Package 'permubiome'

March 23, 2016

Type Package

Title A Permutation Based Test for Biomarker Discovery in Microbiome Data

Version 1.1

Date 2016-03-22

Author Alfonso Benitez-Paez

Maintainer Alfonso Benitez-Paez <abenitez@iata.csic.es>

Description All the functions compiled in this package were created to perform permutation-based non-parametric analysis on microbiome data for biomarker discovery aims. This test executes thousands of comparisons in pairwise manner, after random shuffling of data into the different groups of study.

License GPL-2

Imports ggplot2

NeedsCompilation no

Repository CRAN

Date/Publication 2016-03-23 00:34:11

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) ...
```

In the latest version 1.1 you will be able to load your data as BIOM 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. & Sanz Y. (2015). Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. In press.

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_1", 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 BIOM): ")
        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(df, 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 three well-known methods whose choice will depend on the research question investigated and 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 perform no normalization on your data (method=0).

Usage

```
normalize(numz = 0.5, method = 1)
```

Arguments

`numz`	The zero control across the features (columns) to keep after the normalization step. If you have 20 samples and declare a numz = 0.5 (default), the algorithm will remove those categories with equal and more than 10 zeros as counts. Although the permutation test deals very well with zeros, we recommend setting a restricted value in order to improve the statistics for the biomarker discovery (i.e numz < 0.3).
`method`	Describes the normalization method to be used. We implemented three different strategies to normalize the microbiome data: (1) corresponds with the relative proportion of counts to the features. After retrieve 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; and (3) corresponds with normalization method described by Paulson et al., (2013), which refers to the Cumulative sum scaling normalization using a "l" parameter that determines the percentile of features to calculate the sum scaling factor. If the user decides not to perform normalization, it must declare (0) as method.

Author(s)

Alfonso Benitez-Paez

References

Benitez-Paez A. & Sanz Y. (2015). Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. In press.

Examples

```
## The function is currently defined as
function (numz = 0.5, 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) *
                numz)) {
                df_norm[, i] <- NULL
            }
        }
    }
    else if (method == 2) {
        for (i in ncol(df_norm):3) {
            if (sum(df_norm[, i] == 0) >= (nrow(df_norm) * numz)) {
                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 == 3) {
        for (i in ncol(df_norm):3) {
            if (sum(df_norm[, i] == 0) >= (nrow(df_norm) * numz)) {
                df_norm[, i] <- NULL
            }
        }
      quantil <- as.numeric(readline("Type the 'l' parameter (percentile between 0.01  and 0.99)
        to perform paulson's normalization (0.95 as default): "))
        if (is.numeric(quantil) != TRUE & quantil > 1) {
            quantile <- 0.95
        }
        y <- array(, nrow(df_norm))
        sfactor <- array(, nrow(df_norm))
        for (m in 1:nrow(df_norm)) {
            x <- array(, ncol(df_norm) - 2)
            for (l in 3:ncol(df_norm)) {
                if (df_norm[m, l] <= quantile(df_norm[m, 3:ncol(df_norm)],
                  quantil, na.rm = T)) {
                  x[l - 2] <- df_norm[m, l]
                }
                else {
                  x[l - 2] <- NA
                }
                sfactor[m] <- sum(x, na.rm = T)
            }
        }
        for (a in 3:ncol(df_norm)) {
            for (b in 1:nrow(df_norm)) {
                df_norm[b, a] <- round(((df_norm[b, a]/median(sfactor)) *
                  1e+06), 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'), 3('paulson'), 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")
  }
```

permubiome *A Permutation Based Test for Biomarker Discovery in Microbiome Data*

Description

All the functions compiled in this package were created to perform permutation-based non-parametric analysis on microbiome data for biomarker discovery aims. This test executes thousands of comparisons in pairwise manner, after random shuffling of data into the different groups of study.

Details

The DESCRIPTION file:

Package:	permubiome
Type:	Package
Title:	A Permutation Based Test for Biomarker Discovery in Microbiome Data
Version:	1.1
Date:	2016-03-22
Author:	Alfonso Benitez-Paez
Maintainer:	Alfonso Benitez-Paez <abenitez@iata.csic.es>
Description:	All the functions compiled in this package were created to perform permutation-based non-parametric analysis
License:	GPL-2
Imports:	ggplot2

Index of help topics:

```
get.data                Parsing the data file.
normalize               Normalize the microbiome dataset prior to
                        perform the permutation test.
permubiome              A Permutation Based Test for Biomarker
                        Discovery in Microbiome Data
permutation             Permutation-based non-parametric analysis to
                        infer differential abundance of features
                        between groups.
plots                   Plotting the features with differential
                        abundance.
```

The permubiome R package was created to perform permutation-based non-parametric analysis on microbiome data for biomaker discovery aims. This test executes thousands of comparisons in pairwise manner, after random shuffling of data into the different groups of study. Previous to the permutation test itself, data can be normalized according to different methods proposed to handle microbiome data ("proportions, "anders", and "paulson"). The median-based differences between groups resulting from the multiple simulations are fitted to normal distribution with the aim to calculate their significance. A multiple testing correction (fdr) is finally applied to extract the differentially presented features between groups of your dataset.

Author(s)

Alfonso Benitez-Paez

Maintainer: Alfonso Benitez-Paez <abenitez@iata.csic.es>

References

Benitez-Paez A. & Sanz Y. (2016). Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. In press.

Examples

```
get.data()
normalize(numz = 0.5, method = 1)
permutation(nperm = 1000, write.output = TRUE)
plots()
```

`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 median differences are calculated for all simulations. Differences calculated from simulations are fitted to the normal distribution and the probability of the observed difference is then calculated. A multiple testing correction is done to accurately discover the biomarker associated with your dataset classes.

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 1000 as the minimum and using values higher than 100000 produces no additional information.
`write.output`	When "T" (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" and "select" parameters prompted.

Author(s)

Alfonso Benitez-Paez

References

Benitez-Paez A. & Sanz Y. (2015). Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. In press.

Examples

```
## The function is currently defined as
function (nperm = 1000, write.output = TRUE)
{
    Class<-NULL
    load("permubiome.RData")
    df_norm <- df_norm
    classes <- levels(df_norm$Class)
    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(classes[1],
        "(median)"), paste(classes[2], "(median)"), "p.value",
        "p.adjust (fdr)")
    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 <- 1 - pnorm(ref_score)
            }
            else {
                pvalue.i <- pnorm(ref_score)
            }
            padjust.i <- p.adjust(pvalue.i, method = "fdr", n = nrow(pvalue_matrix))
            if (pvalue.i <= 1) {
                print(paste(category, signif(pvalue.i, 4), signif(padjust.i,
                    4)))
            }
            else {
                print(paste(category, "1.000", signif(padjust.i,
                    4)))
            }
            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 <= 1) {
                pvalue_matrix[(i - 2), 4] <- format(pvalue.i, digits = 7,
                    scientific = F)
            }
            else {
                pvalue_matrix[(i - 2), 2] <- 1
            }
            if (pvalue.i != 0) {
                pvalue_matrix[(i - 2), 5] <- format(padjust.i, digits = 7,
                    scientific = F)
            }
            invisible()
        }
        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("Features under what level of significance do you want
                to retrieve (i.e. 0.2) : "))
                significant <- subset(pvalue_matrix, pvalue_matrix[,
                    5] <= select)
                ordered <- significant[order(significant[, 5]), ]
            }
            else {
                significant <- subset(pvalue_matrix, pvalue_matrix[,
                    5] <= 1)
                ordered <- significant[order(significant[, 5]), ]
            }
            write.table(ordered, file = "permutation.output", quote = F,
                row.names = F, sep = "\t")
            print(paste("Permutation test done and output table printed!"))
        }
        else {
            significant <- subset(pvalue_matrix, pvalue_matrix[,
                4] <= 1)
            ordered <- significant[order(significant[, 4]), ]
            ordered
            print(paste("Permutation test done!"))
        }
      }
```

`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. & Sanz Y. (2015). Permubiome: an R package to perform permutation based test for biomarker discovery in microbiome analyses. In press.

Examples

```
## The function is currently defined as
function ()
{
    Class<-NULL
    loadNamespace("ggplot2")
    load("permubiome.RData")
    a <- array(, nrow(df_norm))
    for (j in 1:nrow(df_norm)) {
        a[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]/a[m]), digits = 6)
        }
    }
    category <- readline("Type the category you want plotting : ")
    if (category == "") {
        category <- colnames(df_norm[3])
      print(paste("As you declare no categories, the first one of your dataset is plotted!"))
    }
    else {
        ggplot(df_norm, aes(Class, df_norm[, category]), environment = environment()) +
            geom_boxplot(notch = F, outlier.colour = "blue",
                outlier.shape = 1, outlier.size = 3, binaxis = "y",
                stackdir = "center", dotsize = 3) + ggtitle(category) +
            ylab("Normalized read proportion") + xlab("Classes") +
          theme(axis.text = element_text(size = 12), axis.title = element_text(size = 16,
                face = "bold")) + geom_jitter(position = position_jitter(width = 0,
            height = 0))
        output <- readline("Do you want an output file (yes/no)? : ")
        if (substr(output, 1, 1) == "y") {
            extension <- readline("What extension do you prefer fo the output plot
            (ps, pdf, jpeg, tiff, png, bmp )? : ")
            ggsave(filename = paste(category, extension, sep = "."),
                plot = last_plot(), path = NULL, scale = 1, units = c("cm"),
                dpi = 300, limitsize = TRUE)
        }
        else {
            last_plot()
        }
    }
}
```

Index