Lefser Error Report

YANG Chen

2023-03-05

Data Import

I attached the data I used to the email

```
library(readr)
metadata <-
  read_delim(
    "~/Microbiome/C9orf72/Code And Data/new_metadata.txt",
    delim = "\t",
    escape_double = FALSE,
    trim_ws = TRUE
  )
```

```
## Rows: 50 Columns: 31
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (27): sample_name, Enviroment, Group, Assay Type, bacterial_metagenome_...
## dbl   (3): AvgSpotLen, Bases, Bytes
## dttm  (1): ReleaseDate
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
abundance <- read_delim("~/Microbiome/C9orf72/Code And Data/picrust2_out/KO_metagenome_out/pred_metagen
    delim = "\t", escape_double = FALSE,
    trim_ws = TRUE)
```

```
## Rows: 4952 Columns: 51
## -- Column specification --------------------------------------------------------
## Delimiter: "\t"
## chr  (1): #NAME
## dbl (50): SRR11393747, SRR11393768, SRR11393775, SRR11393761, SRR11393755, S...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
abundance <- column_to_rownames(abundance, var = "#NAME")
abundance <- as.data.frame(abundance)
group <- "Group"
   if (!is_tibble(metadata)) {
     metadata <- tibble::as_tibble(metadata)
    }
    sample_names <- colnames(abundance)
    matches <-
      base::lapply(metadata, function(x) {
```

```
      intersect(sample_names, x)
    })
  matching_columns <-
    names(metadata)[sapply(matches, function(x) {
      length(x) == length(sample_names)
    })]
  sample_names <- colnames(abundance)
  abundance_mat <- as.matrix(abundance)
  metadata_order <-
    match(sample_names, as.matrix(metadata[, matching_columns]))
  metadata <- metadata[metadata_order,]
  metadata_mat <- as.matrix(metadata)
  metadata_df <- as.data.frame(metadata)
  Group <- factor(metadata_mat[, group])
  Level <- levels(Group)
  length_Level <- length(Level)
```

Error Report

It will be a error "Error in svd(X, nu = 0L) : a dimension is zero" because between "Broad Institute" and "Harvard BRI" there are no significant biomarker whcih $p < 0.05$ under kw test, but the error information doesn't reveal it correctly.

```
Lefser_combinations <- utils::combn(Level, 2)
        Lefser_results <- list()
        Lefser_metadata_df <- metadata_df
        i <- 7
          Lefser_sub_metadata_df <-
            Lefser_metadata_df[Lefser_metadata_df[, group] %in% Lefser_combinations[, i],]
          Lefser_sub_abundance <-
            abundance[,Lefser_metadata_df[, group] %in% Lefser_combinations[, i] ]
          Lefser_sub_abundance <- Lefser_sub_abundance[!rownames(Lefser_sub_abundance) %in% c("K06338",
          colnames(Lefser_sub_metadata_df)[colnames(Lefser_sub_metadata_df) == group] <-
            "Group_group_nonsense_"
          Lefser_sub_metadata_df$Group_group_nonsense_ <-
            factor(Lefser_sub_metadata_df$Group_group_nonsense_)
          Lefser_object <-
            SummarizedExperiment(assays = list(counts = as.matrix(Lefser_sub_abundance)),
                                 colData = Lefser_sub_metadata_df)
          Lefser_kw_filter <-
            apply(Lefser_object@assays@data$counts, 1L, function(x) {
              kruskal.test(x ~ as.numeric(Lefser_sub_metadata_df[, "Group_group_nonsense_"]) -
                             1)[["p.value"]]
            })
          print(sum(na.omit(as.numeric(Lefser_kw_filter < 0.05))))
           Lefser_results <-
             cbind(
               feature = lefser(Lefser_object, groupCol = "Group_group_nonsense_")$Names,
               method = "Lefser",
               group1 = Lefser_combinations[, 1][1],
               group2 = Lefser_combinations[, 1][2],
               effect_scores = lefser(Lefser_object, groupCol = "Group_group_nonsense_")$scores
             )
```

Revised

I used the following in my own package ggpicrust2 to avoid the error. In the current lefser code, if the Level in the Group exceeds 2, the code will not execute, so this code not only solves the "Error in svd(X, nu = 0L) : a dimension is zero" problem, but also implements a two-by-two comparison between all levels

```
Lefser_combinations <- utils::combn(Level, 2)
        Lefser_results <- list()
        Lefser_metadata_df <- metadata_df
        for (i in seq_len(ncol(Lefser_combinations))) {
          Lefser_sub_metadata_df <-
            Lefser_metadata_df[Lefser_metadata_df[, group] %in% Lefser_combinations[, i],]
          Lefser_sub_abundance <-
            abundance[,Lefser_metadata_df[, group] %in% Lefser_combinations[, i] ]
          colnames(Lefser_sub_metadata_df)[colnames(Lefser_sub_metadata_df) == group] <-
            "Group_group_nonsense_"
          Lefser_sub_metadata_df$Group_group_nonsense_ <-
            factor(Lefser_sub_metadata_df$Group_group_nonsense_)
          Lefser_object <-
            SummarizedExperiment(assays = list(counts = as.matrix(Lefser_sub_abundance)),
                                 colData = Lefser_sub_metadata_df)
          Lefser_kw_filter <-
            apply(Lefser_object@assays@data$counts, 1L, function(x) {
              kruskal.test(x ~ as.numeric(Lefser_sub_metadata_df[, "Group_group_nonsense_"]) -
                             1)[["p.value"]]
            })
          if (!sum(na.omit(as.numeric(Lefser_kw_filter < 0.05)))) {
            next
          }
          Lefser_results[[i]] <-
            cbind(
              feature = lefser(Lefser_object, groupCol = "Group_group_nonsense_")$Names,
              method = "Lefser",
              group1 = Lefser_combinations[, 1][1],
              group2 = Lefser_combinations[, 1][2],
              effect_scores = lefser(Lefser_object, groupCol = "Group_group_nonsense_")$scores
            )
        }
```