Package 'RAM'

December 9, 2014

Type Package

Title R for Amplicon-based Metagenomics

Version 1.2.0

Date 2014-12-10

Author Wen Chen and Joshua Simpson

Maintainer Wen Chen <Wen.Chen@agr.gc.ca>

Description This package provides a series of functions to make amplicon-based metagenomic analysis more accessible, and publication-quality plots simple. Amplicon-based (or targeted) metagenomics amplifies and sequences selected DNA regions of environmental samples, but not the entire pool of genetic material, which is referred to as shotgun metagenomics. The amplicon-metagenomics mainly aims at characterizing broad microbiota biodiversity in different environments.

License MIT + file LICENSE

Depends vegan, ggplot2, stats

Imports gridExtra, RColorBrewer, gplots, plyr, reshape2, scales, labdsv, grid, ggmap, permute, VennDiagram, data.table, FD, MASS, RgoogleMaps, ape, lattice, reshape

Suggests testthat, mapproj, gtable, indicspecies, Heatplus

Repository CRAN

URL http://cran.r-project.org/web/packages/RAM/index.html

R topics documented:

`RAM-package` *Analysis of Amplicon-Based Metagenomic Data*

Description

The RAM package provides a series of functions to make amplicon based metagenomic analysis more accessible. The package is designed especially for those who have little or no experience with R. This package calls heavily upon other packages (such as `vegan` and `ggplot2`), but the functions in this package either extend their functionality, or increase the ease-of-use.

Details

Package:	RAM
Type:	Package
Version:	1.2.0
Date:	2014-12-10
License:	MIT License, Copyright (c) 2014 Government of Canada

Load data from .csv-formatted OTU files with `read.OTU` or `fread.OTU`, then process the data with other commands. Type the command `library(help = RAM)` for a full index of all help topics, or `ls("package:RAM")` to get a list of all functions in the package.

Type `data(ITS1, ITS2, meta)` to load sample data sets of RAM, which include the following

data of 16 samples: 1) ITS1: OTU table of fungal internal transcribed spacer region1 3) ITS2: OTU table of fungal internal transcribed spacer region2 3) meta: associated

Type `citation("RAM")` for how to cite this package.

This pacakge contains information licensed under the Open Government Licence - Canada. See `group.spatial` for further details.

Author(s)

Wen Chen and Joshua Simpson.

Maintainer: Wen Chen <wen.chen@agr.gc.ca>

See Also

`vegan`, `ggplot2`

Examples

```
## Not run:
# load data from your own files...
otu1 <- fread.OTU("path/to/OTU/table")
otu2 <- read.OTU("path/to/OTU/table")
meta1 <- fread.meta("path/to/meta/table")
meta2 <- read.meta("path/to/meta/table")

# ...or use the included sample data
data(ITS1, ITS2, meta)
data <- list(ITS1=ITS1, ITS2=ITS2)
dissim.heatmap(ITS1, meta, row.factor=c(City="City"))
dissim.alleig.plot(data)

# type library(help = RAM) to get a full listing of help documents

## End(Not run)
```

`assist.ado` *Perform ADONIS Analysis for OTU Tables Or Taxonomic Abundance Matrix*

Description

This function simplifies ADONIS analysis by abstracting away some of the complexity and returning a list of useful measures.

Usage

```
assist.ado(data, meta, is.OTU=TRUE, ranks=NULL,
           data.trans=NULL, dist=NULL, meta.strata=NULL,
           perm=1000, top=NULL, mode="number")
```

Arguments

data	an OTU table or a taxonomy abundance matrix.
is.OTU	logical. If the data is an OTU table, set is.OTU TRUE; otherwise, set it as FALSE.
meta	the metadata table to be used (must have same samples as data.
ranks	optional. If ranks is not provided, will test for OTUs, otherwise, will test on taxa at defined ranks. If data is a taxonomic abundance matrix, ranks can be NULL
data.trans	optional. Transform the data using method from the function decostand
dist	optional. the name of any method used in vegdist to calculate pairwise distances. See also adonis and vegdist.
meta.strata	optional. A metadata variable within which to constrain permutations. See also adonis
perm	a numeric number of replicate permutations used for the hypothesis test used in adonis.
top	optional. Select the top taxa or OTUs. See also data.revamp
mode	a character vector, one of "percent" or "number". If number, then top groups will be selected based on total sequence count. If percent, then top groups will be selected based on relative abundance. See also data.revamp

Value

This function returns a list containing outputs from adonis test.

- If is.OTU is TRUE and ranks is not given: the output is a length one list named LCA_OTU.
- If is.OTU is TRUE and ranks is given: the output is a list with a length same as the number of taxonomic ranks provided. Each member of the list is named after the rank it processed at.
- If is.OTU is FALSE, the output is a length one list named Taxa.

Author(s)

Wen Chen.

