7. Worksheet: Diversity Synthesis

Trang Nguyen; Z620: Quantitative Biodiversity, Indiana University

19 février, 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.

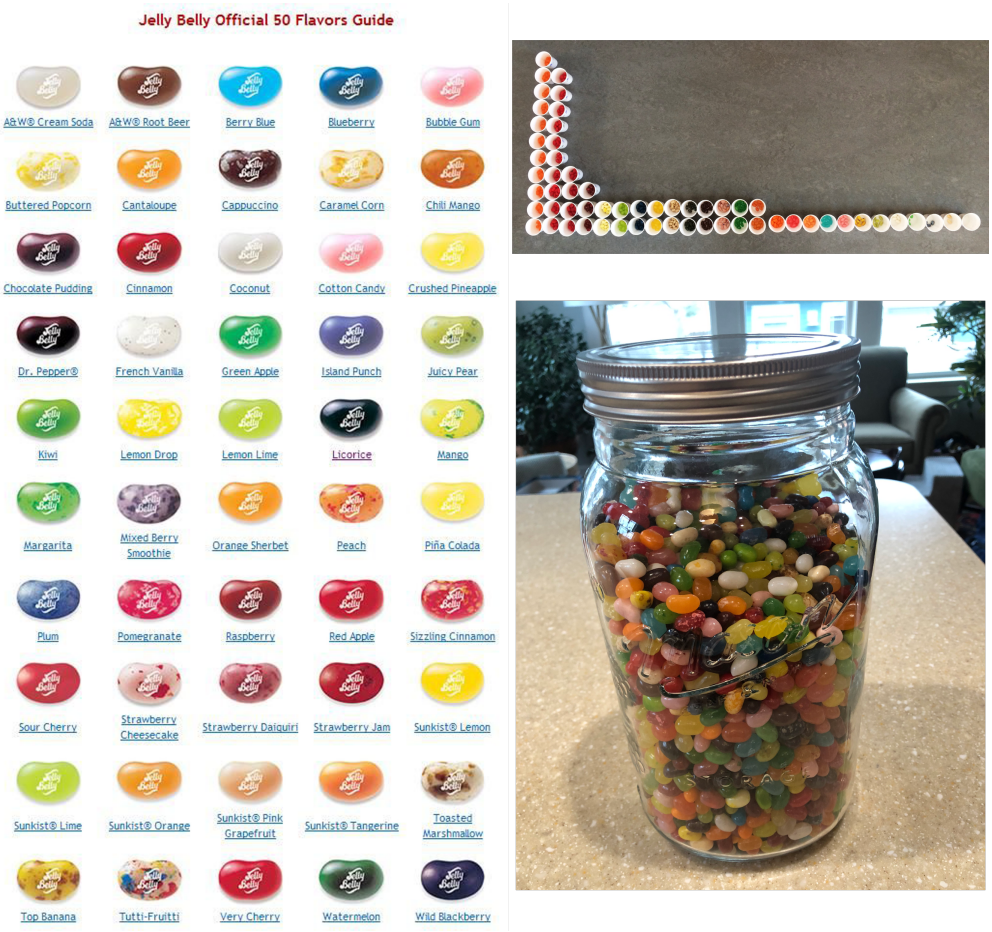


Figure 1: **Left**: taxonomic key, **Top right**: rank abundance distribution, **Bottom right**: source community

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).

```r
data = read.csv("./data/jelly.source.comm.csv")
data
```

```
##                     Type abundance
## 1                 cherry       244
## 2             juicy.pear       177
## 3              blueberry       142
## 4              tangerine       128
## 5               coconunt       123
## 6                  peach       119
## 7                  lemon       109
## 8              red.apple       101
## 9      sizzling.cinnamon        96
## 10         blue.rasberry        95
## 11                  kiwi        95
## 12     crushed.pineapple        92
## 13        wild.blueberry        91
## 14            lemon.drop        91
## 15           green.apple        90
## 16 strawberry.cheesecake        86
## 17           grape.crush        82
## 18             dr.pepper        81
## 19           pomegranate        81
## 20          orange.crush        81
## 21           carmel.corn        74
## 22             margarita        74
## 23            berry.blue        72
## 24        strawberry.jam        70
## 25             bubblegum        68
## 26         aw.cream.soda        68
## 27            cantaloupe        63
## 28        french.vanilla        63
## 29          cotton.candy        59
## 30     stawbery.daiquiri        58
## 31                  plum        57
## 32          tutti.frutti        57
## 33           watermellon        55
## 34                 mango        54
## 35           lemon.lime        50
## 36   toasted.marshmallow        50
## 37              cinnamon        50
## 38                orange        49
## 39      buttered.popcorn        47
## 40       orange.sherbert        44
## 41          aw.root.beer        42
## 42           top.bananna        40
## 43           pina.colada        34
## 44           sour.orange        29
## 45            sour.green        26
## 46           sour.yellow        23
## 47             sour.blue        19
## 48              sour.red        13
```

