Package 'permubiome'

July 31, 2020

Type Package

Title A Permutation Based Test for Biomarker Discovery in Microbiome Data

Version 1.3.1

Date 2020-07-29

Author Alfonso Benitez-Paez

Maintainer Alfonso Benitez-Paez <alfbenpa@gmail.com>

Description The permubiome R package was created to perform a permutation-based non-parametric analysis on microbiome data for biomarker discovery aims. This test executes thousands of comparisons in a pairwise manner, after a random shuffling of data into the different groups of study with a prior selection of the microbiome features with the largest variation among groups. Previous to the permutation test itself, data can be normalized according to different methods proposed to handle microbiome data ('proportions' or 'Anders'). The median-based differences between groups resulting from the multiple simulations are fitted to a normal distribution with the aim to calculate their significance. A multiple testing correction based on Benjamini-Hochberg method (fdr) is finally applied to extract the differentially presented features between groups of your dataset. LATEST UPDATES: v1.1 and olders incorporates function to parse COLUMN format; v1.2 and olders incorporates -optimize- function to maximize evaluation of features with largest inter-class variation; v1.3 and olders includes the -size.effect- function to perform estimation statistics using the bootstrap-coupled approach implemented in the 'dabestr' (>=0.3.0) R package.

License GPL-3

Imports ggplot2, rlang, dabestr, gridExtra, Matrix

NeedsCompilation no

Repository CRAN

Date/Publication 2020-07-31 06:40:03 UTC

R topics documented:

`get.data` *Parsing the data file.*

Description

This function prompts for the file contained all the data needed to process. You only have to execute this function in the working directory where your file is stored properly formatted as requested.

The input file is a tab-delimited text matrix as follows:

Sample	Class	feature(1)	feature(2)	feature(n)	...
sampleA	classX	counts(A1)	counts(A2)	counts(An)	...
sampleB	classY	counts(B1)	counts(B2)	counts(Bn)	...
sampleC	classX	counts(C1)	counts(C2)	counts(Cn)	...
sampleD	classY	counts(D1)	counts(D2)	counts(Dn)	...

From the version 1.1 on you will be able to load your data as COLUMN format, just adding the "Class" information in the second row as follows:

Sample	sampleA	sampleB	sampleC	sampleD	...
Class	classX	classY	classX	classY	...
feature(1)	counts(A1)	counts(B1)	counts(C1)	counts(D1)	...
feature(2)	counts(A2)	counts(B2)	counts(C2)	counts(D2)	...
feature(3)	counts(A3)	counts(B3)	counts(C3)	counts(D3)	...
feature(4)	counts(A4)	counts(B4)	counts(C4)	counts(D4)	...
feature(n)	counts(An)	counts(Bn)	counts(Cn)	counts(Dn)	...

Usage

```
get.data()
```

Author(s)

Alfonso Benitez-Paez

References

Benitez-Paez A. 2020. Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. [https://cran.r-project.org]. Benitez-Paez A, et al. mSystems. 2020;5:e00857-19. doi: 10.1128/mSystems.00857-19.

Examples

```
## The function is currently defined as
function ()
{
    DATA <- readline("Type the name of your data set : ")
    if (DATA == "") {
        tb <- read.table(system.file("extdat", "DATA_2", package = "permubiome"),
            header = T, sep = "\t")
       print(paste("As you declare no input file, the permubiome test data was loaded"))
        save(tb, file = "permubiome.RData")
    }
    else {
        FORMAT <- readline("Type the format of your data set (PERMUBIOME or COLUMN): ")
        if (FORMAT == "PERMUBIOME") {
            tb <- read.table(DATA, header = T, sep = "\t")
            save(tb, file = "permubiome.RData")
        }
        else {
            biom <- read.table(DATA, sep = "\t")
            tb <- t(biom)
            colnames(tb) <- tb[1, ]
            rownames(tb) <- NULL
            tb = tb[-1, ]
            labels <- colnames(tb)
            tb <- as.data.frame(tb)
            for (i in 3:length(labels)) {
                tb[, i] <- as.numeric(as.character(tb[, i]))
            }
            save(tb, file = "permubiome.RData")
        }
    }
    load("permubiome.RData")
    df <- as.data.frame(tb)
    classes <- levels(df$Class)
    samples <- nrow(df)
    print(paste("Your data file contains:", samples, "samples"))
    print(paste("The classes in your data file are:", classes[1],
        "and", classes[2]))
    print(paste("The number of different categories to compare are:",
        (ncol(tb) - 2)))
    save(DATA, df, REFERENCE, classes, file = "permubiome.RData")
  }
```