See Also

adonis

Examples

```
data(ITS1, meta)
## Not run:
# test OTUs
data <- list(ITS1=ITS1, ITS2=ITS2)
assist.ado(data, is.OTU=TRUE,meta=meta, ranks=NULL,
           data.trans="log", dist=NULL)

# test taxa at different ranks
```

```
ranks <- c("p", "c", "o", "f", "g")
ado <- assist.ado(data, is.OTU=TRUE, meta=meta, ranks=ranks,
          data.trans="log", dist="bray" )

# test genera
g1 <- tax.abund(data, rank="g", drop.unclassified=TRUE)
data <- list(g1=g1)
assist.ado(data, is.OTU=FALSE, meta=meta, ranks=NULL,
          data.trans="log", dist="bray" )

## End(Not run)
```

`assist.NB` *Negative Binomial Test For OTUID or Taxon*

Description

This function does negative binomial test for a given otuID or taxon

Usage

```
assist.NB(data, meta, is.OTU=TRUE, rank=NULL, meta.factors=NULL,
          anov.fac=NULL, taxon="")
```

Arguments

`data`	an ecology data set to be analyzed.
`meta`	the metadata table to be analyzed.
`is.OTU`	logical. If an OTU table was provided, `is.OTU` should be set as `TRUE`; otherwise, it should be set as `FALSE`.
`rank`	optional. If no `rank` was provided, the data will be used as it is, if `rank` is provided, if data is an OTU table, it will be converted to taxonomic abundance matrix at the given rank, no change will be made for a data that has already been a taxonomic abundance matrix. See also `tax.abund` and `data.revamp`
`meta.factors`	optional. If provided, will only test the model on selected metadata variables; otherwise, will test all variables in the metadata table.
`anov.fac`	optional. Whether or not to do anova test on a metadata variable.
`taxon`	a length one charactor. Can either be an otuID or a taxon name.

Value

This function return a list of outputs of the negative bionomial modeling for a selected otuID or taxa. Members of this output list are: "NB.model", "tax.met", "taxon", "factors", "anova".

`NB.model`	is the negative bionomial model
`tax.met`	is a dataframe with combined the taxon and metadata

taxon	is either a taxon name or in LCA_otuID format, see also LCA.OTU
factors	shows which metadata variable had significant impact
anova	shows anova test of a metadata variable, this will not be available if anov.fac is NULL

Author(s)

Wen Chen

Examples

```
data(ITS1, meta)
m <- meta[, c(2,3,5,7)]
## Not run:
# for usage demonstration purpose only, may not fit the negative
# binomial distribution model.
nb <- assist.NB(ITS1, meta=m, rank="g", anov.fac="Harvestmethod",
              taxon=rownames(ITS1)[1])

## End(Not run)
```

---

assist.ordination *Perform CCA and RDA Analysis for OTU Tables*

Description

This function simplifies CCA and RDA analysis by abstracting away some of the complexity and returning a list of useful measures.

Usage

```
assist.cca(otu1, otu2 = NULL, meta, full = TRUE, exclude = NULL,
           rank, na.action=na.exclude)
assist.rda(otu1, otu2 = NULL, meta, full = TRUE, exclude = NULL,
           rank, na.action=na.exclude)
```

Arguments

otu1	the first OTU table to be used.
otu2	the second OTU table to be used.
meta	the metadata table to be used (must have same samples as otu1/otu2).
full	logical. Should a full model be considered? (If not, a restricted model is used).
exclude	A vector, either numeric or logical, specifying the columns to be removed from meta. If a character vector, columns with those names will be removed; if a numeric vector, columns with those indices will be removed.
rank	a character vector representing a rank. Must be in one of three specific formats (see ?RAM.rank.formatting for help).

na.action	choice of one of the following: "na.fail", "na.omit" or "na.exclude", see na.action in cca for detail.

Value

If both otu1 and otu2 are given, a list of length 2 will be returned with the following items (if only otu1 is given, a list of length 1 will be returned with these items):

$GOF	the goodness of fit scores for the model.
$VIF	the VIF scores for the model.
$percent_variation	the percent variation explained by each axis
$CCA_eig	Eigenvalues for CCA axes.
$CA_eig	Eigenvalues for CA axes.
$anova	the ANOVA results for the model.

Author(s)

Wen Chen and Joshua Simpson.

See Also

cca, anova.cca

Examples

```
data(ITS1, meta)

cca.help <- assist.cca(ITS1, meta=meta, rank="p")
cca.help$anova
```

---

col.splitup *Split Column Of Data Frame*

Description

This function output consumes a data frame and split one by defined separator.

Usage

```
col.splitup(df, col="", sep="", max=NULL, names=NULL, drop=TRUE)
```

Arguments

df	a data frame.
col	name of a column in df.
sep	the separator to split the column. It can be regular expression.
max	optional. The number of columns to be split to.
names	optional. The names for the new columns.
drop	logical. Whether or not to keep the original column to be split in the output.

Value

The value returned by this function is a data frame. The selected column is split each separator and appended to the original data frame. The original column may may not to be kept in the output as defined by option drop.