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

```
## [1] "C:/Users/ttran/OneDrive - Indiana University/SP25 - Quantitative Biodiversity/QB2025_Nguyen/Weel
```

```r
setwd(getwd())
library(vegan)
```

```
## Warning: le package 'vegan' a été compilé avec la version R 4.4.2

## Le chargement a nécessité le package : permute

## Warning: le package 'permute' a été compilé avec la version R 4.4.2

## Le chargement a nécessité le package : lattice

## This is vegan 2.6-8
```

```r
library(BiodiversityR)
```

```
## Warning: le package 'BiodiversityR' a été compilé avec la version R 4.4.2

## Le chargement a nécessité le package : tcltk

## 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)
```

```r
# Load data
sbys = read.table("./data/Jb_sbs.tsv", header = TRUE, sep = "\t", row.names = 1)
pref = read.table("./data/Jb_pref.tsv", header = TRUE, sep = "\t", row.names = 1)

# Clean data
sum(is.na(sbys)) # check NA
```

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

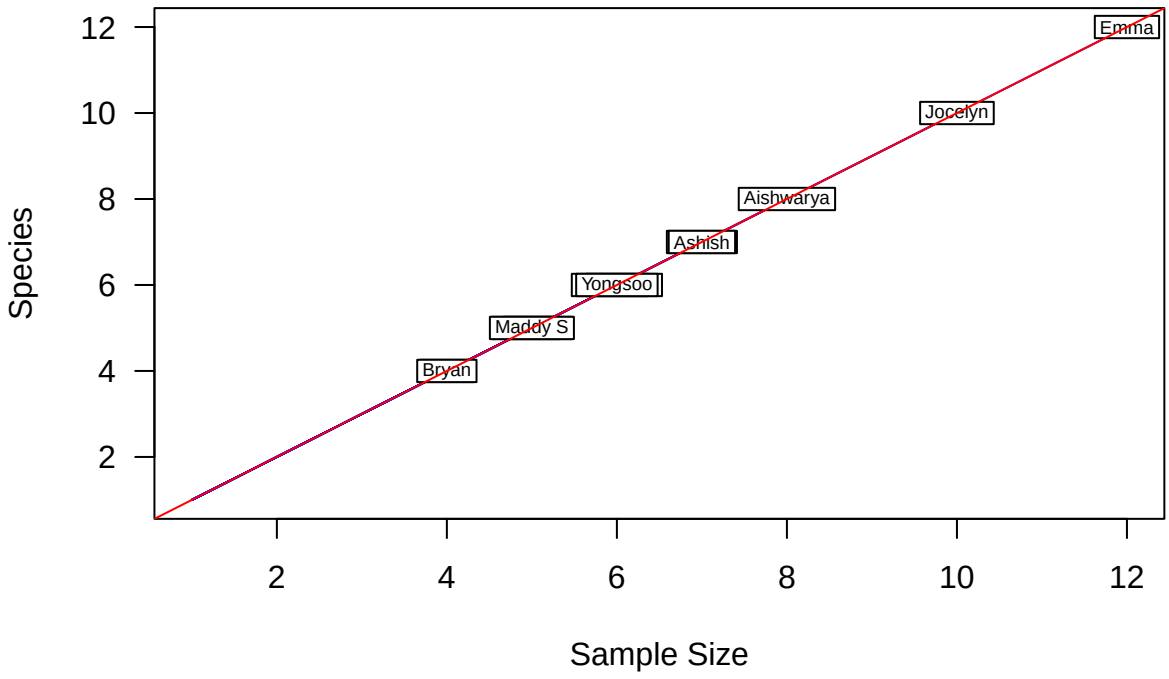
```r
sbys[is.na(sbys)] <- 0

# Clean preference data
sum(is.na(pref)) # check NA
```

```
## [1] 0
```

