Shotgun Metagenomic

Mahe Alam

2024-06-25

Data Handling, Visualization, and Comparative Analysis

The data on "all_metaphlan_genera.tsv" captures the relative abundance for all genera found in all of nine samples. Now we read the file and clean the data. Firstly we load necessary libraries.

```
library(tidyverse)
library(magrittr)
library(ggpubr)
library(viridis)
library(ggbiplot)
library(ape)
library(ggrepel)
library(readr)
library(dplyr)
library(vegan)
library(gridExtra)
library(patchwork)
#install.packages('tinytex')
#tinytex::install_tinytex()

#Read in the data - Skipping over the header line
input_file = "all_metaphlan_genera.tsv"
raw_data <- read_tsv(input_file, skip = 1)
head(raw_data)
```

```
## # A tibble: 6 x 10
##   clade_name  `03-Cecum_mph` `03-Colon_mph` `03-Rumen_mph` `04-Cecum_mph`
##   <chr>                <dbl>          <dbl>          <dbl>          <dbl>
## 1 g__GGB33322          17.0           16.1           0              0
## 2 g__GGB87498          13.8            4.50          0              0
## 3 g__GGB82392           8.62           7.15          0              0
## 4 g__GGB70891           6.75          10.3           0              0
## 5 g__GGB85338           4.11           5.02          0              0.137
## 6 g__GGB25376           3.54           2.31          0.0392         0.0836
## # i 5 more variables: `04-Colon_mph` <dbl>, `04-Rumen_mph` <dbl>,
## #   `05-Cecum_mph` <dbl>, `05-Colon_mph` <dbl>, `05-Rumen_mph` <dbl>
```

The clade_name column has accession number, need to check if there is any unnecessary value rather then accession number. the clade_name column has "UNCLASSIFIED". so we need to filter out "UNCLASSIFIED" row from clade_name column.

```
#The filter function is used to subset rows in the dataframe. In this case, it's being used to #keep on
filtered_data <- raw_data %>% filter(clade_name != "UNCLASSIFIED" )
#form a true community matrix. This command takes the filtered_data dataframe and sets the #clade_name
filtered_data <- filtered_data %>% column_to_rownames('clade_name')
#Transpose the data so that accession number are across top.
community_matrix<- as.data.frame(t(filtered_data))
#remove the '__g' from rownames so that we can get the proper accession number and use for NCBI #search
colnames(community_matrix) <- gsub('g__', '', colnames(community_matrix))
#clean up sample names for plotting as well
rownames(community_matrix) <- gsub('_mph', '', rownames(community_matrix))
#str(community_matrix)
```

