HEB1410 Gut Microbiome and Human Health

Computation Lab Section

Yijia Liow

2023-11-28

Outline

- Installing necessary packages
- Linear mixed-effects models
- Differential abundance analysis: `MaAsLin2`
- Heatmap construction

Install packages

```
# lme4: linear mixed-effects models
install.packages("lme4")
library(lme4)

# MaAsLin2: multivariate association with linear models
BiocManager::install("Maaslin2")
library(MaAsLin2)

# pheatmap: heatmap construction
install.packages("pheatmap")
library(pheatmap)
```

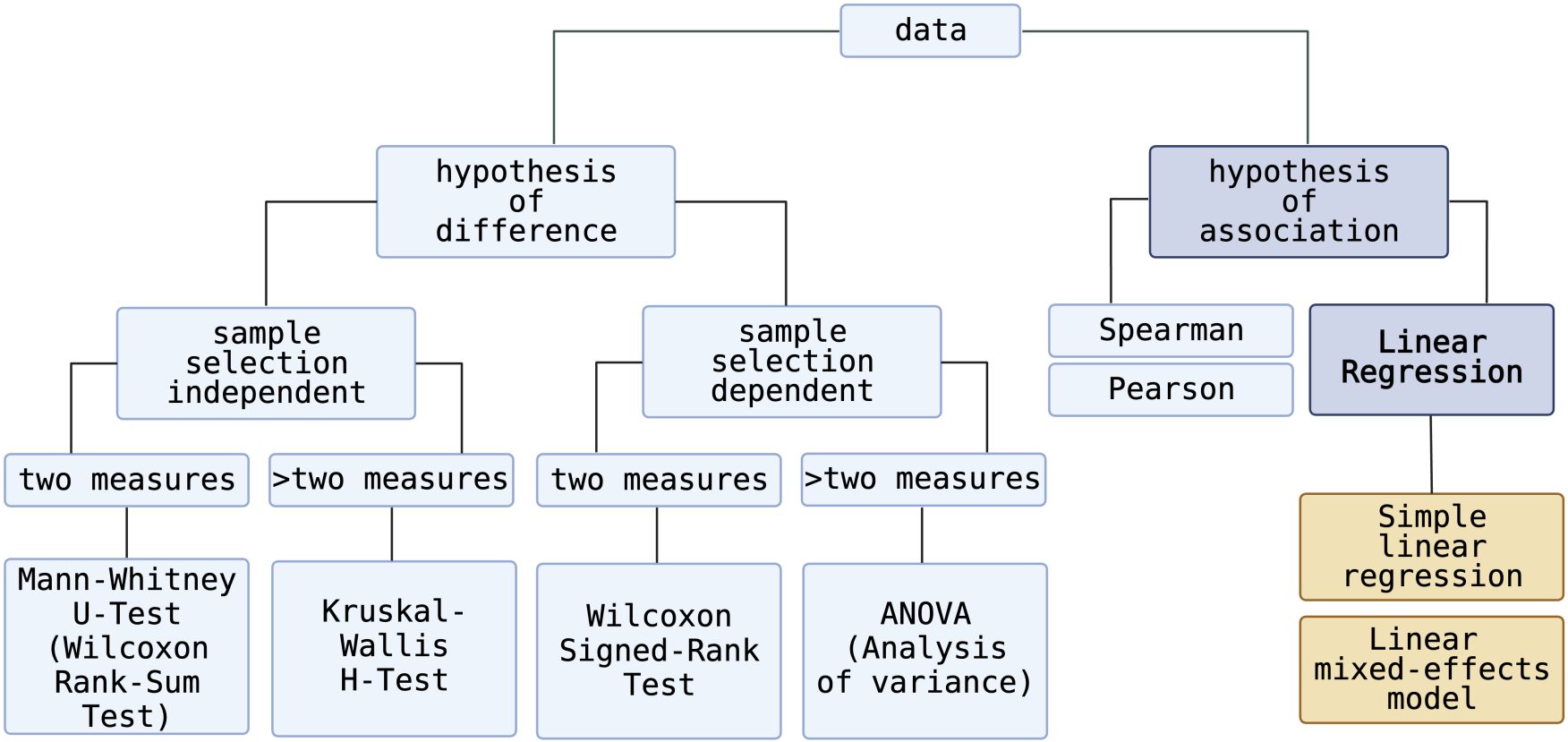
Load data

```r
# phenotypic data
preference <- read.csv("data/preference.csv", header = TRUE, sep = ",")
intervention <- read.csv("data/intervention.csv", header = TRUE, sep = ",")
echomri <- read.csv("data/echomri.csv", header = TRUE, sep = ",")
```

