An Introduction to zitools

15 April 2025

Contents

1 Introduction

This vignette provides an introductory example on how to work with the 'zitools' package, which implements a weighting strategy based on a fitted zero inflated mixture model. 'zitools' allows for zero inflated count data analysis by either using down-weighting of excess zeros or by replacing an appropriate proportion of excess zeros with NA. Through overloading frequently used statistical functions (such as mean, median, standard deviation), plotting functions (such as boxplots or heatmap) or differential abundance tests, it allows a wide range of downstream analyses for zero-inflated data in a less biased manner.

2 Installation

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("zitools")
```

Let's start with loading the 'zitools'-package:

```
library(zitools)
```

and loading additionally required packages.

```
library(phyloseq)
library(DESeq2)
library(tidyverse)
library(microbiome)
```

2.1 Example Dataset

An example microbiome dataset from the R package microbiome is used to display the 'zitools' workflow. The data used here are described in [Lahti et. al.] {https://pubmed.ncbi.nlm.nih.gov/23638368/}.

The study, in which this dataset was generated, was conducted to investigate the impact of probiotic intervention on the human intestinal microbiome. Therefore, the intestinal microbiota diversity was analysed by performing 16S rRNA sequencing obtained from fecal samples. The datasetcomprises 22 subjects - 8 from the probiotic group and 14 from the placebo group, all samples were analysed before and after the intervention (44 samples in total).

In a first step, the dataset is loaded.

```
data("peerj32")
phyloseq <- peerj32[["phyloseq"]]
sample_data(phyloseq)$time <- factor(sample_data(phyloseq)$time)
```

```
#> Formal class 'phyloseq' [package "phyloseq"] with 5 slots
#>   ..@ otu_table:Formal class 'otu_table' [package "phyloseq"] with 2 slots
#>   ..@ tax_table:Formal class 'taxonomyTable' [package "phyloseq"] with 1 slot
#>   ..@ sam_data :'data.frame':    44 obs. of  5 variables:
#> Formal class 'sample_data' [package "phyloseq"] with 4 slots
```

```
#>   ..@ phy_tree : NULL
#>   ..@ refseq   : NULL
```

2.2   Analysis using zitools

The zero inflation analysis steps are wrapped into a single function, called ziMain. It fits a zero-inflated mixture model to the data. Per default structural zeros are estimated from counts using features (=rows) and samples (=columns) as predictor variables when fitting a zero inflated negative binomial model. Based on the fitted model, probabilities that count values are structural zeros are calculated. Considering these probabilities, the function generates a deinflated count matrix by replacing predicted structural zeros with NA and simultaneously computes weights given that a zero count is a structural zero. Thus, the ziMain function integrates the fitting process, probability calculation, deinflation, and weight calculation by a single function call.

```
Zi <- ziMain(phyloseq)
print(Zi)
#> Formal class 'Zi' [package "zitools"]
#>   features (rows), samples (columns)
#>  5720 data points, 1276(22.308%) zeros, 1219(21.311%)
#>        structual zeros estimated with count ~ sample + feature
#> Formal class 'Zi' [package "zitools"] with 5 slots
#>   ..@ inputdata       :Formal class 'phyloseq' [package "phyloseq"] with 5 slots
#>   ..@ inputcounts     : int [1:130, 1:44] 0 6 0 224 0 169 0 20 360 10 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..@ model           :List of 1
#>   ..@ deinflatedcounts: int [1:130, 1:44] NA 6 NA 224 NA 169 NA 20 360 10 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..@ weights         : num [1:130, 1:44] 0.0748 1 0.0554 1 0.0084 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#> Use str(object) to inspect the whole object structure.
```

2.3   Basic Statistic Quantities

Following the OOP concept of polymorphism, already implemented functions can be called with an Zi-object without further arguments as demonstrated in the following examples. Please note, that the basic functionality was not reimplemented. Instead only a wrapper methods for Zi-class objects were written that pass the appropriate information to existing functions. This ensures that the full functionality coincides.