The number of columns to be split to depends on three factors, 1) the maximum columns that the original column can be split to by each separator; 2) the user definde max; and 3) the length of the column names defined by names. This function will split the column to the maximun number of the 3, empty columns will be filled with empty strings.

Author(s)

Wen Chen.

Examples

```
data(ITS1)

# filter.OTU() returns a list
otu <- filter.OTU(list(ITS1=ITS1), percent=0.001)[[1]]
# split and keep taxonomy column
otu.split <- col.splitup(otu, col="taxonomy", sep="; ",
                         drop=FALSE)
## Not run:
# give new column names
tax.classes <- c("kingdom", "phylum", "class",
                 "order", "family", "genus")
otu.split <- col.splitup(otu, col="taxonomy", sep="; ",
                          drop=TRUE, names=tax.classes)


## End(Not run)
```

combine.OTU *Combine Non Overlapped OTU tables From The Same Community*

Description

This function combines otu tables from the same community but based on independent sequencing runs. Such combined otu table gives a more complete profile of the microbial community than each individual otu table does. This function should NOT be used to combine ITS1 and ITS2 otu tables if they were extracted from long NGS sequences.

Usage

```
combine.OTU(data, meta)
```

Arguments

data	a list of otu tables to be combined.
meta	the metadata that should have the same number and order of the samples as the otu tables do.

Value

combine.OTU returns a data frame of combined otu tables which have the same samples. Samples in the output will match those in the metadata provided.

Author(s)

Wen Chen

See Also

match.data

Examples

```
data(ITS1, ITS2, meta)
meta.new <- head(meta)
## Not run:
# for demonstration purposes only, Not recommend to combine
# ITS1 and ITS2 otu tables that both regions were extracted from
# long NGS sequences
comb <- combine.OTU(data=list(ITS1=ITS1, ITS2=ITS2), meta=meta.new)
stopifnot(identical(colnames(comb)[1:(ncol(comb)-1)],
                    rownames(meta.new)))

## End(Not run)
```

core.OTU *Summary Of Core OTUs*

Description

This function returns a list showing otus that present in a pre-defined percent of samples in each level of a given metadata category.

Usage

```
core.OTU(data, meta, meta.factor="", percent=1)
```

Arguments

data	a list of OTU tables to be analyzed. See RAM.input.formatting.
meta	the metadata table to be analyzed.
meta.factor	the metadata qualitative variable
percent	the percent of samples in each level of the given metadata variable

Value

core.OTU returns a list containing otus that present in a pre-defined percent of samples in each level of a given metadata category. The outputs describe the following information for each level of a given metadata variable: 1) core otuID; 2) taxa the core otus assigned to; and 3) percent of core otus sequences vs. total sequences in each levels of the given metadata variable. The last item in the list show the same information of otus that in all levels.

Note

The OTUs are determined to be absent/present using the "pa" method from the function decostand.

Author(s)

Wen Chen

See Also

decostand

Examples

```
data(ITS1, meta)
## Not run:
core.OTU(data=list(ITS1=ITS1), meta=meta,
         meta.factor="City", percent=0.90)

## End(Not run)
```

core.OTU.rank *Summary Of Core OTUs*

Description

This function returns a list showing otus that present in a pre-defined percent of samples in each level of a given metadata category.

Usage

```
core.OTU.rank(data, rank="g", drop.unclassified=TRUE,
                meta, meta.factor="", percent=1)
```

Arguments

data	a list of OTU tables to be analyzed. See also RAM.input.formatting.
rank	the taxonomic rank(s) of otu classification (see ?RAM.rank.formatting for formatting details).
drop.unclassified	logical, whether or not exclude unclassified groups.
meta	the metadata table to be analyzed.
meta.factor	the metadata qualitative variable
percent	the percent of samples in each level of the given metadata variable

Value

core.OTU.rank returns a list containing otus that present in a pre-defined percent of samples in each level of a given metadata category. The outputs describe the following information for each level of a given metadata variable: 1) core otuID; 2) taxa the core otus assigned to; and 3) percent of core otus sequences vs. total sequences in each group. The last item in the list show the same information of otus that in all levels.

Note

The taxon groups are determined to be absent/present using the "pa" method from the function decostand.

Author(s)

Wen Chen

See Also

decostand

Examples

```
data(ITS1, meta)
## Not run:
core <- core.OTU.rank(data=list(ITS1=ITS1), rank="g", meta=meta,
         meta.factor="City", percent=0.90)

## End(Not run)
```

`core.Taxa` *Show Summary of Core Taxa*

Description

This function returns a list showing taxa at the given taxonomic rank that present in a pre-defined percent of samples in each level of a given metadata category.