Now we need to form a data object containing metadata. ordinarily these would be curated beforehand, but we can develop a rudimental one based on our sample names.

```
#define metadata
metadata <- data.frame(sampleID = rownames(community_matrix))
#form sampling site based on sample name
metadata <- metadata %>% mutate(anatomical_site = case_when(
  grepl('Colon', sampleID) ~ 'Colon',
  grepl('Cecum', sampleID) ~ 'Cecum',
  grepl('Rumen', sampleID) ~ 'Rumen'
))
# make the sample ID a column
stacked_data <- community_matrix %>% rownames_to_column('sampleID')
# make the data long in format
stacked_data <- stacked_data %>% pivot_longer(!sampleID,
                                              names_to = 'genus',
                                              values_to = 'value')
# Merge in metadata based on the sampleID
stacked_data <- merge(stacked_data, metadata, by='sampleID')
#sum stacked data genera and set 'threshold' for 'other' genera
other_genera <- stacked_data %>%
  group_by(genus) %>%
  dplyr::summarise(total_abundance = sum(value)) %>%
  filter(total_abundance < 6)
#now rename genera in steak data with 'other'
stacked_data <- stacked_data %>% mutate(genus = case_when(
  genus %in% other_genera$genus ~ 'other',
  TRUE ~ genus
))


#unique(data$genus): This function returns the unique values from the genus column, ensuring #that each
stacked_data$genus <- factor(stacked_data$genus, levels = unique(stacked_data$genus))
#write the tsv file with accession number
output_file = 'data_with_accession.tsv'
write_tsv(stacked_data, output_file)
```

```
# Set global chunk options
#knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE)

#loading library
```

```r
library(rentrez)
```

```
## Warning: package 'rentrez' was built under R version 4.2.3
```

```r
# Define the function to fetch the genus name
fetch_genus_name <- function(accession_number) {
  email <- "mahealam2007@gmail.com"  # Set your email address
  #print(paste("Fetching record for accession number:", accession_number))

  # Try to fetch the record from NCBI
  record <- tryCatch({
    entrez_fetch(db="protein", id=accession_number, rettype="gb", retmode="text", email=email)
  }, error = function(e) {
    #print(paste("Error fetching record:", e$message))
    return(NA)
  })

  # Check if the record was successfully fetched
  if (!is.na(record)) {
    #print("Record successfully fetched. Parsing the record...")
    # Split the record into lines
    lines <- strsplit(record, "\n")[[1]]
    # Find the line containing "ORGANISM"
    genus_line <- grep("ORGANISM", lines, value=TRUE)
    #print(paste("Genus line found:", genus_line))
    # If the "ORGANISM" line is found, extract the genus name
    if (length(genus_line) > 0) {
      # Split the genus line and extract the second element
      genus <- strsplit(genus_line, " +")[[1]][3]
      return(genus)
    } else {
      #print("Organism information not found in record.")
      return(NA)
    }
  } else {
    return(NA)
  }
}

# Read the data from a file
input_file <- "data_with_accession.tsv"  # Replace with your input file
output_file <- "data_with_genus_names.tsv"  # Output file to save the results

#Read the data file (assuming CSV format, adjust if necessary)
data_with_accession <- read_tsv(input_file, show_col_types = FALSE)

# Check the structure of the data
head(data_with_accession)
```

```
## # A tibble: 6 x 4
##   sampleID genus    value anatomical_site
##   <chr>    <chr>    <dbl> <chr>
## 1 03-Cecum GGB33322 17.0  Cecum
```

```
## 2 03-Cecum GGB87498 13.8  Cecum
## 3 03-Cecum GGB82392  8.62 Cecum
## 4 03-Cecum GGB70891  6.75 Cecum
## 5 03-Cecum GGB85338  4.11 Cecum
## 6 03-Cecum GGB25376  3.54 Cecum
```

```r
# Process each entry in the `genus` column
for (i in 1:nrow(data_with_accession)) {
  accession_number <- data_with_accession$genus[i]
  # Check if the entry is an accession number (assuming accession numbers start with "GGB")
  if (grepl("^GGB", accession_number)) {
    genus_name <- fetch_genus_name(accession_number)
    # If a genus name was found, replace the accession number
    if (!is.na(genus_name)) {
      data_with_accession$genus[i] <- genus_name
    }
  }
}

# Write the updated data to a new tsv file
head(data_with_accession)
```

```
## # A tibble: 6 x 4
##   sampleID genus            value anatomical_site
##   <chr>    <chr>            <dbl> <chr>
## 1 03-Cecum Sphingomonas     17.0  Cecum
## 2 03-Cecum Staphylococcus   13.8  Cecum
## 3 03-Cecum Deinococcus       8.62 Cecum
## 4 03-Cecum Shewanella        6.75 Cecum
## 5 03-Cecum Marinobacterium   4.11 Cecum
## 6 03-Cecum Mucilaginibacter  3.54 Cecum
```

```r
write_tsv(data_with_accession, output_file)
```