```
mean(Zi)
#> [1] 259.3661
sd(Zi)
#> [1] 524.7669
var(Zi)
#> [1] 275380.3

median(Zi)
#> [1] 67
quantile(Zi)
```
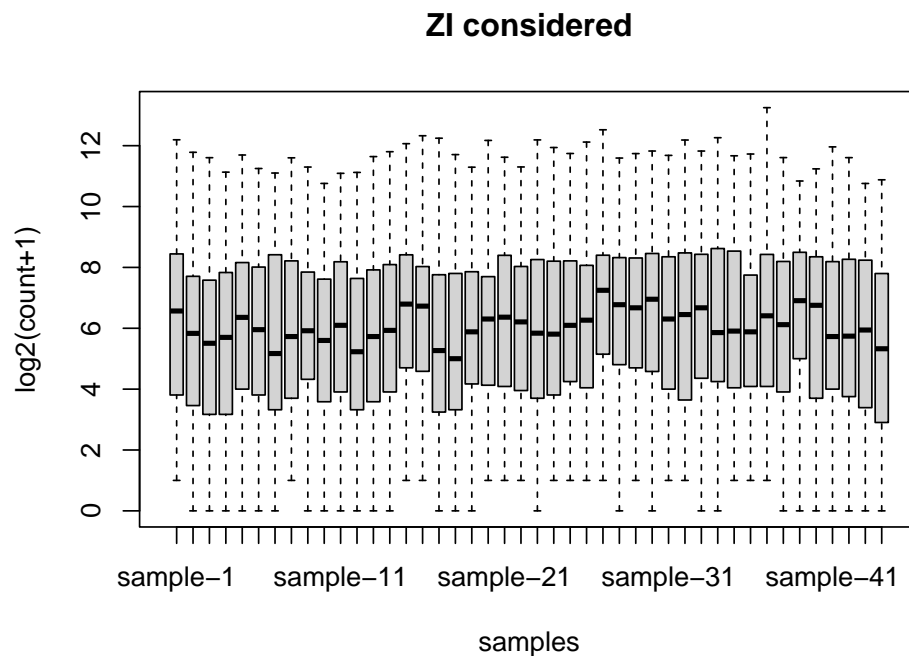
```
#>    0%  25%  50%  75% 100%
#>     0   13   67  289 9734
```
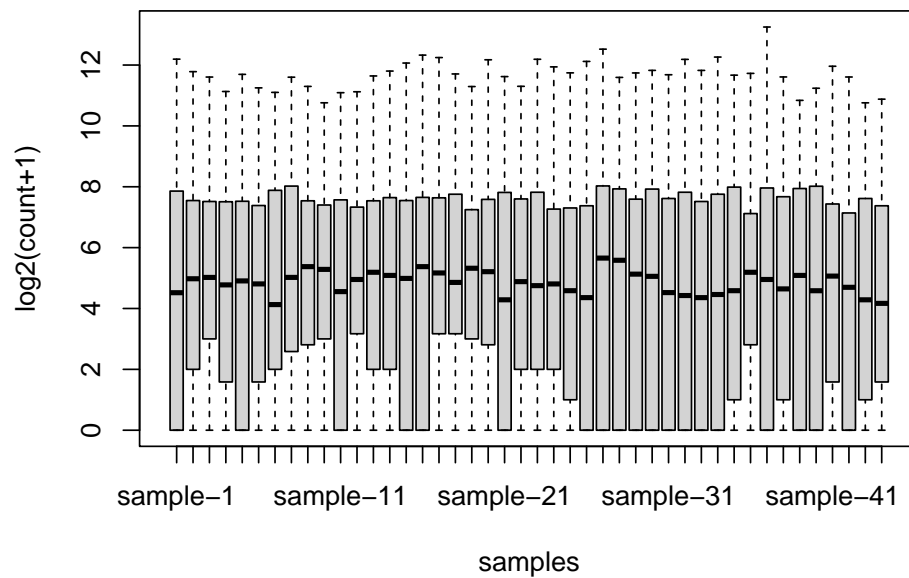
2.4   Boxplots

Batch effects within the dataset can be visualized by plotting the data as boxplots over samples. The boxplot function is overloaded for the Zi-class and can be called without further arguments.