Usage

```
core.Taxa(data, is.OTU=FALSE, rank="g",
         drop.unclassified=TRUE,
         meta, meta.factor="", percent=1)
```

Arguments

`data`	a list of OTU tables or taxonomy abundance matrices to be analyzed.
`is.OTU`	logical. If `TRUE`, data is an OTU table; otherwise a taxonomy abundance matrix should be provided.
`rank`	the taxonomic rank of classification (see ?RAM.rank.formatting for formatting details).
`drop.unclassified`	logical, whether or not exclude unclassified groups. See also `tax.abund`
`meta`	the metadata table to be analyzed.
`meta.factor`	the metadata qualitative variable
`percent`	the percent of samples in each level of the given metadata variable

Value

`core.Taxa` returns a list containing taxa at a given rank that present in a pre-defined percent of samples in each level of a given metadata category. The outputs describe the following information for each level of a given metadata variable: 1) core taxa; 2) percent of core taxa sequences vs. total sequences in each levels of the given metadata variable. The last item in the list show the same information of taxa that in all levels.

Note

The taxa are determined to be absent/present using the "pa" method from the function `decostand`.

Author(s)

Wen Chen

See Also

decostand

Examples

```
data(ITS1, meta)
# taxa shared by 50 percent samples of each city
core <- core.Taxa(data=list(ITS1=ITS1), is.OTU=TRUE, meta=meta,
                  rank="g", meta.factor="City", percent=0.5)
## Not run:
data(ITS1, ITS2, meta)
core <- core.Taxa(data=list(ITS1=ITS1, ITS2=ITS2), is.OTU=TRUE,
                  meta=meta, rank="g", meta.factor="City",
                  percent=0.7)

# use taxonomy abundance matrix
g1<-tax.abund(ITS1, rank="g")
core <- core.Taxa(data=list(genus_ITS1=g1), is.OTU=FALSE, meta=meta,
                  rank="g", meta.factor="City", percent=0.9)

## End(Not run)
```

---

correlation *Plot Of Correlation Coefficient*

Description

This function plot correlation relationship among taxa at a give rank and / or numeric variables of metadata.

Usage

```
correlation(data=NULL, is.OTU=TRUE, meta=NULL, rank="g",
            sel=NULL, sel.OTU=TRUE, data.trans=NULL,
            method="pearson", main=NULL, file=NULL,
            ext=NULL, width=8, height=8)
```

Arguments

data	a data frame that either an OTU table or taxonomy abundance matrix, can be missing but if metadata is also missing, an error message will be raised.
is.OTU	logical. Whether or not the data is an OTU table.
meta	the metadata table to be used.

rank	the taxonomic rank to use (see ?RAM.rank.formatting for formatting details).
sel	optional. It is a character vector of selected otuIDs or taxa names at a given taxonomic rank. If provided, sel.OTU should be set to decribe the type of IDs, i.e. TRUE means otuIDs, FALSE means taxa names. If provide, only the selected taxa will be ploted; otherwise, all taxa will be ploted.
sel.OTU	logical. Whether or not the selected items from data are otuIDs. If FALSE, sel should be a string vector of taxa names at a given rank.
data.trans	a character string of one of the following, "total", "log", "hellinger" etc, see ?vegan::decostand for details and other data transformation methods.
method	a character string, can be one of the following, "pearson", "kendall", "spearman" for the calculation of correlation coefficient (or covariance) is to be computed (see ?stats::cor for details)
main	a character string. The title of the plot.
file	the file path where the image should be created (see ?RAM.plotting).
ext	filename extension, the type of image to be saved to. (see ?RAM.plotting).
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).

Details

This function uses stats::cor to calculate correlation coefficient (or covariance), and uses lattice::levelplot to generate the graph. (see References)

Option sel is optional, however, it raises an error if the total number of variables to be plotted was too big, and no plot will be generated.

Value

This function generates a graph showing correlation relationship among OTUs or taxa at a given rank, and numeric variables of metadata

Author(s)

Wen Chen.

References

Sarkar, Deepayan (2008) _Lattice: Multivariate Data Visualization with R_, Springer. <URL: http://lmdvr.r-forge.r-project.org/>

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) _The New S Language_. Wadsworth & Brooks/Cole.

See Also

cor levelplot

Examples

```
data(ITS1, meta)

# only plot the first 10 OTUs
sel <- rownames(ITS1)[1:10]
correlation(data=ITS1, meta=meta, is.OTU=TRUE, sel.OTU=TRUE,
            sel=sel)
## Not run:
sel <- c("Fusarium", "Cladosporium", "Alternaria")
correlation(data=ITS1, meta=meta, is.OTU=TRUE, sel.OTU=FALSE,
            sel=sel, rank="g", data.trans="total",
            file="test.pdf", ext="pdf")

## End(Not run)
```

---

data.clust *Plot Hierarchical Cluster Of Samples Based on OTU Table or Taxonomic Abundance Matrix*

Description

This function plot hierarchical cluster Of ecology data set.