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.

```r
# Remove rows with zero counts
sbys <- sbys[rowSums(sbys) > 0, ]

# Find minimum sample size for rarefaction
min.N_sbys <- min(rowSums(sbys))

# Perform rarefaction (estimates richness but does NOT modify raw data)
sbys.rarefy <- rarefy(sbys, sample = min.N_sbys, se = TRUE)

# Plot rarefaction curves using the ORIGINAL sbys
rarecurve(sbys, step=20, col="blue", cex=0.6, las=1)
# 5. Add the 1:1 line and label.
abline(0, 1, col="red")
text(1500, 1500, "1:1 line", col="red", pos=2)
```



Answer 1: The rarefaction plot allows us to compare species richness across different sample sizes. These curves show that as more individuals are sampled, the number of observed species increases but never reaches an asymptote. This suggests that the jar had has lots of different jelly beans types and that new types continue to be found with additional sampling. The rarefaction curves also show that the class sampling efforts capture only a small fraction of the total diversity in jar.

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?

```
# Calculate Bray-Curtis distance
sbys.db = vegdist(sbys, method = "bray")

# PCoA
sbys.pcoa = cmdscale(sbys.db, k = 2, eig = TRUE)
sbys.pcoa
```

```
## $points
##                  [,1]        [,2]
## El Park   -0.22944522  0.04788401
## Trang     -0.03324229 -0.21846275
## Madison   -0.27716445 -0.18938405
## Emma       0.02519591  0.16036267
## Maddy S    0.40696868  0.37787031
## Anna       0.48376068 -0.18481205
## Jaeyoung  -0.21400181 -0.25068859
## Elaine    -0.46443202 -0.10475063
## Jocelyn   -0.23540435  0.21980461
## Bryan     -0.30059050  0.22327415
## Aishwarya  0.16119134  0.46895753
## Yongsoo    0.23830409 -0.32500658
## Ashish     0.43885993 -0.22504865
##
## $eig
##  [1]  1.213486e+00  8.367230e-01  6.753000e-01  5.552262e-01  4.287226e-01
##  [6]  3.354262e-01  1.870910e-01  1.024055e-01  8.558913e-02  1.387779e-16
## [11] -4.282794e-02 -7.233264e-02 -1.473726e-01
##
## $x
## NULL
##
## $ac
## [1] 0
##
## $GOF
## [1] 0.4378447 0.4638514
```

```
# Explained variance for first 3 axes
explained_variance = round(sum(sbys.pcoa$eig[1:3]/sum(sbys.pcoa$eig)) * 100, 2)
print("Explained variance of the 3 first axes:")
```

```
## [1] "Explained variance of the 3 first axes:"
```

```
print(explained_variance)
```