```
boxplot(log2p(Zi), xlab = "samples", ylab = "log2(count+1)",
        main = "ZI considered")
```
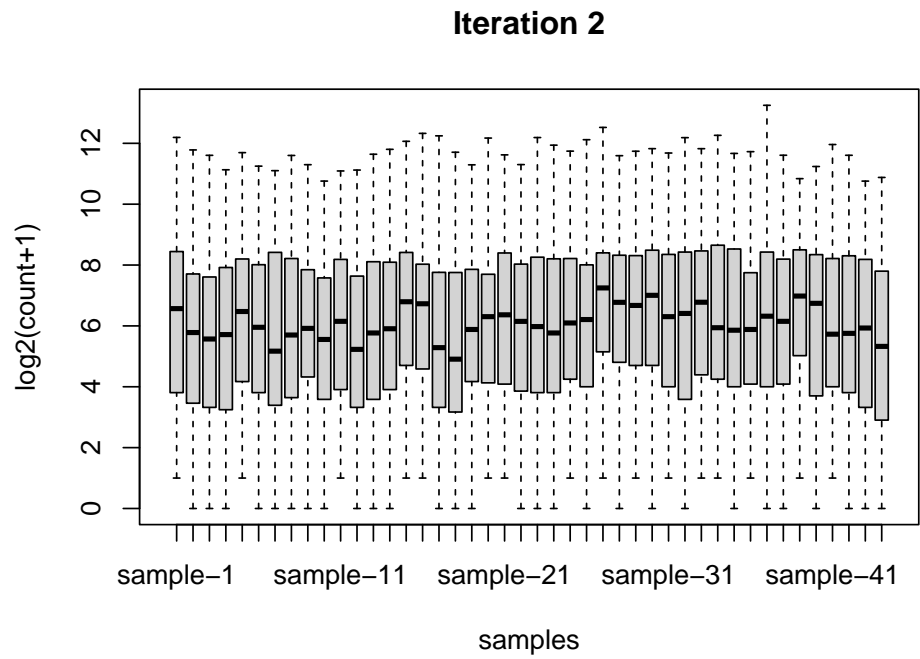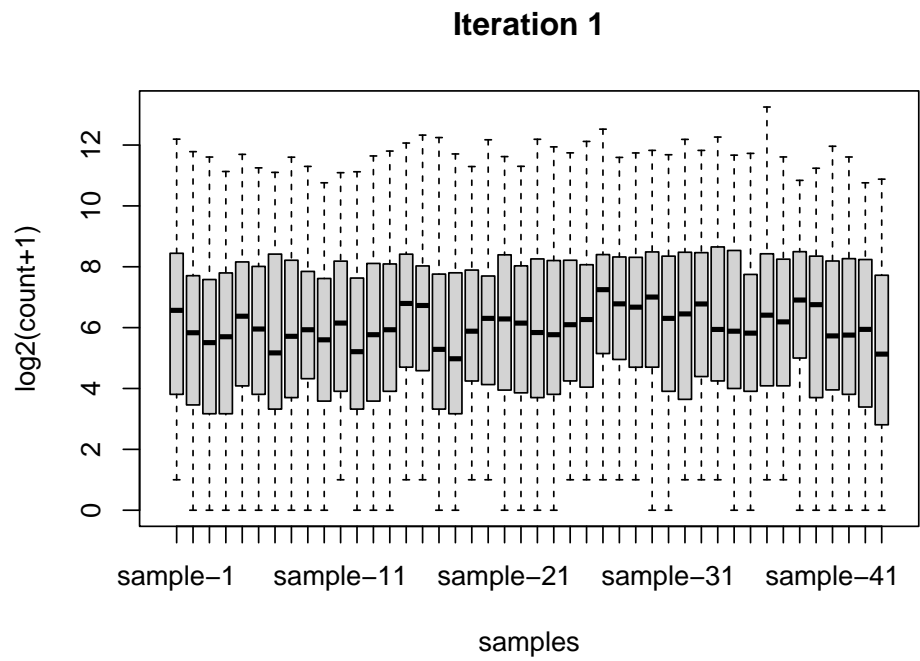
ZI considered