Usage

```
data.clust(data, is.OTU=TRUE, meta, rank=NULL, top=NULL,
           mode="number", group=4, data.trans=NULL,
           dist=NULL, clust=NULL, type=NULL, main=NULL,
           file=NULL, ext=NULL, width=8, height=8)
```

Arguments

data	an ecology data set to be analyzed.
is.OTU	logical. If an OTU table was provided, is.OTU should be set as TRUE; otherwise, it should be set as FALSE.
meta	the metadata table associated with ecology data set.
rank	optional. If no rank was provided, the data will be used as it is, if rank is provided, if data is an OTU table, it will be converted to taxonomic abundance matrix at the given rank, no change will be made for a data that has already been a taxonomic abundance matrix. See also tax.abund and data.revamp
top	the top otuIDs or taxa to be considered for the clustering analysis. See also data.revamp
mode	either be "number" or "percent". See also data.revamp
group	an integer or a metadata variable. If an integar, will cut tree into corresponding groups and color them accordingly; if a metadata variable was provided, tree leaves (sampleIDs) will be colored by each level.

data.trans	optional. If was provided, numeric data will be transformed. See also decostand
dist	optional. If was provided, distance matrix will be calculated using the given method; otherwise use vegdist default Bray-Curtis method. See also vegdist and gowdis.
clust	optional. If was not provided, will use the default agglomeration method used by hclust, i.e. "complete". Otherwise, will used user defined method for clustering. See also hclust.
type	optional. Can be one of the following: "triangle", "rectangle", "phylogram", "cladogram", "fan", "unrooted", "radial".
main	The title of the plot.
file	optional. Filename that the plot to be saved to.
ext	optional. File type that the plot to be saved to.
width	an integer, width of the plot.
height	an integer, height of the plot.

Value

This function returns a tree plot of the hierarchical cluster of the samples based on ecological data.

Author(s)

Wen Chen

Examples

```
data(ITS1, meta)
## Not run:
data.clust(data=ITS1, is.OTU=TRUE, data.trans="total", dist="bray",
           type="fan", meta=meta, group="Plots")

## End(Not run)
```

---

data.revamp *Transform OTU Table*

Description

This function consumes and transforms either an OTU table or a taxonomy abundance matrix. If an OTU table was provided, it will be either transposed without the "taxonomy" column, but each otuID will be renames with it's LCA classification appended; or being transformed to be taxonomic abundance matrix at the ranks set by ranks. If a taxonomic abundance matrix is provided, it will be kept the same with proper data transformation as defined by stand.method option.

Usage

```
data.revamp(data, is.OTU=TRUE, ranks=NULL, stand.method=NULL,
            top=NULL, mode="number")
```

Arguments

data	an OTU table or a taxonomic abundance matrix.
is.OTU	logical. If an OTU table was provided, is.OTU should be set as TRUE; otherwise, it should be set as FALSE.
ranks	optional. If no ranks was provided, the OTU table will be processed by LCA.OTU and then transposed with sampleIDs being row names and otuIDs being column names. If ranks was provided, the OTU table will be processed by tax.abund at each given taxonomic ranks. See also RAM.rank.formatting. The unclassified taxon groups are removed.
stand.method	optional. Transform the output using method from the function decostand
top	optional. Select the top taxa or OTUs.
mode	a character vector, one of "percent" or "number". If number, then top many groups will be selected. If percent, then all groups with relative abundance in at least one sample above top will be selected.

Value

The value returned by this function is a list, so for convenience, any nested lists have been given descriptive items names to make accessing its elements simple (see Examples).

- If is.OTU is TRUE and ranks is not given: the output is a length one list named LCA_OTU.
- If is.OTU is TRUE and ranks is given: the output is a list with a length same as the number of taxonomic ranks provided. Each member of the list is named after the rank it processed at.
- If is.OTU is FALSE, the output is a length one list named Taxa.

Author(s)

Wen Chen

Examples

```
data(ITS1, ITS2, meta)
data.new <- data.revamp(data=list(ITS1=ITS1), is.OTU=TRUE,
                        ranks=c("f", "g"), stand.method="log")
## Not run:
data.new <- data.revamp(data=list(ITS1=ITS1), is.OTU=TRUE,
                        ranks=NULL, stand.method="log")
data.new <- data.revamp(data=list(ITS1=ITS1, ITS2=ITS2),
           is.OTU=TRUE, ranks=c("f", "g"), stand.method="total")
names(data.new)

## End(Not run)
```

`data.subset` *Subset OTU And Metadata*

Description

This function subset OTUs and metadata based on user defined values of metadata variables.

Usage

```
data.subset(data, meta, factors="", values="", and=TRUE)
```

Arguments

`data`	a list of otu tables to be processed. See also `RAM.input.formatting`.
`meta`	the metadata for subset.
`factors`	a vector containing metadata variables.
`values`	a vector containing values of interest in metadata variables.
`and`	logical. Determine whether all conditions needs to be met or not.

Value

The value returned by this function is a list containing otu tables matching the filtering requirement. The last item in the output list is the associated new metadata table fit the requirement.