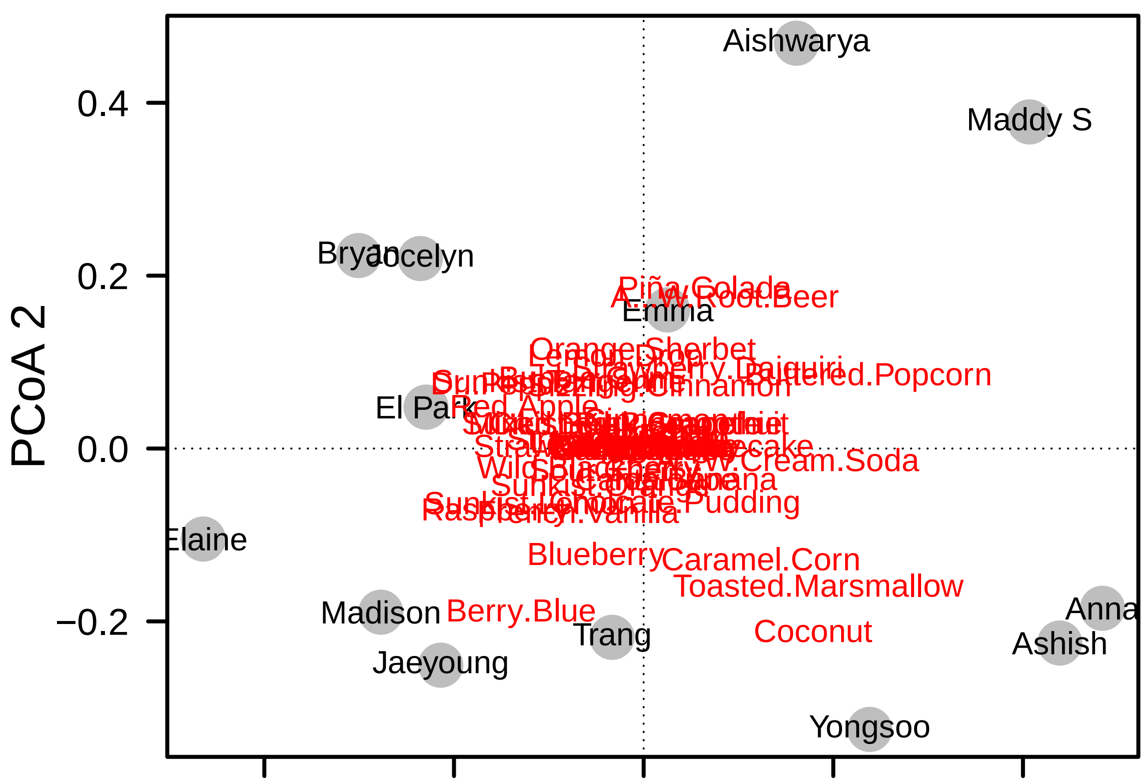
```
## [1] 65.56
```

```r
## Plot PCoA
##===============================
# Define Plot Parameters
par(mar=c(1,5,2,2) + 0.1)

# Initiate Plot
plot(sbys.pcoa$points[ ,1],
  sbys.pcoa$points[ ,2],
     xlab = "PCoA 1",
     ylab = "PCoA 2",
     pch = 16, cex = 2,  type = "n", cex.lab = 1.5,
     cex.axis = 0.5, axes = FALSE)

# # Add Axes
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

# # Add People
points(sbys.pcoa$points[ ,1],
  sbys.pcoa$points[ ,2],
  pch = 19, cex = 3, bg = "gray", col = "gray")
text(sbys.pcoa$points[ ,1], sbys.pcoa$points[ ,2],
     labels = row.names(sbys.pcoa$points))

# Add candies
sbysCAN <- sbys
for(i in 1:nrow(sbys)){
    sbysCAN[i, ] = sbys[i, ] / sum(sbys[i, ])
    }
# Plot the candies
sbys.pcoa <- add.spec.scores(
  sbys.pcoa,
  sbysCAN,
  method = "pcoa.scores")
```

```
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
```

```
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l’écart type est nul
```

```r
text(
  sbys.pcoa$cproj[ ,1],
  sbys.pcoa$cproj[ ,2],
  labels = row.names(sbys.pcoa$cproj), col = "red",
  cex=1)
```




```r
# Try another dissimilarities
sbys.dk = vegdist(sbys, method = "euclidean")

# PCoA
sbys.pcoa = cmdscale(sbys.dk, k = 2, eig = TRUE)
sbys.pcoa
```

```
## $points
```