```
boxplot(log2p(inputcounts(Zi)), xlab = "samples", ylab = "log2(count+1)",
        main = "ZI not considered")
```

ZI not considered
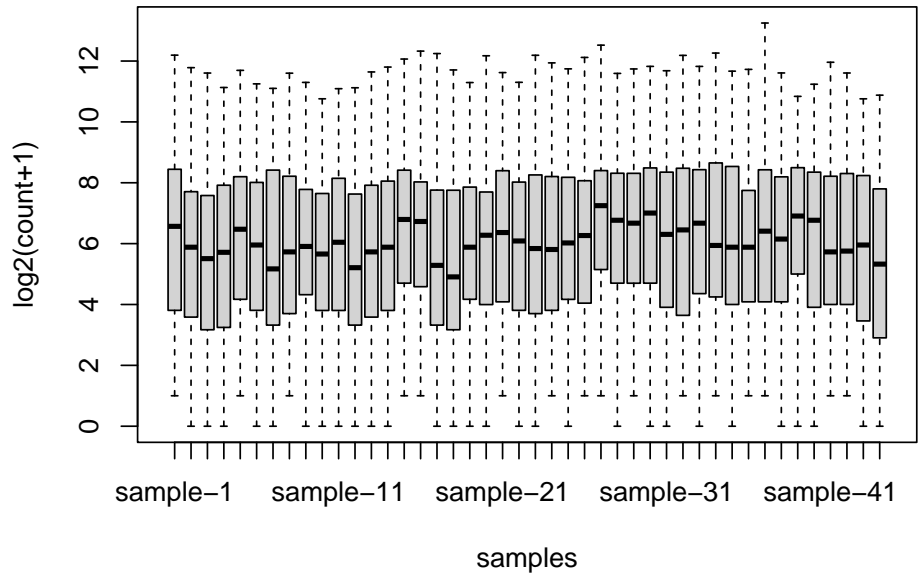

Considering that the process of generating deinflatedcounts is based on a random drawings, resample_deinflatedcounts repeating the process of drawing structural zeros can be performed to evaluate influences of this random drawing process. Visualization based on resampled deinflatedcounts via boxplots shows only minor differences. Slight differences are highlighted by red circles. This suggests that the randomness inherent in the drawing process has only a small influence on the distributions and summary statistics within individual samples.
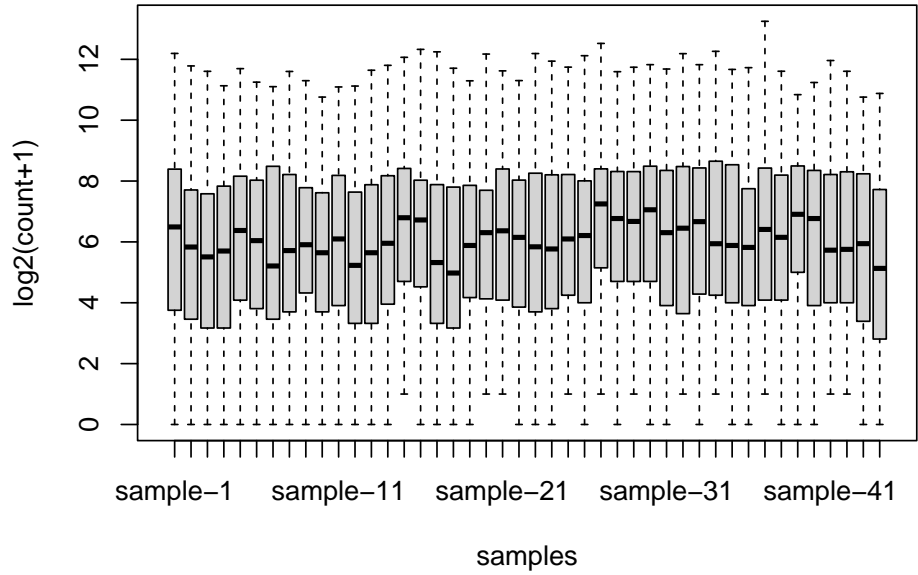
```
i <- 1
repeat {
    Zi <- resample_deinflatedcounts(Zi)
    boxplot(log2p(Zi), xlab = "samples", ylab = "log2(count+1)",
        main = paste("Iteration", i))
    i = i+1
    if(i==6){
    break
    }
}
```
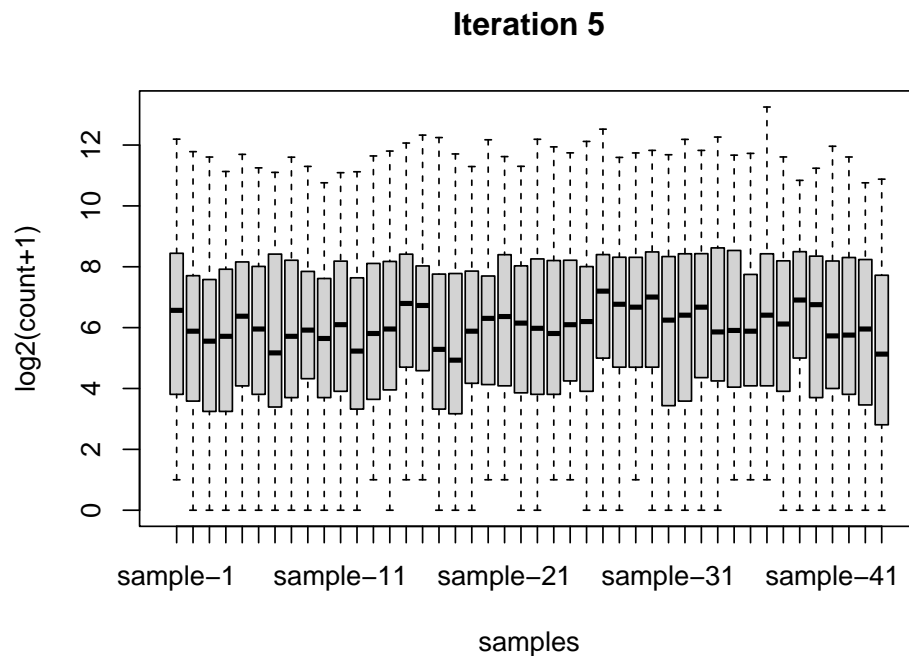
Iteration 1

samples

Iteration 2

samples

Iteration 3

Iteration 4

Iteration 5

2.5   Differential Abundance Analysis

To identify differentially abundant taxa between two groups of the dataset (e.g. patients vs. controls), differential abundance analysis using the [DESeq2]{https://bioconductor.org/packages/release/bioc/html/DESeq2.html} package can be performed. As a first step, the object of a Zi-class has to be transformed in a DESeqDataSet object using zi2deseq2. In this process, weights for down-weighting structural zeros are included in the DESeqDataSet. Therefore, when performing the actual differential abundance analysis weights are automatically incorporated in the calculation of differentially abundant taxa. Log2 fold changes are calculated by DESeq() and the Wald statistic is used to calculate p-values and to identify differentially abundant taxa. In this example, only positive counts are used to estimate the size factors required for normalization within DESeq2, i.e. zeros do not contribute. For simplicity, the model used for the differential abundance analysis is a simple model with only one factor (time), with the timepoints being 1, before intervention and 2, after intervention.