Author(s)

Wen Chen

Examples

```
data(ITS1, ITS2, meta)
names(meta)
factors <- c("City", "Harvestmethod")
values <- c("City1", "Method1")

# match all requirements, and=TRUE
sub <- data.subset(data=list(ITS1=ITS1, ITS2=ITS2), meta=meta,
                   factors=factors, values=values, and=TRUE)

# match either of the requirements, and=FALSE
sub <- data.subset(data=list(ITS1=ITS1, ITS2=ITS2), meta=meta,
                   factors=factors, values=values, and=FALSE)
## Not run:
names(sub)
ITS1.sub <- sub[["ITS1"]]
ITS2.sub <- sub[["ITS2"]]
meta.sub <- sub[["meta"]]

## End(Not run)
```

dissim *Calculate Dissimilarity Matrix Data*

Description

These functions calculate different measures related to dissimilarity matrices. All of these functions allow you to specify one of many dissimilarity indices to be used.

Usage

```
dissim.clust(elem, is.OTU=TRUE, stand.method=NULL,
             dist.method="morisita", clust.method="average")
dissim.eig(elem, is.OTU=TRUE, stand.method=NULL,
           dist.method="morisita")
dissim.ord(elem, is.OTU=TRUE, stand.method=NULL,
           dist.method="morisita", k=NULL)
dissim.GOF(elem, is.OTU=TRUE, stand.method=NULL,
           dist.method="morisita")
dissim.tree(elem, is.OTU=TRUE, stand.method=NULL,
           dist.method="morisita", clust.method="average")
dissim.pvar(elem, is.OTU=TRUE, stand.method=NULL,
            dist.method="morisita")
```

Arguments

elem	an ecology data set that can be an OTU table or a taxonomy abundance table. See RAM.input.formatting for details.
is.OTU	logical, whether the ecology data sets are OTU tables or taxonomy abundance matrices. See RAM.input.formatting for details.
stand.method	optional, if is.null, the standardization method for data transforamtion; must be one of the following: "total", "max", "frequency", "normalize", "range", "standardize", "pa", "chi.square", "hellinger", "log". See also decostand.
dist.method	the dissimilarity index to be used; one of "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower","morisita", "horn", "mountford", "raup", "binomial", "chao", or "cao". See also vegdist.
k	the number of dimensions desired. If NULL, the maximum value will be calculated and used.
clust.method	the method used for clustering the data. Must be one of "ward", "single", "complete", "average", "mcquitty", "median", or "centroid". See also hclust.

Value

dissim.clust	returns a hierarchical clustering of the dissimilarity matrix.
dist.eigenval	returns the eigenvalues of the dissimilarity matrix.

dissim.ord	returns a list: the first item is the the ordination distances, the second is the dissimilarity matrix distances.
dissim.GOF	returns the goodness of fit values of the dissimilarity matrix, for various numbers of dimensions used.
dissim.tree	returns a list: the first item is the tree distances, the second is the dissimilarity matrix distances.
dissim.pvar	returns a numeric vector containing the percent variation explained by each axis (where each sample corresponds to an axis).

Author(s)

Wen Chen and Joshua Simpson

See Also

decostand, vegdist, hclust, dissim.plot

Examples

```
data(ITS1)

# calculate clustering, using default method
dissim.clust(ITS1)

# calculate tree distances, specifying a distance method
# (but use default clustering method)
dissim.tree(ITS1, dist.method="euclidean")

# calcualte ordination distances, specifying both distance
# and ordination methods
dissim.ord(ITS1, dist.method="bray", k=3)
```

---

dissim.heatmap *Plot Distance Matrix Heatmap for OTU Samples*

Description

This function consumes an OTU table, along with some optional parameters, and creates a heatmap showing the distance matrix for the samples of the given OTU table.

Usage

```
dissim.heatmap(data, is.OTU=TRUE, meta=NULL, row.factor=NULL,
               col.factor=NULL, stand.method="chi.square",
               dissim.method="euclidean",
               file=NULL, ext=NULL, height=8, width=9,
               leg.x=-0.05, leg.y=0)
```

Arguments

data	the OTU table to be used.
is.OTU	logical. Whether or not the data is an OTU table.
meta	the metadata table to be used.
row.factor	a factor from the metadata to show along the rows of the heatmap (see Details below).
col.factor	a factor from the metadata to show along the columns of the heatmap (see Details below).
stand.method	a method used to standardize the OTU table. One of "total", "max", "freq", "normalize", "range", "standardize", "pa", "chi.square", "hellinger" or "log" (see ?decostand).
dissim.method	the dissimilarity index to be used; one of "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower","morisita", "horn", "mountford", "raup", "binomial","chao", or "cao" (see ?vegdist).
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).
leg.x	how far the legend should be inset, on the x axis.
leg.y	how far the legend should be inset, on the y axis.

Details

Both row.factor and col.factor should be named character vectors specifying the names of the rows to be used from meta (see RAM.factors). They should also be factors; if they are not, a warning is raised and they are coerced to factors (see factor). A warning is also raised when a factor has more than 8 levels, as that is the most colours the current palettes support. The factor must also correspond to the OTU table; i.e. they should have the same samples. If not, an error is raised.