`normalize` *Normalize the microbiome dataset prior to perform the permutation test.*

Description

A critical aspect when working with microbiome data is to achieve a proper normalization to the retrieved counts, thus overpassing the variability in terms of sequencing efforts or coverage. There

are several ways to do normalization, and we have implemented two well-known methods whose choice will depend on the research question investigated and the researcher's preference. Optionally, if you don't feel comfortable with normalization methods implemented in this package or if your data are already normalized, you have the option of performing no normalization on your data (*method*=0).

Usage

```
normalize(prevalence = 0.3, method = 1)
```

Arguments

`prevalence`	This controls the prevalence of microbiome features across samples in order to keep those with higher occurrence in the cohort of samples under survey. If you have 20 samples and declare a *prevalence* = 0.3 (default), the algorithm will remove those categories with less prevalence than 6 samples. Although the permutation test deals fairly well with zeros, we recommend setting a restricted value in order to improve the statistics for the biomarker discovery (i.e *prevalence* => 0.3).
`method`	Describes the normalization method to be used. We implemented two different strategies to normalize the microbiome data: (1) corresponds to the relative proportion of counts to the features. After retrieving the relative abundance for every feature in very sample the normalization process generate the number of reads corresponding to the features per million reads; (2) corresponds with normalization method described by Anders & Huber (2010), which uses a size factor to correct differences in sequencing coverage. If the user decides not to perform normalization, it must declare *method* = 0.

Author(s)

Alfonso Benitez-Paez

References

Benitez-Paez A. 2020. Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. [https://cran.r-project.org]. Benitez-Paez A, et al. mSystems. 2020;5:e00857-19. doi: 10.1128/mSystems.00857-19.

Examples

```
## The function is currently defined as
function (prevalence = 0.3, method = 1)
{
    load("permubiome.RData")
    df_norm <- df
    if (method == 1) {
        y <- array(, nrow(df_norm))
        for (j in 1:nrow(df_norm)) {
            y[j] <- sum(df_norm[j, 3:ncol(df_norm)])
        }
```

```
        for (l in 3:ncol(df_norm)) {
            for (m in 1:nrow(df_norm)) {
                df_norm[m, l] <- round((df_norm[m, l]/y[m]) *
                  1e+06, digits = 0)
            }
        }
        for (i in ncol(df_norm):3) {
            if (sum(df_norm[, i] == "0") >= (nrow(df_norm) *
                1 - prevalence)) {
                df_norm[, i] <- NULL
            }
        }
    }
    else if (method == 2) {
        for (i in ncol(df_norm):3) {
            if (sum(df_norm[, i] == 0) >= (nrow(df_norm) * 1 - prevalence)) {
                df_norm[, i] <- NULL
            }
        }
        sfactor_matrix <- matrix(, ncol = ncol(df_norm) - 2,
            nrow = nrow(df_norm))
        y <- array(, nrow(df_norm))
        for (m in 1:nrow(df_norm)) {
            for (l in 3:ncol(df_norm)) {
                sfactor_matrix[m, l - 2] <- signif((df_norm[m,
                  l]/mean(df_norm[, l])), digits = 3)
            }
            y[m] <- median(sfactor_matrix[m, 1:ncol(sfactor_matrix)])
        }
        for (a in 3:ncol(df_norm)) {
            for (b in 1:nrow(df_norm)) {
                df_norm[b, a] <- round((df_norm[b, a] * y[b]),
                  digits = 0)
            }
        }
    }
    else if (method == 0) {
        head(df_norm)
        print(paste("Your dataset was not normalized according to method option: 0"))
    }
    else {
        print(paste("Select and appropiate method for normalization: 1 ('proportions'),
        2 ('anders'), or 0 ('none')"))
    }
    print(paste("Your normalized data now contains:", ncol(df_norm) -
        2, "normalize categories ready to analize"))
    save(df_norm, file = "permubiome.RData")
}
```

---

optimize *Optimization for detection of features with larger variation between classes*

Description

This function is the previous step to the permutation test and it optimizes the detection of features differentially distributed between classes. The intra- and inter-classes distances are calculated with log-transformed data and then a ratio test is done to maximize the variation between classes. This pre-processing penalizes the microbiome features with a larger intra-class and lower inter-class variation, which would interfere with the statistical estimations to executed during the permutation test.