```
DDS <- zi2deseq2(Zi, ~time)
#> converting counts to integer mode
DDS$Subject <- relevel(DDS$time, ref = "1")
DDS <-  DESeq(DDS, test = "Wald", fitType = "local", sfType = "poscounts")
#> estimating size factors
#> estimating dispersions
#> gene-wise dispersion estimates
#> mean-dispersion relationship
#> final dispersion estimates
#> fitting model and testing
#> -- replacing outliers and refitting for 18 genes
#> -- DESeq argument 'minReplicatesForReplace' = 7
#> -- original counts are preserved in counts(dds)
#> estimating dispersions
#> fitting model and testing
```
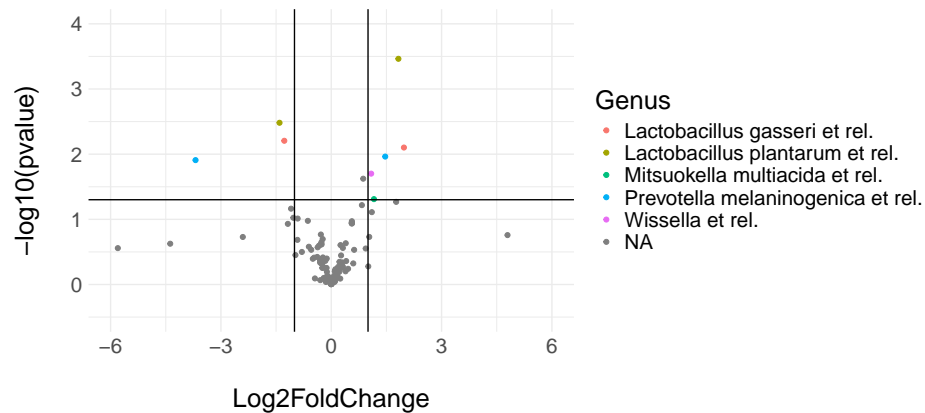
```
(result <- results(DDS, cooksCutoff = FALSE))
#> log2 fold change (MLE): time 2 vs 1
#> Wald test p-value: time 2 vs 1
#> DataFrame with 130 rows and 6 columns
#>                             baseMean log2FoldChange     lfcSE        stat
#>                            <numeric>      <numeric> <numeric>   <numeric>
#> Actinomycetaceae            4.188827    -0.00666631  0.472869  -0.0140976
#> Aerococcus                  5.641624     0.13599050  0.470069   0.2892990
#> Aeromonas                   6.101542     0.31312343  0.495150   0.6323809
#> Akkermansia               507.121152     0.05519229  0.347252   0.1589404
#> Alcaligenes faecalis et rel.  0.764668  -0.97271514  1.048221  -0.9279675
#> ...                              ...            ...       ...         ...
#> Weissella et rel.          13.82174      -0.381414  0.431929   -0.883049
#> Vibrio                      9.28438       0.394165  0.330282    1.193422
#> Wissella et rel.           10.25831      -1.276348  0.466582   -2.735529
#> Xanthomonadaceae           11.70166      -0.302808  0.384044   -0.788473
#> Yersinia et rel.            2.39337      -0.500261  0.598083   -0.836440
#>                               pvalue       padj
#>                            <numeric> <numeric>
#> Actinomycetaceae            0.988752  0.989108
#> Aerococcus                  0.772353  0.989108
#> Aeromonas                   0.527138  0.989108
#> Akkermansia                 0.873716  0.989108
#> Alcaligenes faecalis et rel.  0.353424  0.989108
#> ...                              ...       ...
#> Weissella et rel.          0.37720989  0.989108
#> Vibrio                     0.23270416  0.931082
#> Wissella et rel.           0.00622802  0.257167
#> Xanthomonadaceae           0.43042027  0.989108
#> Yersinia et rel.           0.40290717  0.989108
result <- as.data.frame(result)
```

2.6   Plot Differential Abundance Result

The most common depiction of the magnitude of the differential abundances and their significance is a so-called Volcano plot with fold-changes on the horizontal axis and p-values on the negative log10-scale on the vertical axis. Volcano plot can be generated manually as demonstrated in the following.