Note

This function creates the heatmap using the heatmap.2 function from the gplots package. That function calls layout to set up the plotting environment, which currently prevents plotting two data sets side by side, or to automatically place the (metadata) legend. Unfortunately, this means that the leg.x and leg.y parameters must be used to adjust the legend by trial and error. It is possible to move the legend outside of the plotting area; if not legend appears, try with small leg.x and leg.y values.

Author(s)

Wen Chen and Joshua Simpson.

See Also

decostand, vegdist, factor, heatmap.2, RAM.factors

Examples

```
data(ITS1, meta)

# plot to the screen with one meta factor and standard
# calculation methods
dissim.heatmap(ITS1, is.OTU=TRUE, meta=meta, row.factor=c(Plot="Plots"))

## Not run:
# plot the heatmap to a .tiff file using Hellinger standardization
# and Manhattan distances
dissim.heatmap(ITS1, dissim.method="manhattan",
               stand.method="hellinger",
               file="my/sample/path", ext="tiff")
## End(Not run)
```

---

dissim.plot *Plot Dissimilarity Matrix Data for Different Methods*

Description

These functions all produce a plot of some measure related to dissimilarity matrices. All of these functions allow you to specify a vector of methods to be used when creating the plot.

Usage

```
dissim.clust.plot(data, is.OTU=TRUE, stand.method=NULL,
                  dist.methods=NULL,
                  clust.methods=NULL, file=NULL)
dissim.eig.plot(data, is.OTU=TRUE, stand.method=NULL,
                dist.methods=NULL, file=NULL)
dissim.alleig.plot(data, is.OTU=TRUE, stand.method=NULL,
                   dist.methods=NULL, file=NULL)
dissim.ord.plot(data, is.OTU=TRUE, stand.method=NULL,
                dist.methods=NULL, k=NULL, file=NULL)
dissim.GOF.plot(data, is.OTU=TRUE, stand.method=NULL,
                dist.methods=NULL, file=NULL)
dissim.tree.plot(data, is.OTU=TRUE, stand.method=NULL,
                 dist.methods=NULL,
                 clust.methods=NULL, file=NULL)
dissim.pvar.plot(data, is.OTU=TRUE, stand.method=NULL,
                 dist.methods=NULL, file=NULL)
```

Arguments

data	a list of ecology data. See also RAM.input.formatting
is.OTU	logical, whether the ecology data sets are OTU tables or taxonomy abundance matrices.

`stand.method` optional, if `is.null`, the standardization method for data transforamtion; must be one of the following: "total", "max", "frequency", "normalize", "range", "standardize", "pa", "chi.square", "hellinger", "log". See also `decostand`.

`dist.methods` a character vector representing the dissimilarity indices to be used; each element must be one of one of "`manhattan`", "`euclidean`", "`canberra`", "`bray`", "`kulczynski`", "`jaccard`", "`gower`", "`altGower`","`morisita`", "`horn`", "`mountford`", "`raup`", "`binomial`","`chao`", or "`cao`".

`clust.methods` a character vector representing the methods used for clustering the data. Each element must be one of "`ward`", "`single`", "`complete`", "`average`", "`mcquitty`", "`median`", or "`centroid`".

`k` the number of dimensions desired. If `NULL`, the maximum value will be calculated and used.

`file` the file path for the plot. If not provided (defaults to NULL), then the plot is displayed to the screen. If `file` is provided, that is where the .tiff file will be created.

Details

All of these functions (other than `dissim.alleig.plot`) call `dissim.X` counterparts and plot the data. If `file` is given, a `.tiff` file will be created at `file`; otherwise the plot is displayed to the screen.

Value

All functions create a plot and return the plotted data invisibly.

`dissim.clust.plot`
: plots a hierarchical clustering of the dissimilarity matrix.

`dissim.eig.plot`
: plots a bar plot of the eigenvalues of the dissimilarity matrix.

`dissim.alleig.plot`
: plots a line plot showing the relative importance of all eigenvalues for a variety of methods.

`dissim.ord.plot`
: plots a scatter plot comparing the "euclidean" distances among all samples in ordination space to the dissimilarity matrix distances.

`dissim.GOF.plot`
: plots a scatter plot of the goodness of fit values of the dissimilarity matrix, for various numbers of dimensions used.

`dissim.tree.plot`
: plots a scatter plot comparing the tree distances to the dissimilarity matrix distances.

`dissim.pvar.plot`
: plots a bar plot showing the percent variation explained by each axis (where each sample corresponds to an axis).

Note

If `file` does not end in ".tiff", then ".tiff" will be appended to the end of `file`.

Function `dissim.alleig.plot` uses the `ggplot2` package for creating the plot, and returns the plot object. This means that you can store this plot and add other features manually, if desired (see Examples).