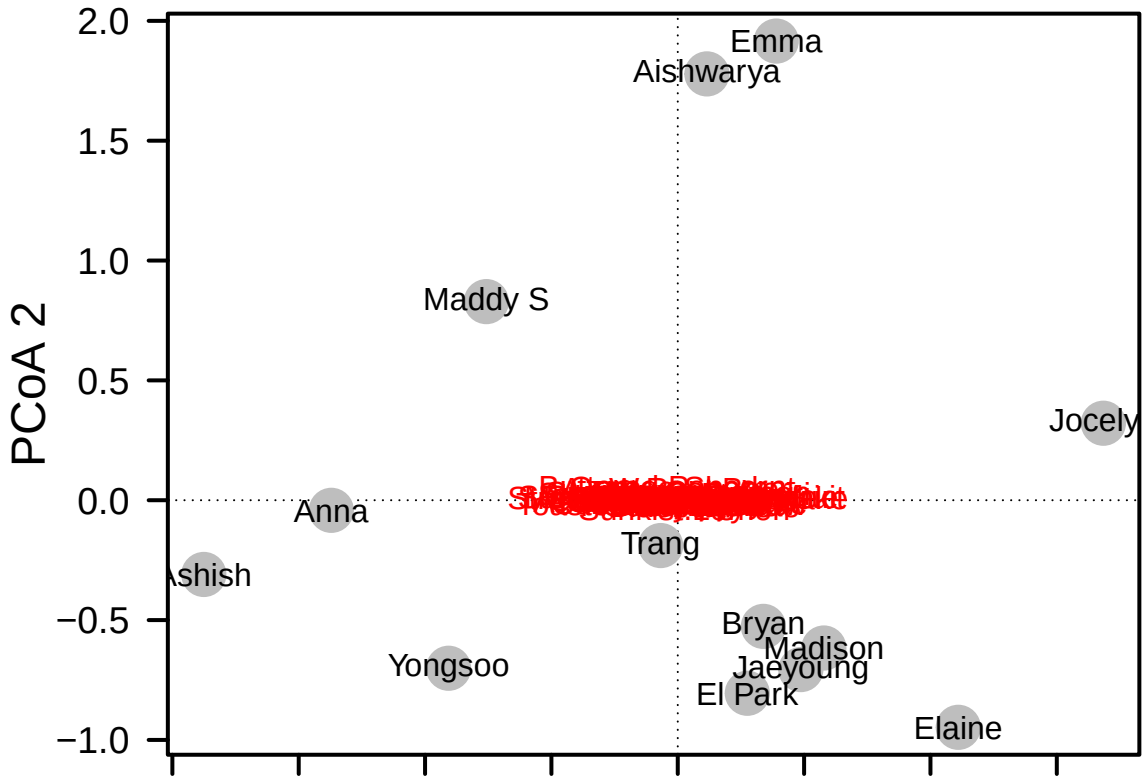
```
##                  [,1]        [,2]
## El Park     0.27521169 -0.80537694
## Trang      -0.06754689 -0.18911571
## Madison     0.57819995 -0.61654448
## Emma        0.38970067  1.91561109
## Maddy S    -0.75751984  0.82865675
## Anna       -1.37106334 -0.04215634
## Jaeyoung    0.48698399 -0.70654762
## Elaine      1.11008865 -0.94737546
## Jocelyn     1.68427596  0.32126431
## Bryan       0.33889797 -0.52632490
## Aishwarya   0.11530708  1.77864942
## Yongsoo    -0.90694641 -0.70101432
## Ashish     -1.87558946 -0.30972581
##
## $eig
##  [1] 1.179491e+01 1.095044e+01 8.335795e+00 7.616381e+00 6.362360e+00
##  [6] 4.907569e+00 4.413489e+00 2.968881e+00 2.558193e+00 1.742509e+00
## [11] 1.146210e+00 7.417261e-01 4.111282e-16
##
## $x
## NULL
##
## $ac
## [1] 0
##
## $GOF
## [1] 0.3579777 0.3579777
```

```r
# Explained variance for first 3 axes
explained_variance = round(sum(sbys.pcoa$eig[1:3]/sum(sbys.pcoa$eig)) * 100, 2)

## Plot PCoA
##==============================
# Define Plot Parameters
par(mar=c(1,5,2,2) + 0.1)
# Initiate Plot
plot(sbys.pcoa$points[ ,1],
  sbys.pcoa$points[ ,2],
     xlab = "PCoA 1",
     ylab = "PCoA 2",
     pch = 16, cex = 2,  type = "n", cex.lab = 1.5,
     cex.axis = 0.5, axes = FALSE)

# # Add Axes
axis(side = 1, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
axis(side = 2, labels = T, lwd.ticks = 2, cex.axis = 1.2, las = 1)
abline(h = 0, v = 0, lty = 3)
box(lwd = 2)

# # Add People
points(sbys.pcoa$points[ ,1],
  sbys.pcoa$points[ ,2],
  pch = 19, cex = 3, bg = "gray", col = "gray")
```

```r
text(sbys.pcoa$points[ ,1], sbys.pcoa$points[ ,2],
     labels = row.names(sbys.pcoa$points))

# Add candies
sbysCAN <- sbys
for(i in 1:nrow(sbys)){
    sbysCAN[i, ] = sbys[i, ] / sum(sbys[i, ])
    }

library(BiodiversityR)
# Plot the candies
sbys.pcoa <- BiodiversityR::add.spec.scores(
  sbys.pcoa,
  sbysCAN,
  method = "pcoa.scores")
```

```
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
## Warning in cor(comm[, i], ordiscores[, j]): l'écart type est nul
```

```r
text(
  sbys.pcoa$cproj[ ,1],
  sbys.pcoa$cproj[ ,2],
  labels = row.names(sbys.pcoa$cproj), col = "red",
  cex=1)
```