Including bar plot Plots

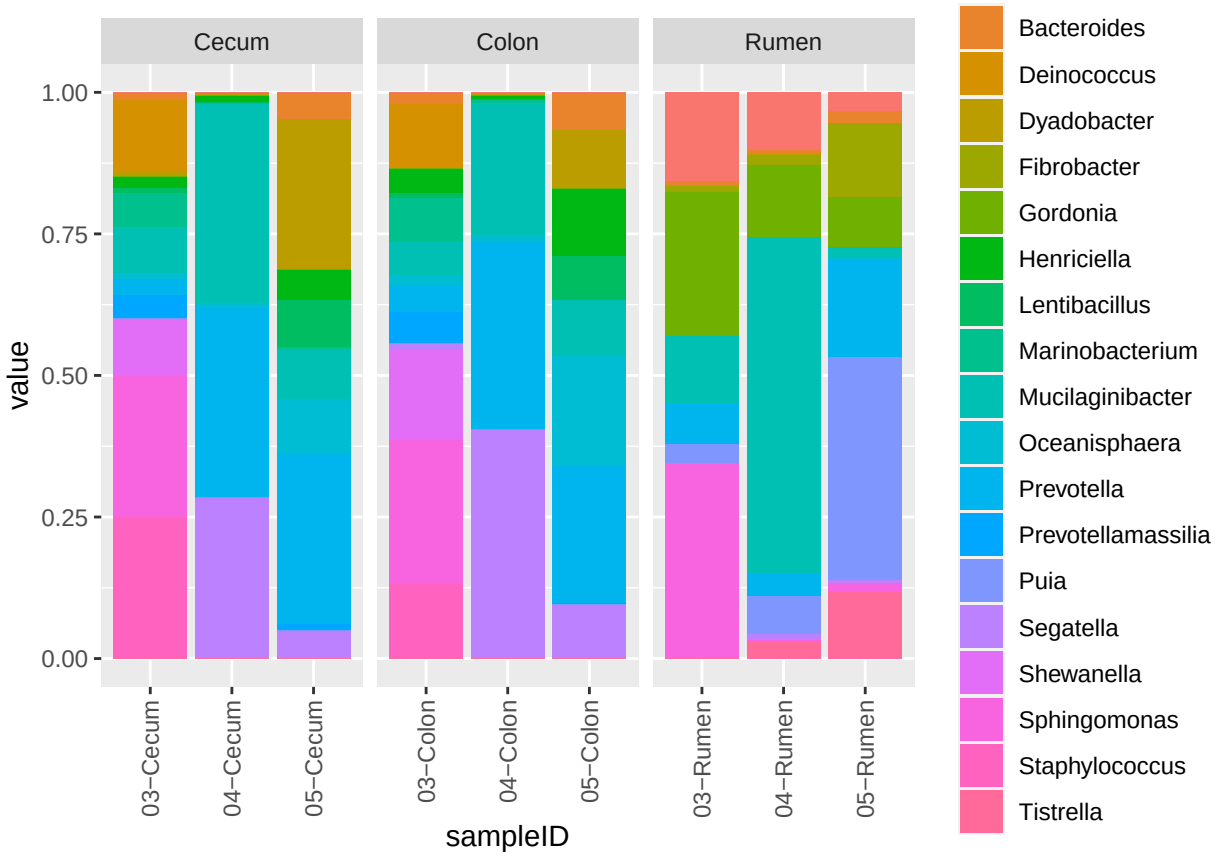


Figure is a stacked bar chart demonstrating relative abundance of bacterial genera reported by MetaPhlAn run cattle cecum (right), Rumen (middle) and colon(left) samples.

The samples appear dominated by prevotella, with some other genera such as Bacteroides, lentibacillus, oceanispaera, Prevotellamassilia, staphylococcus and Tistrella. The composition of rumen samples appears different to cecum and colon samples, which contain a higher relative sbundance of prevotella.

```
#community_matrix have constant values (i.e., all values in the column are the same)
#or are composed entirely of zeros. PCA cannot properly scale these columns because
#the variance is zero, making it impossible to rescale them to unit variance.
####################################
# Step 2: Perform PCA on the filtered community matrix
# Step 1: Perform PCA
pca_result <- prcomp(community_matrix, scale. = TRUE)

# Calculate percentage of variance explained by each PC
variance_explained <- round((pca_result$sdev^2 / sum(pca_result$sdev^2)) * 100, 2)

# Generate PCA plot
pca_plot <- ggbiplot(pca_result,
                     labels = rownames(community_matrix),
                     groups = metadata$anatomical_site,
                     var.axes = FALSE) +
  labs(x = paste0("PC1 (", variance_explained[1], "%)"),
       y = paste0("PC2 (", variance_explained[2], "%)"),
```