Data wrangling

```r
# data wrangling for downstream analyses
intervention <- intervention %>%
  mutate(timepoint = factor(timepoint,
                            levels = c("d2", "d4", "d6",
                                       "d8", "d10", "d12", "d14"))) %>%
  arrange(mouse_id, timepoint) %>%
  group_by(mouse_id) %>%
  mutate(
    cumulative_intake = cumsum(feed_intake),
    baseline_body_weight = body_weight[which(timepoint == "d2")],
    percent_bw_change =
    (body_weight - baseline_body_weight) /
    baseline_body_weight * 100
  ) %>%
  ungroup()
```

Hypothesis of association



Simple linear regression

```
# basic syntax
lm(response ~ predictor, data = dataset)
```

- summarize and study the relationship between two variables
- `response` is the outcome variable we are trying to predict/explain
- `predictor` is the variable we are using to predict/explain the outcome

Simple linear regression

The model for simple linear regression can be expressed as:

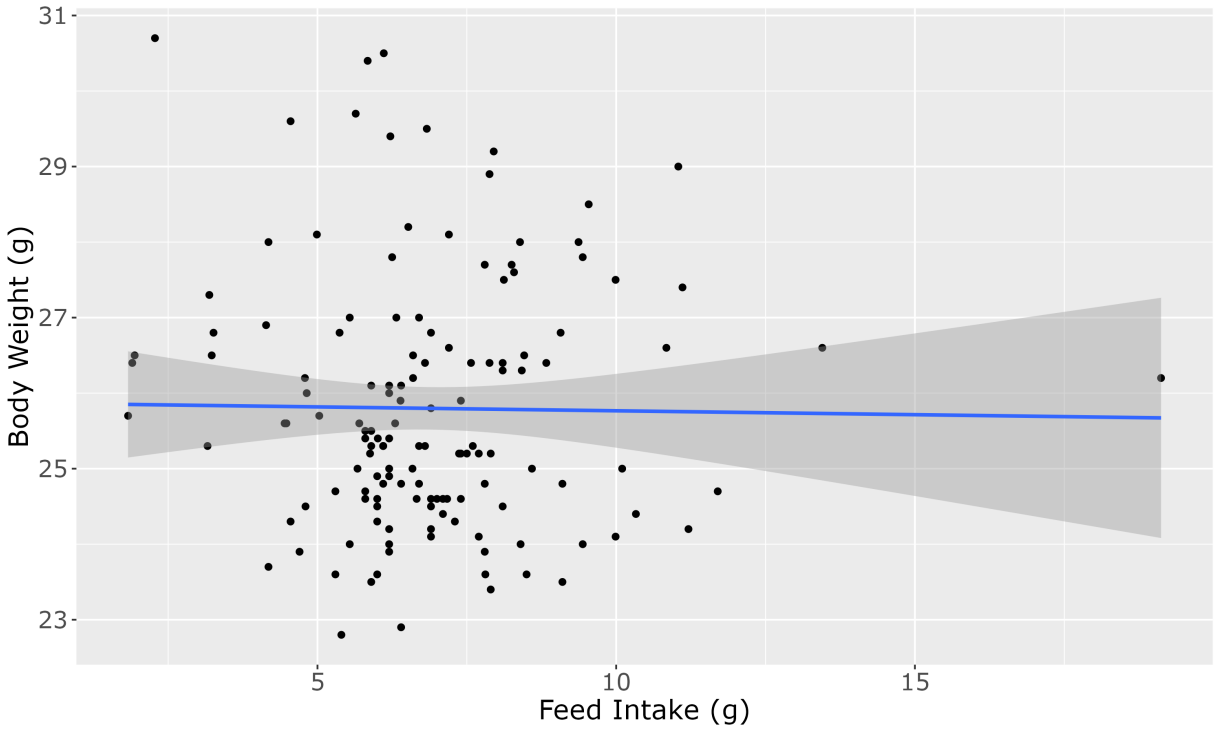
$$y = \beta_0 + \beta_1 x + \epsilon$$

where:

- y: response variable we are trying to predict/explain
- x: predictor variable we are using to predict/explain the outcome
- β_0: intercept of the line (the value of y when x is 0)
- β_1: slope of the line (the change in y for a one-unit change in x)
- ϵ: error term (the variability in y not explained by x)

Simple linear regression

```
# basic syntax
lm(response ~ predictor, data = dataset)

# example: modeling body weight as a function of feed intake
lm(body_weight ~ feed_intake, data = intervention)
```



Linear mixed-effects (LME) models

LME models account for both fixed and random effects in your data:

$$y = X\beta + Z\gamma + \epsilon$$

where:

- y is the response variable
- X is the matrix of fixed-effects predictors
- β is the vector of fixed-effects coefficients
- Z is the matrix of random-effects predictors
- γ is the vector of random-effects coefficients
- ϵ is the vector of residuals or errors

Linear mixed-effects (LME) models

```
# Basic syntax
lmer(response ~ fixed_effects + (random_effects | group), data = dataset)

# Model with treatment
fit_lm1 <- lmer(body_weight ~ feed_intake + treatment + (1|mouse_id),
                data = intervention)
summary(fit_lm1)

# Model without treatment
fit_lm2 <- lmer(body_weight ~ feed_intake + (1|mouse_id),
                data = intervention)
summary(fit_lm2)
```

LME output interpretation

```
Random effects:
 Groups   Name        Variance Std.Dev.
 mouse_id (Intercept) 1.6462   1.2830
 Residual             0.6896   0.8304
Number of obs: 139, groups:  mouse_id, 20

Fixed effects:
                  Estimate Std. Error t value
(Intercept)      25.075376   0.486702  51.521
feed_intake      -0.006397   0.036364  -0.176
treatmentwestern  1.542105   0.590847   2.610

Correlation of Fixed Effects:
            (Intr) fd_ntk
feed_intake -0.513
trtmntwstrn -0.606 -0.002
```

Differential Abundance Analysis: `MaAsLin2`

- `MaAsLin2`: Multivariate Association with Linear Models
- identify associations between host metadata and microbial abundance
- analysis of multiple predictors and adjust for confounding factors
- find more information about `MaAsLin2` here

- Step 1: Installing the `MaAsLin2` package
- Step 2: Prepare the input data for `MaAsLin2`
- Step 3: `MaAsLin2` model fitting
- Step 4: Visualizing the `MaAsLin2` model output

Step 1: MaAsLin2 installation

```r
# Install Bioconductor if not already
if(!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

# Install MaAsLin2
BiocManager::install("Maaslin2")

# Load the package
library(Maaslin2)
```

Step 2: MaAsLin2 data preparation

```
# MaAsLin2 model basic syntax
result <- Maaslin2(
  input_data = input_data,
  input_metadata = input_metadata,
  output = "path/to/output_directory",
  fixed_effects = c("YourFixedEffect1", "YourFixedEffect2")
)
```

- input_data: ASVs as rows and samples as columns
- input_metadata: samples as rows and metadata as columns
- output: path to the output directory
- fixed_effects: variables we use to predict the outcome

MaAsLin2 data preparation

```
# Load gut microbiome data
ps_course <- readRDS("data/ps_course.rds")

# Convert phyloseq object to data frames
sample_df <- data.frame(sample_data(ps_course))
otu_df <- data.frame(otu_table(ps_course))
taxonomy_df <- data.frame(tax_table(ps_course))
```