Usage

```
optimize()
```

Author(s)

Alfonso Benitez-Paez

References

Benitez-Paez A. 2020. Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. [https://cran.r-project.org]. Benitez-Paez A, et al. mSystems. 2020;5:e00857-19. doi: 10.1128/mSystems.00857-19.

Examples

```
## The function is currently defined as
function ()
{
    load("permubiome.RData")
    df_norm <- df_norm
    REFERENCE <- REFERENCE
    classes <- levels(df_norm$Class)
    if (REFERENCE == "") {
        REFERENCE <- classes[1]
    }
    else if (REFERENCE == classes[2]) {
        classes[2] <- classes[1]
        classes[1] <- REFERENCE
    }
    df_norm$Class <- factor(df$Class, levels = (c(classes[1],
        classes[2])))
    group1 <- subset(df_norm, Class == classes[1])
    group2 <- subset(df_norm, Class == classes[2])
    categories <- colnames(df_norm)
    distance_matrix <- matrix(, nrow = ncol(df_norm) - 2, ncol = 7,
        byrow = T)
    colnames(distance_matrix) <- c("Category", paste("SumDist ",
        "[", classes[1], "]", sep = ""), paste("SumDist ", "[",
        classes[2], "]", sep = ""), "ClassDist", "RatioDist",
        "Delta-Log", "Z-score")
    for (i in 3:(ncol(group1))) {
        mydata1 <- group1[, i]
```

```
        sumdist1 <- log10(sum(abs(apply(combn(mydata1, 2), 2,
            diff))))
        distance_matrix[(i - 2), 1] <- categories[i]
        distance_matrix[(i - 2), 2] <- sumdist1
    }
    for (j in 3:(ncol(group2))) {
        mydata2 <- group2[, j]
        sumdist2 <- log10(sum(abs(apply(combn(mydata2, 2), 2,
            diff))))
        distance_matrix[(j - 2), 3] <- sumdist2
    }
    classes_matrix <- matrix(, nrow = ncol(group1) - 2, ncol = (nrow(group1) *
        nrow(group2)), byrow = T)
    rownames(classes_matrix) <- colnames(group1[3:ncol(group1)])
    features <- colnames(group1)
    for (k in 3:(ncol(group1))) {
        classdist <- vector()
        for (l in 1:nrow(group1[k])) {
            classdist_tmp <- as.list(abs(group2[k] - group1[l,
                k]))
            classdist <- c(classdist, classdist_tmp[[features[k]]])
        }
        classes_matrix[(k - 2), ] <- classdist
    }
    inter.class.dist <- as.list(rowSums(classes_matrix))
    for (m in 3:(ncol(group1))) {
        distance_matrix[(m - 2), 4] <- log10(inter.class.dist[[features[m]]])
    }
   distance_matrix[, 5] <- as.numeric(distance_matrix[, 4])/((as.numeric(distance_matrix[,
        3])/as.numeric(distance_matrix[, 2])))
    distance_matrix[, 6] <- abs(as.numeric(distance_matrix[,
        5]) - as.numeric(distance_matrix[, 4]))
    distance_matrix[, 7] <- (as.numeric(distance_matrix[, 6]) -
        mean(as.numeric(distance_matrix[, 6])))/sd(as.numeric(distance_matrix[,
        6]))
    selected_features <- subset(distance_matrix, abs(as.numeric(distance_matrix[,
        6])) > quantile(as.numeric(distance_matrix[,6]),0.96))
    save(df, df_norm, REFERENCE, classes, distance_matrix, selected_features,
        file = "permubiome.RData")
  }
```

`permutation` *Permutation-based non-parametric analysis to infer differential abundance of features between groups.*

Description

This function performs multiple simulations for every feature present in your dataset. All observations are randomly distributed between groups and the median's differences are calculated for all simulations. Differences calculated from simulations are fitted to the normal distribution, Z-scores

are obtained, and the respective probability to reject the null hypothesis is then calculated. A multiple testing correction based on Benjamini-Hochberg method is done to uncover the biomarkers associated with your dataset classes. FDR threshold for differential abundance can be set at 0.1.