Answer 2: The PCoA plots reveal how different jelly bean preference profiles group together based on similarity in abundance and presence-absence patterns. The Bray-Curtis dissimilarity matrix emphasizes relative abundances, meaning common species have a greater influence on distances. The Euclidean distance matrix treats all taxa equally and may be more affected by rare species. **Bray_Curtis**: On the Bray Curtis PCOA plot, students are clustered based on their jelly bean preferences. Students who are close together on the plot have similar jelley bean sampled. Bryan and Jocelyn are very close, suggesting they have highly similar jelly bean samples. Elaine and Madison are positioned in the lower-left quadrant, implying their sampled jelly beans are distinct from others. Caramel Corn, Toasted Marshmallow, and Coconut are positioned near Anna and Ashish, indicating they share these flavors. Blueberry and Berry Blue are closer to myself and Jaeyoung, suggesting they share these flavors. The Euclidean PCoA plot shows a different pattern, with students more evenly distributed across the plot.

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).

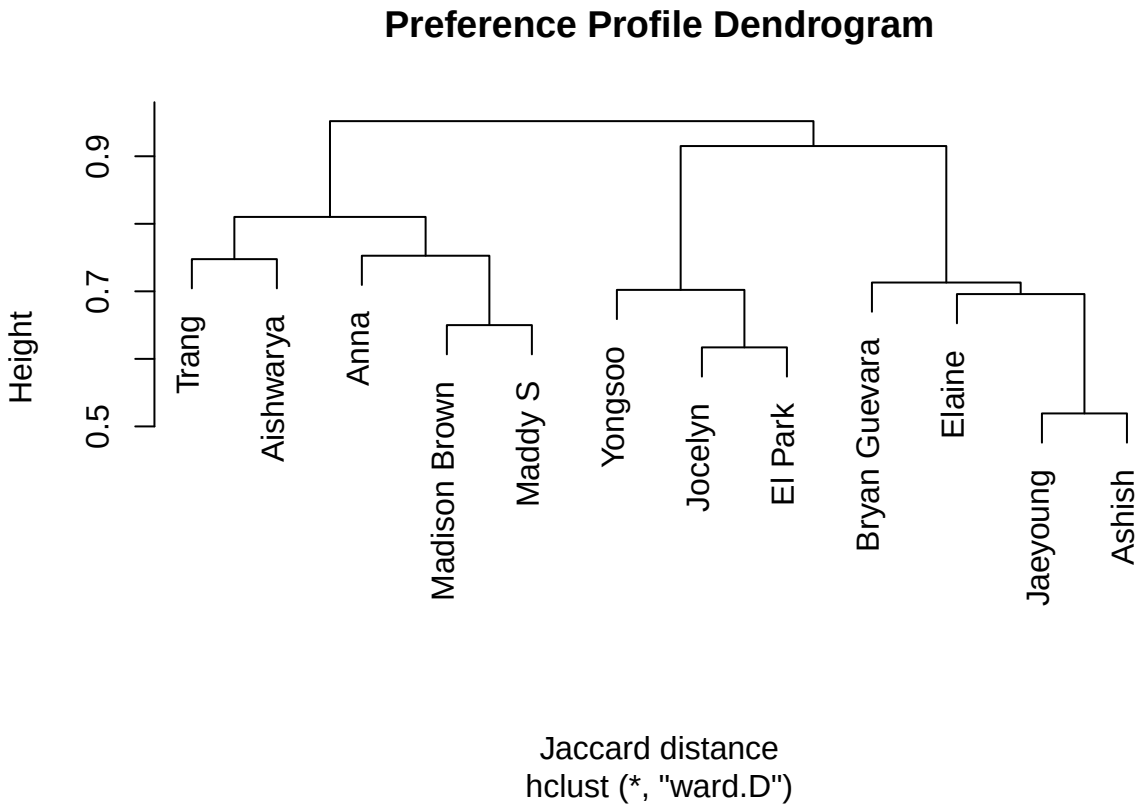
```
# Find the most popular jelly bean
# Exclude the people column
most_popular <- colnames(pref)[which.max(colSums(pref))]

# Print the most popular item
print(most_popular)
```

```
## [1] "Raspberry"
```

Answer 3: The most popular jelly bean is Raspberry.

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.

```
pref.d = vegdist(pref, method = "jaccard")
pref.hc = hclust(pref.d, method = "ward.D")

# Plot the dendrogram
plot(pref.hc, main = "Preference Profile Dendrogram", xlab = "Jaccard distance")
```



Answer 4: Based on the dendogram, my closest preference match is Aishwarya. If I zoom out a bit, Aishwarya and I are also closely related to Anna, forming a subgroup that shares similar preferences.

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)**.