Shotgun Metagenomic

Mahe Alam

2024-05-28

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_R1_mph` `03-Colon_R1_mph` `03-Rumen_R1_mph`
##   <chr>                   <dbl>             <dbl>             <dbl>
## 1 g__GGB33322             17.6              0                 0
## 2 g__GGB87498             13.9              0                 0
## 3 g__GGB82392              8.85             0                 0
## 4 g__GGB70891              6.89             0                 0
## 5 g__GGB85338              4.18             0.219             0
## 6 g__GGB25376              3.59             0.00638           0.0126
## # i 6 more variables: `04-Cecum_R1_mph` <dbl>, `04-Colon_R1_mph` <dbl>,
## #   `04-Rumen_R1_mph` <dbl>, `05-Cecum_R1_mph` <dbl>, `05-Colon_R1_mph` <dbl>,
## #   `05-Rumen_R1_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.

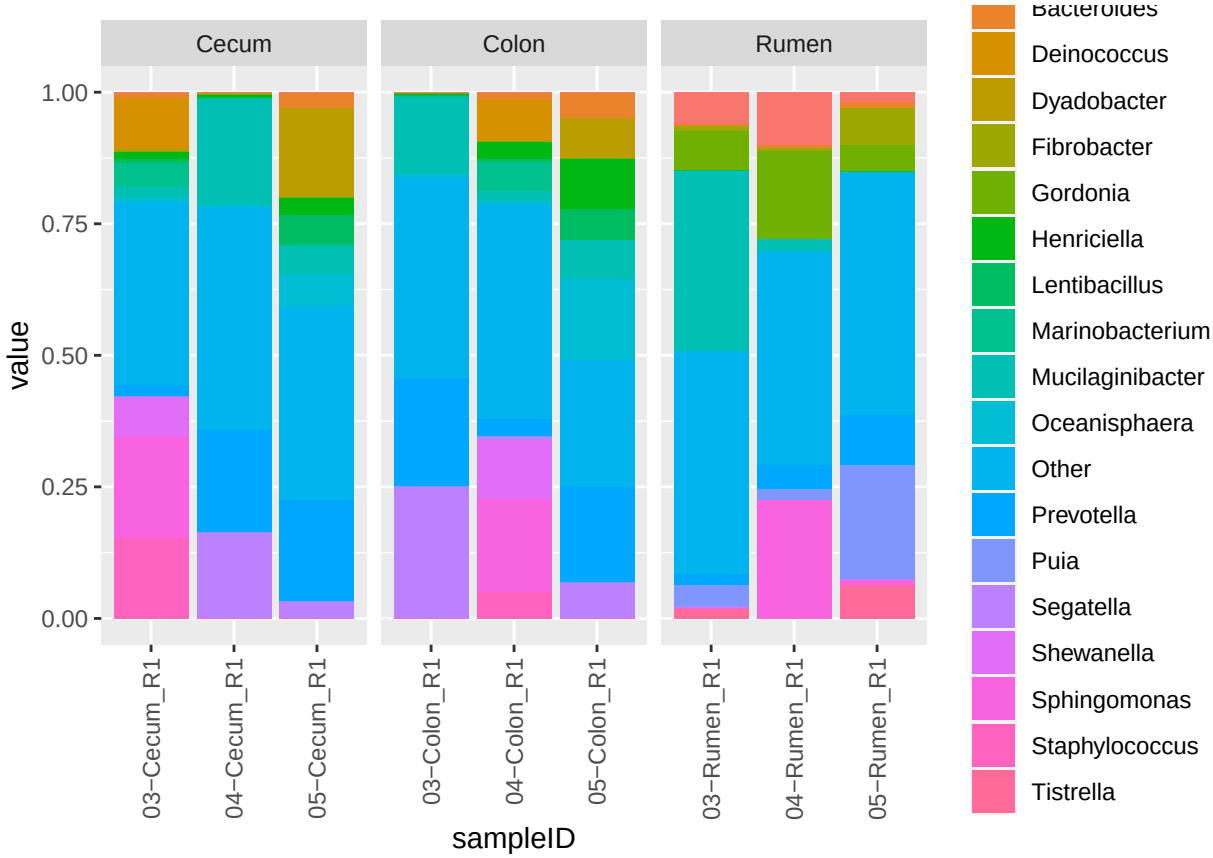
```
#The filter function is used to subset rows in the dataframe. In this case, it's being used to #keep on
filtered_data <- raw_data %>% filter(clade_name != "UNCLASSIFIED" )
#form a true community matrix. This command takes the filtered_data dataframe and sets the #clade_name
filtered_data <- filtered_data %>% 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)
```

Including bar plot Plots



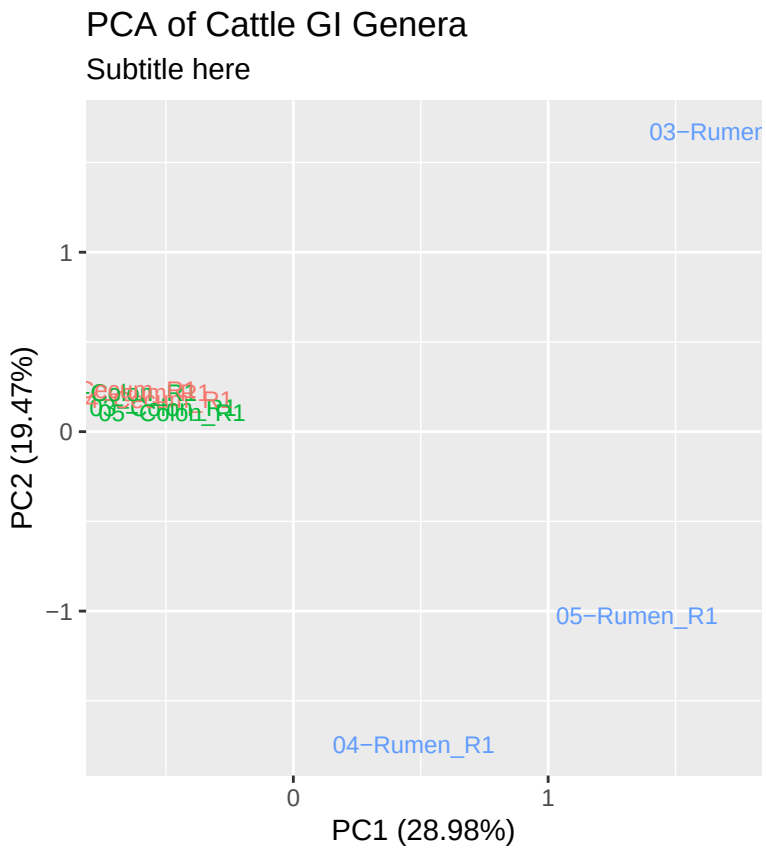

```
#community_matrix have constant values (i.e., all values in the column are the same)
#or are composed entirely of zeros. PCA cannot properly scale these columns because
#the variance is zero, making it impossible to rescale them to unit variance.
####################################
# Step 2: Perform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

# Print the PCA plot
```

```
print(pca_plot)
```




```
#print(pca_plot)

# 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)
```