Usage

```
permutation(nperm = 1000, write.output = TRUE)
```

Arguments

nperm	The number of permutations to be executed during the analysis (1000 as default). The higher the number of permutations, the more precise will be the p-value returned and the function becomes more time-consuming. We recommend to use *nperm* = 1000 as the minimum.
write.output	When *TRUE* (as default), a sorted output file is generated and stored in the working directory. Control for the number of features to be present in the output is allowed with the "all" or "selected" parameters prompted.

Author(s)

Alfonso Benitez-Paez

References

Benitez-Paez A. 2020. Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. [https://cran.r-project.org]. Benitez-Paez A, et al. mSystems. 2020;5:e00857-19. doi: 10.1128/mSystems.00857-19.

Examples

```
## The function is currently defined as
function (nperm = 1000, write.output = TRUE)
{
    Class <- NULL
    load("permubiome.RData")
    df_norm <- df_norm
    selected_features <- selected_features
    tags.in <- selected_features[, 1]
    tags.out <- setdiff(colnames(df_norm[3:ncol(df_norm)]), tags.in)
    for (a in ncol(df_norm):3) {
        if ((colnames(df_norm[a]) %in% tags.out)) {
            df_norm[, a] <- NULL
        }
    }
    classes <- levels(df_norm$Class)
    if (REFERENCE == "") {
        REFERENCE <- classes[1]
    }
    else if (REFERENCE == classes[2]) {
        classes[2] <- classes[1]
        classes[1] <- REFERENCE
```

```
}
df_norm$Class <- factor(df$Class, levels = (c(classes[1],
    classes[2])))
group1 <- subset(df_norm, Class == classes[1])
group2 <- subset(df_norm, Class == classes[2])
categories <- colnames(df_norm)
size1 <- nrow(group1)
size2 <- nrow(group2)
size <- size1 + size2
pvalue_matrix <- matrix(, nrow = ncol(df_norm) - 2, ncol = 5,
    byrow = T)
colnames(pvalue_matrix) <- c("Category", paste("Median ",
    classes[1], sep = ""), paste("Median ", classes[2], sep = ""),
    "p.value", "p.adj (fdr)")
print(paste("Permutation test in progress - This can take some seconds or minutes!"))
for (i in 3:(ncol(df_norm))) {
    category <- categories[i]
    diff <- median(group1[, i]) - median(group2[, i])
    x <- c(group1[, i], group2[, i])
    y <- array(, nperm)
    for (j in 1:nperm) {
        set <- sample(size, size2, replace = FALSE)
        diff_iter <- median(x[set]) - median(x[-set])
        y[j] <- diff_iter
        ref_score <- (diff - mean(y))/sd(y)
    }
    if (ref_score >= 0) {
        pvalue.i <- pnorm(ref_score, lower.tail = F)
    }
    else {
        pvalue.i <- pnorm(ref_score)
    }
    if (pvalue.i != 0) {
        pvalue_matrix[(i - 2), 1] <- category
    }
    if (pvalue.i != 0) {
        pvalue_matrix[(i - 2), 2] <- round(median(group1[,
            i]), digits = 0)
    }
    if (pvalue.i != 0) {
        pvalue_matrix[(i - 2), 3] <- round(median(group2[,
            i]), digits = 0)
    }
    if (pvalue.i != 0) {
        pvalue_matrix[(i - 2), 4] <- format((pvalue.i * 2),
            digits = 7, scientific = F)
    }
    else {
        pvalue_matrix[(i - 2), 2] <- 1
    }
    pb = txtProgressBar(min = 0, max = (ncol(df_norm) - 2),
        initial = 0, style = 3)
    setTxtProgressBar(pb, (i - 2))
```

```
        invisible()
    }
    pvalue_matrix <- pvalue_matrix[order(pvalue_matrix[, 4]),
        ]
    pvalue_matrix[, 5] <- format(p.adjust(as.numeric(pvalue_matrix[,
        4], n = nrow(pvalue_matrix), method = "fdr")), digits = 7,
        scientific = F)
    cat("\n")
    if (write.output == TRUE) {
        all <- readline("Do you want to include all fetures in the output? (yes/no) : ")
        if (substr(all, 1, 1) == "n") {
        select <- as.numeric(readline("Level of significance to output features (i.e. 0.2) : "))
            significant <- subset(pvalue_matrix, pvalue_matrix[,
                5] <= select)
        }
        else {
            significant <- pvalue_matrix
        }
        write.table(significant, file = "permutation.output",
            quote = F, row.names = F, sep = "\t")
        print(paste("Permutation test done and output table printed!"))
    }
    else {
        significant <- pvalue_matrix
        significant
        print(paste("Permutation test done!"))
    }
    save(df, df_norm, REFERENCE, classes, selected_features,
        nperm, tags.in, tags.out, pvalue_matrix, file = "permubiome.RData")
}
```