MaAsLin2 data preparation

```
# post-intervention data preparation
input_metadata_post <- sample_df %>%
  filter(timepoint == "D14") %>%
  select(timepoint, treatment, group)

input_data_post <- otu_df %>%
  rownames_to_column("ASV") %>%
  pivot_longer(-ASV, names_to = "SampleID", values_to = "value") %>%
  mutate(SampleID = str_replace(SampleID, "\\.", "-")) %>%
  filter(SampleID %in% rownames(input_metadata_post)) %>%
  pivot_wider(names_from = ASV, values_from = value) %>%
  column_to_rownames("SampleID")
```

Step 3: Running the MaAsLin2 model

```
# Install and load the fs package for file system operations
install.packages("fs")
library(fs)

fs::dir_create("output/post")

result_post_raw <- Maaslin2::Maaslin2(
  input_data = input_data_post,
  input_metadata = input_metadata_post,
  fixed_effects = c("treatment"),
  output = "output/post"
)
```

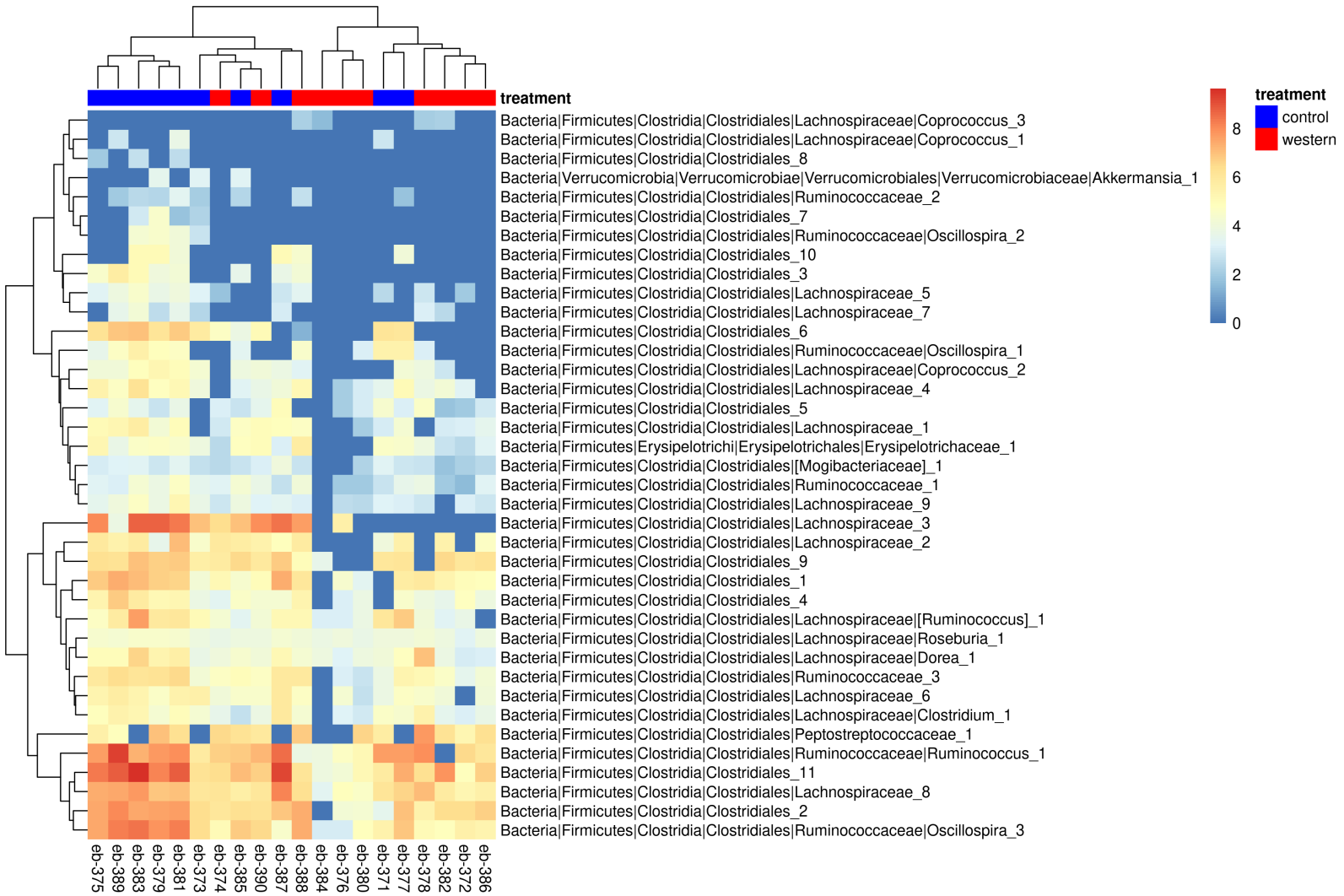
Step 4: MaAsLin2 output visualization

```r
# Join with taxonomy information
result_post <- result_post_raw$results %>%
  left_join(taxonomy_df, by = c("feature" = "ASV"))

# Filter and rank for top 5 increasing and decreasing
top_taxa <- result_post %>%
  filter(qval < 0.25) %>%
  mutate(rk = rank(coef),
         irk = rank(desc(coef))) %>%
  filter(rk <= 5 | irk <= 5) %>%
  mutate(
    taxa = paste(Kingdom, Phylum, Class, Order, Family, Genus, sep = "|"),
    taxa = str_replace_all(taxa, "\\|NA", ""),
    taxa = if_else(taxa == "NA", "Unclassified", taxa),
  )
```