```
print(ggplot(data=result_df, aes(x=log2FoldChange,
    y=-log10(pvalue),
    color = Genus)) +
    geom_point()+
    theme_minimal()+
    xlim(-6,6)+
    ylim(-0.5,4)+
    ylab("-log10(pvalue)\n")+
    xlab("\nLog2FoldChange")+
    geom_vline(xintercept=c(-1, 1), col="black") +
    geom_hline(yintercept=-log10(0.05), col="black")+
```
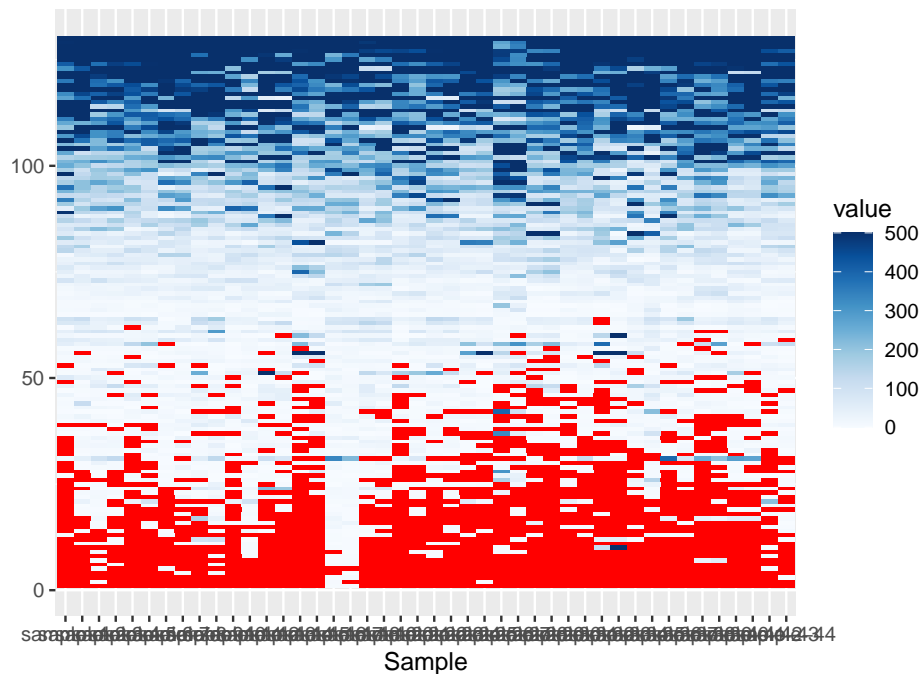
```
        theme(text = element_text(size = 20))+
        theme(panel.spacing.x = unit(2, "lines")))
```


2.7   Missing Value Heatmap

Heatmaps allow for discovering patterns of variation in the data by identifying regions of high or low abundance. By plotting a MissingValueHeatmap the amount of predicted structural zeros can be visualized as they are represented by NA values of the deinflated count matrix. For demonstration purposes, all values > 500 are set to 500 for better color coding.

```
MissingValueHeatmap(Zi2, xlab = "Sample")
```
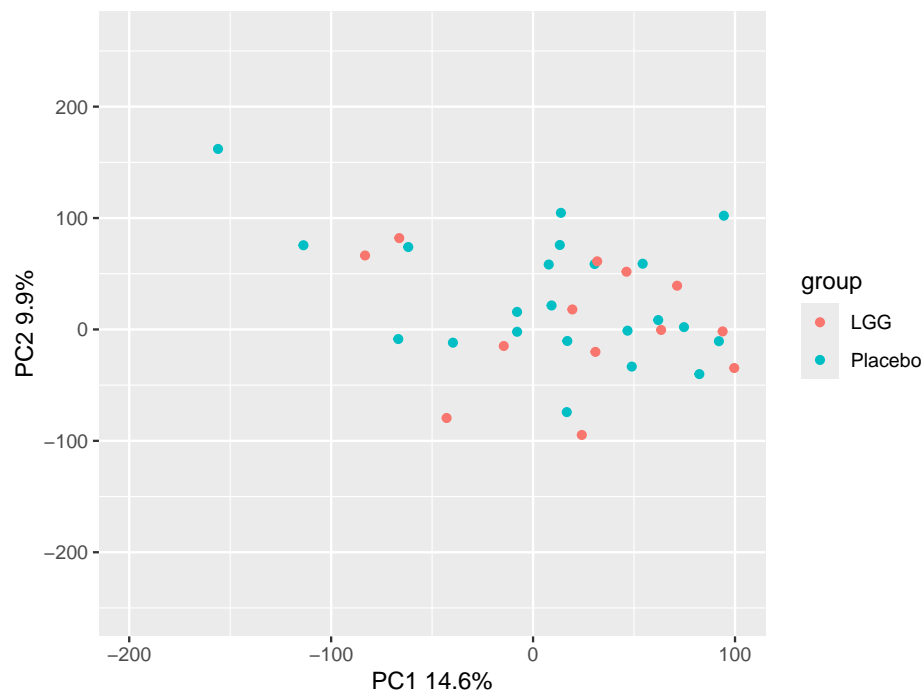
2.8   Principal Component Analysis

There is no principal component analysis method that enables the direct consideration of individual weights for each data point. However, by splitting the calculations, visualization is feasible. First, weighted correlations are computed by taking into account the weights assigned to each data point. Then, principal components are calculated from these correlations and the weighted scaled data is projected to these principal components.