`plots` *Plotting the features with differential abundance.*

Description

Option to plot individually all features found to be differentially presented in the classes of your dataset.

Usage

```
plots()
```

Details

When executed, the name of the feature as well as the different output options will be prompted.

Author(s)

Alfonso Benitez-Paez

References

Benitez-Paez A. 2020. Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. [https://cran.r-project.org]. Benitez-Paez A, et al. mSystems. 2020;5:e00857-19. doi: 10.1128/mSystems.00857-19.

Examples

```
## The function is currently defined as
function ()
{
    Class <- NULL
    non_zero <- NULL
    Occurring <- NULL
    prevalence <- NULL
    loadNamespace("ggplot2")
    load("permubiome.RData")
    df_norm <- df_norm
    category <- readline("Type the category you want to plot : ")
    if (category == "") {
        category <- colnames(df_norm[3])
        print(paste("As you declared no categories, the very first one of your
        dataset is plotted!"))
    }
    df_to_plot <- df_norm[, c("Sample", "Class", category)]
    classes <- levels(df_to_plot$Class)
    if (REFERENCE == "") {
        REFERENCE <- classes[1]
    }
    else if (REFERENCE == classes[2]) {
        classes[2] <- classes[1]
        classes[1] <- REFERENCE
    }
    df_to_plot$ref <- factor(df_to_plot$Class, levels = (c(classes[1],
        classes[2])))
    p1 <- (ggplot(df_to_plot, aes(df_to_plot$ref, df_to_plot[,
        category], fill = df_to_plot$Class), environment = environment())) +
        geom_boxplot(notch = F, outlier.colour = "blue", outlier.shape = 1,
            outlier.size = 3) + geom_point(colour = "#000000",
        size = 2.5, pch = 19) + scale_fill_manual(values = c("#E41A1C",
        "#377EB8")) + ggtitle(category) + theme(plot.title = element_text(size = 24,
        face = "bold")) + ylab("Normalized read proportion") +
        xlab("Classes") + theme(axis.text = element_text(size = 12),
        axis.title = element_text(size = 16, face = "bold")) +
        coord_flip() + guides(fill = FALSE) + theme(plot.margin = unit(c(0.25,
        0.25, 0.25, 0.25), "cm"))
    non_zero <- as.data.frame((tapply(df_to_plot[[category]],
        df_to_plot$ref, nnzero)))
    total <- as.data.frame((tapply(df_to_plot[[category]], df_to_plot$ref,
        length)))
    prevalence_table <- data.frame(names = factor(c(classes[1],
        classes[2]), levels = c(classes[1], classes[2])), Occurring = c(non_zero[,
        1]), Subjects = c(total[, 1]))
```

```
prevalence_table$prevalence <- (prevalence_table$Occurring/prevalence_table$Subjects) *
    100
 p2 <- ggplot(prevalence_table, aes(x = names, y = prevalence, fill = names,
    width = 0.75)) + geom_bar(stat = "identity", colour = "grey20") +
    scale_fill_manual(values = c("#E41A1C", "#377EB8")) +
    coord_flip() + guides(fill = FALSE) + ggtitle(paste("",
    "", sep = " ")) + theme(plot.title = element_text(size = 24,
  face = "bold")) + ylab("Prevalence (percentage)") + theme(axis.title.y = element_blank(),
    axis.text.y = element_blank(), axis.text.x = element_text(size = 12),
    axis.title = element_text(size = 16, face = "bold")) +
    theme(plot.margin = unit(c(0.25, 0.25, 0.25, 0.25), "cm")) +
    ylim(0, 100)
 output <- readline("Do you want an output file (yes/no)? : ")
 if (substr(output, 1, 1) == "y") {
    extension <- readline("What extension do you prefer for the output plot
    (ps, pdf, jpeg, tiff, png, bmp )? : ")
    tiff(filename = paste(category, extension, sep = "."),
        width = 1000, height = 200, res = 100, units = "px")
    grid.arrange(p1, p2, ncol = 2)
    dev.off()
 }
 else {
    print(grid.arrange(p1, p2, ncol = 2))
 }
 save(df, df_norm, df_to_plot, REFERENCE, classes, file = "permubiome.RData")
}
```

