

# Package ‘lefser’

November 30, 2024

**Type** Package

**Title** R implementation of the LEfSE method for microbiome biomarker discovery

**Description** lefser is the R implementation of the popular microbiome biomarker discovery tool, LEfSe. It uses the Kruskal-Wallis test, Wilcoxon-Rank Sum test, and Linear Discriminant Analysis to find biomarkers from two-level classes (and optional sub-classes).

**Version** 1.17.1

**Date** 2024-10-30

**License** Artistic-2.0

**Depends** SummarizedExperiment, R (>= 4.0.0)

**Imports** coin, MASS, ggplot2, S4Vectors, stats, methods, utils, dplyr, testthat, tibble, tidyr, forcats, stringr, ggtree

**Suggests** knitr, rmarkdown, curatedMetagenomicData, BiocStyle, phyloseq, pkgdown, covr, withr

**Encoding** UTF-8

**BugReports** <https://github.com/waldronlab/lefser/issues>

**URL** <https://github.com/waldronlab/lefser>

**VignetteBuilder** knitr

**biocViews** Software, Sequencing, DifferentialExpression, Microbiome, StatisticalMethod, Classification

**RoxygenNote** 7.3.2

**Roxygen** list(markdown = TRUE)

**git\_url** <https://git.bioconductor.org/packages/lefser>

**git\_branch** devel

**git\_last\_commit** db693db

**git\_last\_commit\_date** 2024-10-30

**Repository** Bioconductor 3.21

**Date/Publication** 2024-11-29

**Author** Sehyun Oh [cre, ctb] (ORCID: <<https://orcid.org/0000-0002-9490-3061>>),  
 Asya Khleborodova [aut],  
 Samuel Gamboa-Tuz [ctb],  
 Marcel Ramos [ctb] (ORCID: <<https://orcid.org/0000-0002-3242-0582>>),  
 Ludwig Geistlinger [ctb] (ORCID:  
 <<https://orcid.org/0000-0002-2495-5464>>),  
 Levi Waldron [ctb] (ORCID: <<https://orcid.org/0000-0003-2725-0694>>)

**Maintainer** Sehyun Oh <shbrief@gmail.com>

## Contents

get_terminal_nodes . . . . .	2
lefser . . . . .	3
lefserClades . . . . .	5
lefserPlot . . . . .	6
lefserPlotClad . . . . .	7
lefserPlotFeat . . . . .	8
relativeAb . . . . .	9
rowNames2RowData . . . . .	9
zeller14 . . . . .	10

**Index** **12**

---

get_terminal_nodes	<i>Identify which elements of a string are terminal nodes</i>
--------------------	---

---

## Description

A terminal node in a taxonomy does not have any child nodes. For example, a species is a terminal node if there are no subspecies or strains that belong to that species. This function identifies which elements of a vector are terminal nodes simply by checking whether that element appears as a substring in any other element of the vector.

## Usage

```
get_terminal_nodes(string)
```

## Arguments

string            A character vector of strings to check for terminal nodes

## Value

A logical vector indicating which elements of the string are terminal nodes

## Examples

```
# What does it do?
data("zeller14")
rownames(zeller14)[988:989]
get_terminal_nodes(rownames(zeller14)[988:989])
# How do I use it to keep only terminal nodes for a lefser analysis?
terminal_nodes <- get_terminal_nodes(rownames(zeller14))
zeller14sub <- zeller14[terminal_nodes, ]
# Then continue with your analysis!
```

---

lefser

*R implementation of the LEfSe method*

---

## Description

Perform a LEfSe analysis: the function carries out differential analysis between two sample classes for multiple features and uses linear discriminant analysis to establish their effect sizes. Subclass information for each class can be incorporated into the analysis (see examples). Features with large differences between two sample classes are identified as biomarkers.

## Usage

```
lefser(
  relab,
  kruskal.threshold = 0.05,
  wilcox.threshold = 0.05,
  lda.threshold = 2,
  classCol = "CLASS",
  subclassCol = NULL,
  assay = 1L,
  trim.names = FALSE,
  checkAbundances = TRUE,
  method = "none",
  ...,
  groupCol,
  blockCol
)
```

## Arguments

**relab** A [SummarizedExperiment](#) with relative abundances in the assay

**kruskal.threshold** numeric(1) The p-value for the Kruskal-Wallis Rank Sum Test (default 0.05). If multiple hypothesis testing is performed, this threshold is applied to corrected p-values.