```r
PCA <- princomp(covmat = cor(t(Zi)), cor = FALSE)
centered_data <- (inputcounts(t(Zi))-colMeans2(t(Zi)))/sqrt(colVars(t(Zi)))
loadings <- PCA$loadings
scores <- centered_data %*% loadings
PCA$scores <- scores

df_PCA <- data.frame("PC1" = PCA[["scores"]][,1], "PC2" = PCA[["scores"]][,2],
    "group" = sample_data(Zi)$group)
```

```
#> Warning: Removed 9 rows containing missing values or values outside the scale range
#> (`geom_point()`).
```
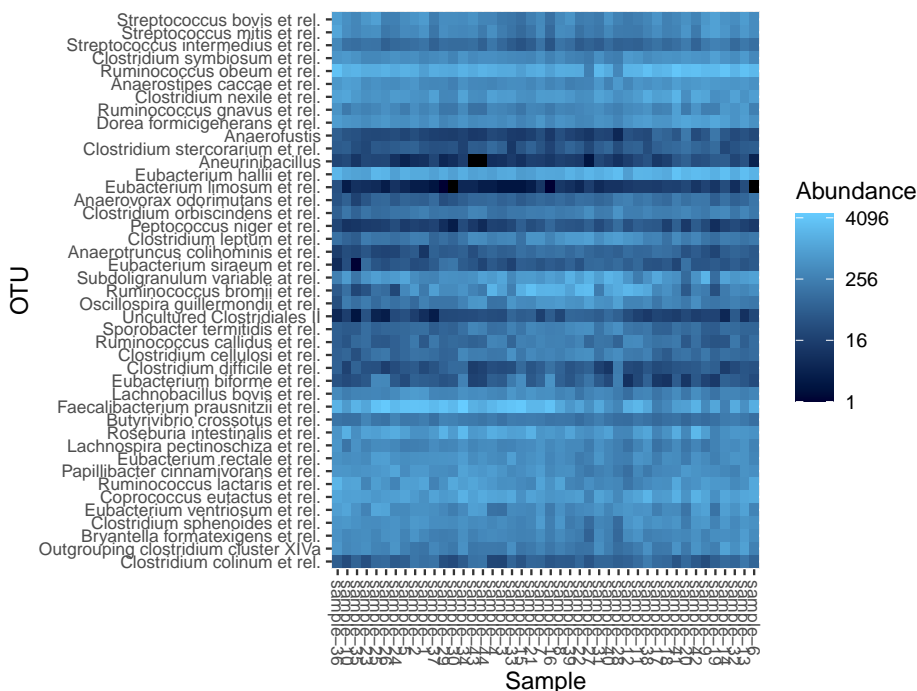

2.9   Interaction with the phyloseq package

To allow for an even wider range of analysis methods, the function zi2phyloseq() creates a phyloseq object where the otu_table is replaced with deinflatedcounts. Therefore, functions of the phyloseq package can be used while also accounting for the zero inflation weights calculated via our zitools package.

```r
new_phyloseq <- zi2phyloseq(Zi)
```
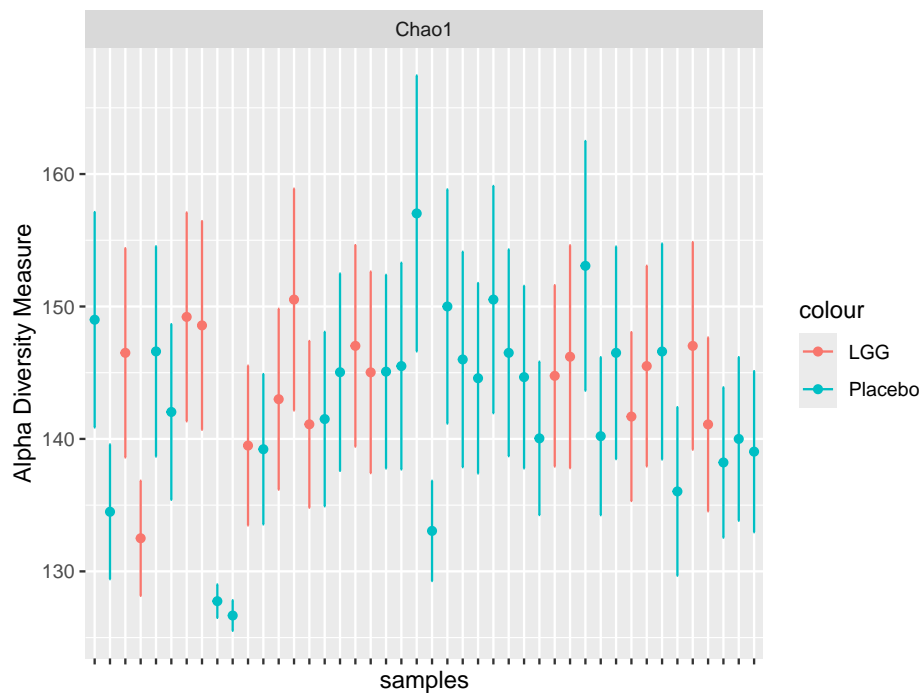
```
str(otu_table(new_phyloseq), max.level = 3)
#> Formal class 'otu_table' [package "phyloseq"] with 2 slots
#>   ..@ .Data       : int [1:130, 1:44] NA 6 NA 224 NA 169 NA 20 360 10 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   ..@ taxa_are_rows: logi TRUE
#>   ..$ dim      : int [1:2] 130 44
#>   ..$ dimnames:List of 2
#>   .. ..$ : chr [1:130] "Actinomycetaceae" "Aerococcus" "Aeromonas" "Akkermansia" ...
#>   .. ..$ : chr [1:44] "sample-1" "sample-2" "sample-3" "sample-4" ...
```