`size.effect` *Executing estimation statistics based on bootstrap-coupled approach*

Description

Assessing the size effect on selected microbiome features found to be differentially abundant between classes. This analysis is based on the Data Analysis using Bootstrap-Coupled Estimation (dabestr) R package and gives you the option to create Gardner-Altman estimation plots individually all features found to be differentially presented in your dataset.

Usage

```
size.effect(category = "", replicates = 5000, n.seed = 12345,
paired = FALSE, plot.file = "tiff", id.pairs = NULL)
```

Arguments

`category`	Name of the microbiome feature, which differential abundance between classes will be further explored.
`replicates`	The number of bootstrap resamples that have to be generated. Integer, default 5000.

n.seed	This specifies the seed used to set the random number generator. Setting a seed ensures that the bootstrap confidence intervals for the same data will remain stable over separate runs/calls of this function. Integer, default 12345.
paired	If TRUE, the two groups are treated as paired samples, please add an extra column (id.pairs) to parse identity of the datapoint. Default FALSE, the control_group group is treated as pre-intervention and the test_group group is considered post-intervention.
plot.file	Extension for plot graphics (ps, pdf, jpeg, tiff, png, bmp). Default "tiff".
id.pairs	Column name for information to parse identity of the datapoint in case of paired data.

Details

Be careful to type the "category" correctly to be analyzed in order to that matches with the table contained information.

Author(s)

Alfonso Benitez-Paez

References

Benitez-Paez A. 2020. Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. [https://cran.r-project.org]. Benitez-Paez A, et al. mSystems. 2020;5:e00857-19. doi: 10.1128/mSystems.00857-19.

Examples

```
## The function is currently defined as
function (category = "", replicates = 5000, n.seed = 12345,
    paired = FALSE, plot.file = "tiff", id.pairs = NULL)
{
    Class <- NULL
    ref <- NULL
    loadNamespace("dabestr")
    loadNamespace("rlang")
    load("permubiome.RData")
    df_norm <- df_norm
    if (paired == TRUE) {
     print(paste("You declared paired data, be sure to include the correct -id.column- argument
to parse the identity of the datapoint!"))
    }
    classes <- levels(df_norm$Class)
    if (REFERENCE == "") {
        REFERENCE <- classes[1]
    }
    else if (REFERENCE == classes[2]) {
        classes[2] <- classes[1]
        classes[1] <- REFERENCE
    }
```

```
    prepare.stats <- dabest(df_norm, Class, category, paired = paired,
        idx = c(classes[1], classes[2]), id.column = id.pairs)
    prepare.stats$y<-quo_set_expr(prepare.stats$y, as.symbol(category))
    print(prepare.stats)
    if (category == "") {
        category <- colnames(df_norm[3])
      print(paste("As you declared no categories, the very first one of your dataset will be
processed!"))
    }
    estimation.stats<-median_diff(prepare.stats, ci = 95, reps = replicates,
        seed = n.seed)
    e_plot <- plot(estimation.stats, group.summaries = "median_quartiles",
        palette = "Set1", rawplot.ylabel = paste(category, "normalized reads",
            sep = " "), tick.fontsize = 12, axes.title.fontsize = 18)
    tiff(filename = paste(category, "estimation", plot.file, sep = "."),
        width = 650, height = 600, res = 100, units = "px")
    e_plot
    dev.off()
    print(e_plot)
    save(df, df_norm, REFERENCE, classes, file = "permubiome.RData")
  }
```

Index