<code>wilcox.threshold</code>	numeric(1) The p-value for the Wilcoxon Rank-Sum Test when 'subclassCol' is present (default 0.05). If multiple hypothesis testing is performed, this threshold is applied to corrected p-values.
<code>lda.threshold</code>	numeric(1) The effect size threshold (default 2.0).
<code>classCol</code>	character(1) Column name in <code>colData(relab)</code> indicating class, usually a factor with two levels (e.g., <code>c("cases", "controls")</code> ); default "CLASS".
<code>subclassCol</code>	character(1) Optional column name in <code>colData(relab)</code> indicating the subclasses, usually a factor with two levels (e.g., <code>c("adult", "senior")</code> ); default NULL, but can be more than two levels.
<code>assay</code>	The i-th assay matrix in the <code>SummarizedExperiment('relab')</code> ; default 1).
<code>trim.names</code>	Default is FALSE. If TRUE, this function extracts the most specific taxonomic rank of organism.
<code>checkAbundances</code>	logical(1) Whether to check if the assay data in the <code>relab</code> input are relative abundances or counts. If counts are found, a warning will be emitted (default TRUE).
<code>method</code>	Default is "none" as in the original LefSe implementation. Character string of length one, passed on to <code>p.adjust</code> to set option for multiple testing. For multiple pairwise comparisons, each comparison is adjusted separately. Options are "holm", "hochberg", "hommel", "bonferroni", "BH", "BY", "fdr" (synonym for "BH"), and "none".
<code>groupCol</code>	(DEFUNCT) Column name in <code>colData(relab)</code> indicating groups, usually a factor with two levels (e.g., <code>c("cases", "controls")</code> ); default "GROUP".
<code>blockCol</code>	(DEFUNCT) Optional column name in <code>colData(relab)</code> indicating the blocks, usually a factor with two levels (e.g., <code>c("adult", "senior")</code> ); default NULL).
<code>...</code>	Additional inputs to lower level functions (not used).

## Details

The LefSe method expects relative abundances in the `expr` input. A warning will be emitted if the column sums do not result in 1. Use the `relativeAb` helper function to convert the data in the `SummarizedExperiment` to relative abundances. The `checkAbundances` argument enables checking the data for presence of relative abundances and can be turned off by setting the argument to FALSE.

## Value

The function returns a `data.frame` with two columns, which are names of features and their LDA scores.

## Examples

```
data(zeller14)
zeller14 <- zeller14[, zeller14$study_condition != "adenoma"]
tn <- get_terminal_nodes(rownames(zeller14))
zeller14tn <- zeller14[tn,]
```

```
zeller14tn_ra <- relativeAb(zeller14tn)

# (1) Using classes only
res_class <- lefser(zeller14tn_ra,
                  classCol = "study_condition")
# (2) Using classes and sub-classes
res_subclass <- lefser(zeller14tn_ra,
                     classCol = "study_condition",
                     subclassCol = "age_category")
```

---

lefserClades	<i>Run lefser at different clades</i>
--------------	---------------------------------------

---

## Description

lefserClades Agglomerates the features abundance at different taxonomic ranks using `mia::splitByRanks` and performs lefser at each rank. The analysis is run at the species, genus, family, order, class, and phylum levels.

## Usage

```
lefserClades(relab, ...)
```

## Arguments

relab	A (Tree) SummarizedExperiment with full taxonomy in the rowData.
...	Arguments passed to the lefser function.

## Details

When running lefserClades, all features with NAs in the rowData will be dropped. This is to avoid creating artificial clades with NAs.

## Value

An object of class 'lefser\_df\_clades', "lefser\_df", and 'data.frame'.

## Examples

```
data("zeller14")
z14 <- zeller14[, zeller14$study_condition != "adenoma"]
tn <- get_terminal_nodes(rownames(z14))
z14tn <- z14[tn, ]
z14tn_ra <- relativeAb(z14tn)
z14_input <- rowNames2RowData(z14tn_ra)

resCl <- lefserClades(relab = z14_input, classCol = "study_condition")
```

---

`lefserPlot`*Plots results from lefser function*

---

### Description

This function plots the biomarkers found by LEfSe, that are ranked according to their effect sizes and linked to their abundance in each class.

### Usage

```
lefserPlot(  
  df,  
  colors = "c",  
  trim.names = TRUE,  
  title = "",  
  label.font.size = 3  
)
```

### Arguments

<code>df</code>	Data frame produced by <code>lefser</code> . This data frame contains two columns labeled as <code>c("features", "scores")</code> .
<code>colors</code>	Colors corresponding to class 0 and 1. Options: "c" (colorblind), "l" (lefse), "g" (greyscale). Defaults to "c". This argument also accepts a <code>character(2)</code> with two color names.
<code>trim.names</code>	Under the default (TRUE), this function extracts the most specific taxonomic rank of organism.
<code>title</code>	A <code>character(1)</code> . The title of the plot.
<code>label.font.size</code>	A <code>numeric(1)</code> . The font size of the feature labels. The default is 3.

### Value

Function returns plot of effect size scores produced by `lefser`. Positive scores represent the biomarker is more abundant in class '1'. Negative scores represent the biomarker is more abundant in class '0'.

### Examples

```
example("lefser")  
lefserPlot(res_class)
```

---

lefsrPlotClad	<i>LEfSer plot cladogram</i>
---------------	------------------------------

---

## Description

lefsrPlotClad plots a cladogram from the results of lefsrClades.