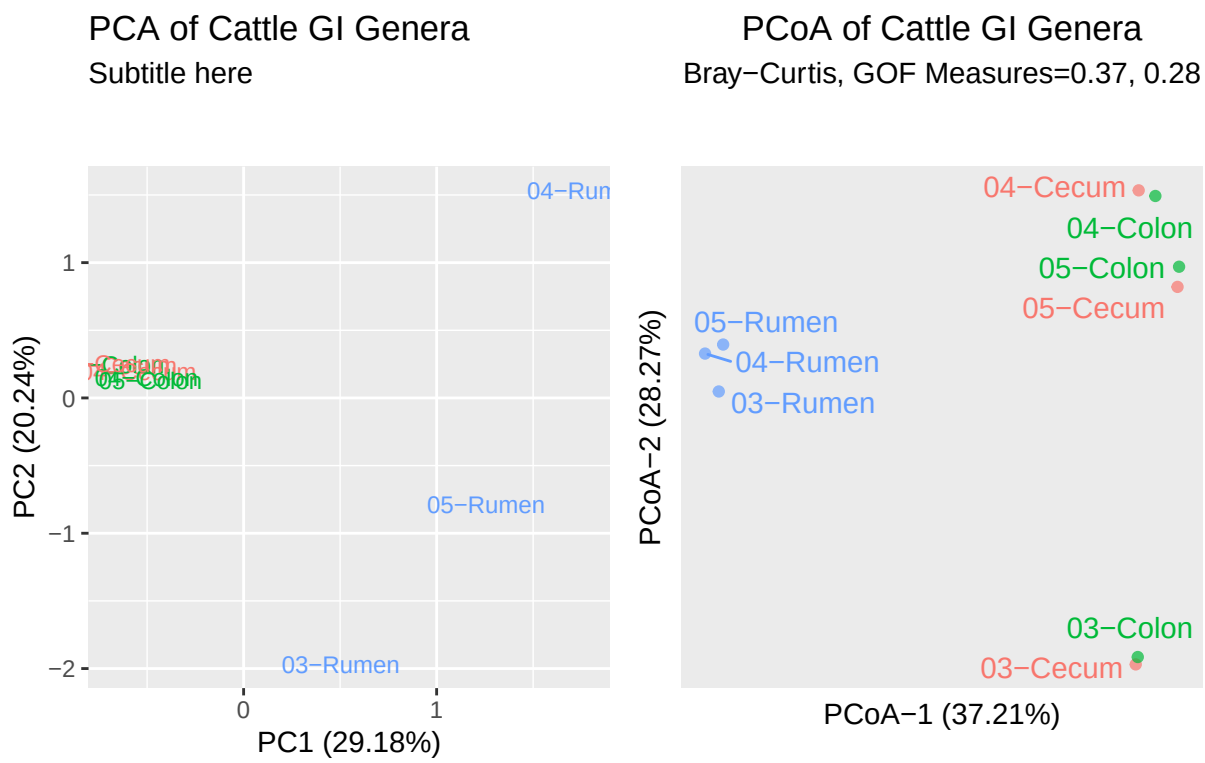
```
       title = "PCA of Cattle GI Genera",
       subtitle = "Subtitle here") +  # Adding a subtitle example
  theme(legend.position = 'none',
        aspect.ratio = 1)


# Compute the Bray-Curtis dissimilarity matrix
bray_dist <- vegdist(community_matrix, method = 'bray')

# Perform PCoA
pcoa_data <- cmdscale(bray_dist, k = 2, eig = TRUE)
pcoa_points <- as.data.frame(pcoa_data$points)
pcoa_points$anatomical_site <- metadata$anatomical_site
# Calculate the proportion of variance explained by each axis (GOF)
eig_values <- pcoa_data$eig
variance_explained <- eig_values / sum(eig_values)
GOF <- variance_explained[1:2]  # First two axes

# Create the PCoA plot
bray_plot <- ggplot(pcoa_points, aes(x = V1, y = V2, col = anatomical_site, label = rownames(pcoa_points
  geom_point(alpha = 0.7) +
  geom_text_repel(aes(label = rownames(pcoa_points)), vjust = 1) +
  labs(
    x = paste0('PCoA-1 (', round(GOF[1] * 100, 2), '%)'),
    y = paste0('PCoA-2 (', round(GOF[2] * 100, 2), '%)'),
    title = 'PCoA of Cattle GI Genera',
    subtitle = paste0('Bray-Curtis, GOF Measures=', paste(round(GOF, 2), collapse = ', '))
  ) +
  theme(
    legend.position = 'none',
    axis.text = element_blank(),
    axis.ticks = element_blank(),
    panel.grid = element_blank(),
    plot.title = element_text(hjust = 0.5),
    plot.subtitle = element_text(hjust = 0.5),
    aspect.ratio = 1
  )

# Print the plot
#print(bray_plot)
#Arrange the plots side by side on the same page
#grid.arrange(pca_plot, bray_plot, ncol = 2)
# Arrange the plots side by side and full page using patchwork
# Combine the plots using patchwork
combined_plot <- (pca_plot | bray_plot) + plot_layout(ncol = 2, widths = 30, heights = 8)
# Print the combined plot
print(combined_plot)
```



PCA(left) and PCoA(Bray-Curtis distance) plots of a selection of cattle gastro-intestinal (GI) samples from the colon, cecum and rumen analysed by MetaPhlAn for bacterial content.

This PCA plot shows separation between rumen samples compared to the rest of the dataset and a relatively tight clustering of the colon and cecum samples. From this, it could be inferred that the community of bacteria is closer between colon and cecum samples than it is in rumen samples, as was suspected from the stacked bar chart. The PCoA plot also shows that colon and cecum samples are separated from the rumen samples, but the tight clustering of colon and rcecum samples are not seen. In the PCoA, Plot samples from the same animal are found closer than between sample type: the colon and cecum samples from subjects 3, 4, and 5 are all group together. The differences observed are probably caused by PCA using the Euclidean distance, whereas the PCoA plot uses Bray-Curtis. These types of dimensionality reduction plots can be used to reveal outlier samples that separate away from the rest of the dataset. In this case the rumen samples shows large separation, but this appears to be due to the variability in rumen samples as opposed to one single outlier. So, In this case, there are no obvious outlier samples.