```
subset <- subset_taxa(new_phyloseq, Phylum=="Firmicutes")
subset <- prune_taxa(names(sort(taxa_sums(subset),TRUE)[1:300]), subset)
(plot_heatmap(subset, ))
```


One option to estimate the alpha diversity is calculating the Chao1 index using plot_richness of the phyloseq package

```
plot_richness(new_phyloseq, measures = c("Chao1"),
    color = sample_data(new_phyloseq)$group)+
    theme(axis.text.x = element_blank())
```

3 Session Info

```
sessionInfo()
#> R version 4.5.0 RC (2025-04-04 r88126)
#> Platform: x86_64-pc-linux-gnu
#> Running under: Ubuntu 24.04.2 LTS
#>
#> Matrix products: default
#> BLAS:   /home/biocbuild/bbs-3.21-bioc/R/lib/libRblas.so
#> LAPACK: /usr/lib/x86_64-linux-gnu/lapack/liblapack.so.3.12.0  LAPACK version 3.12.0
#>
#> locale:
#>  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
#>  [3] LC_TIME=en_GB              LC_COLLATE=C
#>  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
#>  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
#>  [9] LC_ADDRESS=C               LC_TELEPHONE=C
#> [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
#>
#> time zone: America/New_York
#> tzcode source: system (glibc)
#>
#> attached base packages:
#> [1] stats4    stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
```

```
#>  [1] microbiome_1.30.0          lubridate_1.9.4
#>  [3] forcats_1.0.0              stringr_1.5.1
#>  [5] dplyr_1.1.4                purrr_1.0.4
#>  [7] readr_2.1.5                tidyr_1.3.1
#>  [9] tibble_3.2.1               ggplot2_3.5.2
#> [11] tidyverse_2.0.0            DESeq2_1.48.0
#> [13] SummarizedExperiment_1.38.0 Biobase_2.68.0
#> [15] MatrixGenerics_1.20.0      matrixStats_1.5.0
#> [17] GenomicRanges_1.60.0       GenomeInfoDb_1.44.0
#> [19] IRanges_2.42.0             S4Vectors_0.46.0
#> [21] BiocGenerics_0.54.0        generics_0.1.3
#> [23] phyloseq_1.52.0            zitools_1.2.0
#> [25] BiocStyle_2.36.0
#>
#> loaded via a namespace (and not attached):
#>  [1] permute_0.9-7          rlang_1.1.6              magrittr_2.0.3
#>  [4] ade4_1.7-23            compiler_4.5.0           mgcv_1.9-3
#>  [7] vctrs_0.6.5            reshape2_1.4.4           pkgconfig_2.0.3
#> [10] crayon_1.5.3           fastmap_1.2.0            XVector_0.48.0
#> [13] labeling_0.4.3         rmarkdown_2.29           tzdb_0.5.0
#> [16] UCSC.utils_1.4.0       tinytex_0.57             xfun_0.52
#> [19] jsonlite_2.0.0         biomformat_1.36.0        rhdf5filters_1.20.0
#> [22] DelayedArray_0.34.0    Rhdf5lib_1.30.0          BiocParallel_1.42.0
#> [25] VGAM_1.1-13            parallel_4.5.0           cluster_2.1.8.1
#> [28] R6_2.6.1               stringi_1.8.7            RColorBrewer_1.1-3
#> [31] Rcpp_1.0.14            bookdown_0.43            iterators_1.0.14
#> [34] knitr_1.50             pscl_1.5.9               Matrix_1.7-3
#> [37] splines_4.5.0          igraph_2.1.4             timechange_0.3.0
#> [40] tidyselect_1.2.1       abind_1.4-8              yaml_2.3.10
#> [43] vegan_2.6-10           codetools_0.2-20         lattice_0.22-7
#> [46] plyr_1.8.9             withr_3.0.2              Rtsne_0.17
#> [49] evaluate_1.0.3         survival_3.8-3           Biostrings_2.76.0
#> [52] pillar_1.10.2          BiocManager_1.30.25      foreach_1.5.2
#> [55] hms_1.1.3              munsell_0.5.1            scales_1.3.0
#> [58] glue_1.8.0             tools_4.5.0              data.table_1.17.0
#> [61] locfit_1.5-9.12        rhdf5_2.52.0             grid_4.5.0
#> [64] ape_5.8-1              colorspace_2.1-1         nlme_3.1-168
#> [67] GenomeInfoDbData_1.2.14 cli_3.6.4               S4Arrays_1.8.0
#> [70] gtable_0.3.6           digest_0.6.37            SparseArray_1.8.0
#> [73] farver_2.1.2           htmltools_0.5.8.1        multtest_2.64.0
#> [76] lifecycle_1.0.4        httr_1.4.7               MASS_7.3-65
```