MaAsLin2 output visualization

```
# Create a volcano plot
volcano_plot <- ggplot(result_post, aes(x = coef, y = -log10(qval))) +
  geom_point() +
  theme_minimal() +
  labs(x = "Effect Size", y = "-log10(q-value)") +
  ggrepel::geom_text_repel(
    data = top_taxa,
    aes(x = coef, y = -log10(qval), label = taxa))

# Display the plot
print(volcano_plot)
```

Heatmap construction



Heatmap construction

```
# Joining with taxonomy information
result_post <- result_post_raw$results %>%
  left_join(taxonomy_df, by = c("feature" = "ASV"))

# Extracting the significant taxa
sig_taxa <- result_post %>%
  filter(qval < 0.25)

input_data_taxa <- input_data_post %>%
  rownames_to_column("SampleID") %>%
  pivot_longer(-SampleID, names_to = "ASV", values_to = "value") %>%
  filter(ASV %in% sig_taxa$feature) %>%
  left_join(taxonomy_df, by = c("ASV" = "ASV")) %>%
  mutate(
    taxa = paste(Kingdom, Phylum, Class, Order, Family, Genus, sep = "|"),
    taxa = str_replace_all(taxa, "\\|NA", ""),
    taxa = if_else(taxa == "NA", "Unclassified", taxa),
  )
```

Heatmap construction

```
# Creating a unique taxa name for each ASV
taxa_names <- input_data_taxa %>%
  distinct(ASV, taxa) %>%
  group_by(taxa) %>%
  mutate(
    count = row_number(),
    count_max = n(),
    taxa = paste0(taxa, "_", count)
  ) %>%
  select(ASV, taxa)
```

Heatmap construction

```
# Creating a matrix
data_matrix <- input_data_taxa %>%
  select(SampleID, ASV, value) %>%
  left_join(taxa_names, by = "ASV") %>%
  select(-ASV) %>%
  pivot_wider(names_from = SampleID,
              values_from = value,
              values_fill = 0) %>%
  column_to_rownames(var = "taxa") %>%
  as.matrix()

# Log transformation
log_transformed_data <- log(data_matrix + 1)
```

Heatmap construction

```
# Metadata
annotation_data <- input_metadata_post %>%
  select(treatment) %>%
  rownames_to_column("SampleID") %>%
  distinct()

# Match the order of the samples
annotation_data <- annotation_data[
  match(colnames(log_transformed_data), annotation_data$SampleID), ] %>%
  column_to_rownames(var = "SampleID")

annotation_colors <- list(
  treatment = setNames(c("blue", "red"),
  unique(annotation_data$treatment))
  )
```

Heatmap construction

```
# Open a PNG device
png("heatmap.png", width = 8, height = 10, units = "in", res = 300)

# Create the heatmap
pheatmap(
  log_transformed_data, scale = "none",
  annotation_col = annotation_data, annotation_colors = annotation_colors
)

# Close the device to save the file
dev.off()
```