it is clustering, but that leads me to believe that the differences between all of our preferences are minimal.

SUBMITTING YOUR ASSIGNMENT

Use Knitr to create a PDF of your completed `7.DiversitySynthesis_Worksheet.Rmd` document, push it to GitHub, and create a pull request. Please make sure your updated repo includes both the pdf and RMarkdown files.

Unless otherwise noted, this assignment is due on **Wednesday, February 19th, 2025 at 12:00 PM (noon)**.