## Usage

```
lefsrPlotClad(df, colors = "c", showTipLabels = FALSE, showNodeLabels = "p")
```

## Arguments

df	An object of class "lefsr_df_clades".
colors	Colors corresponding to class 0 and 1. Options: "c" (colorblind), "l" (lefsr), "g" (greyscale). Defaults to "c". This argument also accepts a character(2) with two color names.
showTipLabels	Logical. If TRUE, show tip labels. Default is FALSE.
showNodeLabels	Label's to be shown in the tree. Options: "p" = phylum, "c" = class, "o" = order, "f" = family, "g" = genus, "s" = species, "t" = strain. It can accept several options, e.g., c("p", "c").

## Value

A ggtree object.

## Examples

```
data("zeller14")
z14 <- zeller14[, zeller14$study_condition != "adenoma"]
tn <- get_terminal_nodes(rownames(z14))
z14tn <- z14[tn, ]
z14tn_ra <- relativeAb(z14tn)
z14_input <- rowNames2RowData(z14tn_ra)

resCl <- lefsrClades(relab = z14_input, classCol = "study_condition")
ggt <- lefsrPlotClad(df = resCl)
```

---

lefserPlotFeat	<i>Plot Feature</i>
----------------	---------------------

---

### Description

lefserPlotFeat plots the abundance data of a DA feature across all samples.

### Usage

```
lefserPlotFeat(res, fName, colors = "colorblind")
```

### Arguments

res	An object of class lefser_df, output of the lefser function.
fName	A character string. The name of a feature in the lefser_df object.
colors	Colors corresponding to class 0 and 1. Options: "c" (colorblind), "l" (lefse), "g" (greyscale). Defaults to "c". This argument also accepts a character(2) with two color names.

### Details

The solid lines represent the mean by class or by class+subclass (if the subclass variable is present). The dashed lines represent the median by class or by class+subclass (if the subclass variable is present).

### Value

A ggplot object.

### Examples

```
data(zeller14)
zeller14 <- zeller14[, zeller14$study_condition != "adenoma"]
tn <- get_terminal_nodes(rownames(zeller14))
zeller14tn <- zeller14[tn,]
zeller14tn_ra <- relativeAb(zeller14tn)

# (1) Using classes only
res_class <- lefser(zeller14tn_ra,
                  classCol = "study_condition")
# (2) Using classes and sub-classes
res_subclass <- lefser(zeller14tn_ra,
                     classCol = "study_condition",
                     subclassCol = "age_category")
plot_class <- lefserPlotFeat(res_class, res_class$features[[1]])
plot_subclass <- lefserPlotFeat(res_subclass, res_subclass$features[[2]])
```



---

relativeAb	<i>Utility function to calculate relative abundances</i>
------------	--

---

**Description**

The function calculates the column totals and divides each value within the column by the respective column total.

This function calculates the relative abundance of each feature in the SummarizedExperiment object containing count data, expressed as counts per million (CPM)

**Usage**

```
relativeAb(se, assay = 1L)
```

**Arguments**

se	A SummarizedExperiment object with counts
assay	The i-th assay matrix in the SummarizedExperiment ('relab'; default 1).

**Value**

returns a new SummarizedExperiment object with counts per million calculated and added as a new assay named rel\_abs.

**Examples**

```
se <- SummarizedExperiment(  
  assays = list(  
    counts = matrix(  
      rep(1, 4), ncol = 1, dimnames = list(LETTERS[1:4], "SAMP")  
    )  
  )  
)  
assay(se)  
assay(relativeAb(se))
```

---

rowNames2RowData	<i>RowNames to RowData</i>
------------------	----------------------------

---

**Description**

rowNames2RowData transforms the taxonomy stored in the row names to the rowData in a SummarizedExperiment.

**Usage**

```
rowNames2RowData(x)
```

**Arguments**

x                    A SummarizedExperiment with the features taxonomy in the rownames.

**Value**

The same SummarizedExperiment with the taxonomy now in the rowData.

**Examples**

```
data("zeller14")

## Keep only "CRC" and "control" (dichotomous variable)
z14 <- zeller14[, zeller14$study_condition %in% c("control", "CRC")]

## Get terminal nodes
tn <- get_terminal_nodes(rownames(z14))
z14_tn <- z14[tn, ]

## Normalize to relative abundance (also known as Total Sum Scaling)
z14_tn_ra <- relativeAb(z14_tn)

## Add the taxonomy to the rowData
input_se <- rowNames2RowData(z14_tn_ra)
```

---

zeller14

*Example dataset for lefser*

---

**Description**

The ZellerG\_2014 dataset contains microbiome count data for CRC patients and controls. It was for curatedMetagenomicData using the script in the package directory "data-raw".

**Usage**

```
data("zeller14")
```

**Format**

A SummarizedExperiment with 1585 features, 199 samples

**study\_condition** adenoma, control, CRC

**age\_category** adult, senior

**Source**

<https://pubmed.ncbi.nlm.nih.gov/25432777/>

# Index

## \* datasets

zeller14, [10](#)

get\_terminal\_nodes, [2](#)

lefser, [3](#)

lefserClades, [5](#)

lefserPlot, [6](#)

lefserPlotClad, [7](#)

lefserPlotFeat, [8](#)

p.adjust, [4](#)

relativeAb, [9](#)

rowNames2RowData, [9](#)

SummarizedExperiment, [3](#)

zeller14, [10](#)