Author(s)

Wen Chen and Joshua Simpson

See Also

`vegdist`, `hclust`, `dissim`, `ggplot`

Examples

```
data(ITS1, ITS2)
data <- list(ITS1=ITS1, ITS2=ITS2)
# show percent variation for only ITS1 with default methods
dissim.pvar.plot(data=list(ITS1=ITS1))


## Not run:
# show clustering for ITS1 and ITS2 for set methods
dissim.clust.plot(data=data, is.OTU=TRUE, stand.method=NULL,
                  dist.methods=c("morisita", "bray"),
                  clust.methods=c("average", "centroid"))
dissim.ord.plot(data=data, is.OTU=TRUE, stand.method="total",
                dist.method="bray")
# dissim.alleig.plot returns a ggplot2 object:
my.eig.plot <- dissim.alleig.plot(data)
class(my.eig.plot) # returns "gg" "ggplot"
my.eig.plot # view the plot
# update the title, then view the updated plot
my.eig.plot <- my.eig.plot + ggtitle("My New Title")
# update ggplot theme
require("grid")
new_theme <-RAM.color()
my.eig.plot <- my.eig.plot + new_theme
my.eig.plot

# save an image (named file.pdf) with GOF values for ITS1 and ITS2,
using # default methods
dissim.GOF.plot(data=data, file="~/Documents/my/file")

## End(Not run)
```

`diversity.indices` *Calculate True Diversity and Evenness*

Description

These functions calculate true diversity and evenness for all samples.

Usage

```
true.diversity(data, index = "simpson")
evenness(data, index = "simpson")
```

Arguments

`data`	a list of otu tables to be processed. See `RAM.input.formatting`.
`index`	the index to use for calculations; partial match to "`simpson`" or "`shannon`".

Details

For the following sections, S represents the number of species, λ represents the Simpson index, and H' represents the Shannon index.

The formulas for the true diversity of the indices are as follows:

- Simpson: $D_2 = frac1\lambda$
- Shannon: $D_1 = \exp H'$

The formulas for the evenness of the indices are as follows:

- Simpson: $\frac{\frac{1}{\lambda}}{S}$
- Shannon: $\frac{H'}{\ln S}$

Value

Both functions return a numeric data frame, where the rows are the given OTUs, and the columns are the samples.

Note

Credit goes to package `vegan` for the partial argument matching (see References).

Author(s)

Wen Chen and Joshua Simpson.

References

Jari Oksanen, F. Guillaume Blanchet, Roeland Kindt, Pierre Legendre, Peter R. Minchin, R. B. O'Hara, Gavin L. Simpson, Peter Solymos, M. Henry H. Stevens and Helene Wagner (2013). vegan: Community Ecology Package. R package version 2.0-10. http://CRAN.R-project.org/package=vegan

Diversity index. (2014, May 7). In Wikipedia, The Free Encyclopedia. Retrieved 14:57, May 28, 2014, from http://en.wikipedia.org/w/index.php?title=Diversity_index&oldid=607510424

Blackwood, C. B., Hudleston, D., Zak, D. R., & Buyer, J. S. (2007). Interpreting ecological diversity indices applied to terminal restriction fragment length polymorphism data: insights from simulated microbial communities. Applied and Environmental Microbiology, 73(16), 5276-5283.

Examples

```
data(ITS1, ITS2)

# true diversity, using default index (Simpson)
true.diversity(data=list(ITS1=ITS1))

# true diversity for ITS1 and ITS2, using Shannon
true.diversity(data=list(ITS1=ITS1, ITS2=ITS2), index="shannon")

# default evenness (Simpson) for ITS1/ITS2
evenness(data=list(ITS1=ITS1, ITS2=ITS2))

# Shannon evenness
evenness(data=list(ITS1=ITS1), index="shannon")
```

---

envis.NB *Visualize The Negative Binomial Model OF A Given Taxon OR OTUID*

Description

This function plot the negative binomial model for a given otuID or taxon

Usage

```
envis.NB(NB.model="", tax.meta,  taxon="",
         x="", num.col=NULL, group=NULL, group.order=NULL,
         xlab=NULL, ylab=NULL, fill=NULL, facet=NULL,
         file=NULL, ext=NULL, width=8, height=8)
```

Arguments

NB.model	the negative binomial model. Can be obtained by using assist.NB
tax.meta	the combined taxon/otuID and metadata. Can be obtained by using assist.NB
taxon	the taxon or otuID. Can be obtained by using assist.NB
x	a metadata variable name for x axis.

num.col	optional. A metadata numerical variable that will be used as predictor.
group	optional. A metadata factor variable that will be used as predictor.
group.order	optional. The desired order for the group.
xlab	optional. X axis label.
ylab	optional. Y axis label.
fill	optional. Color for fill different categories.
facet	optional. Metadata category variables as faceting variables.
file	optional. Filename that the plot to be saved to.
ext	optional. Filename extension, type of image to be saved.
width	an integer. Filter OTU table by counts.
height	an integer. Filter OTU table by counts.

Value

This function plot predicted taxon/otuID under the impact of metadata variables.

Author(s)

Wen Chen

Examples

```
data(ITS1, meta)

# filter otu table
its1 <- filter.OTU(data=list(ITS1=ITS1), percent=0.01)[[1]]
m <- meta[, c(2,3,5,7)]

## Not run:
# test the model
nb <- assist.NB(its1, meta=m, rank="g", anov.fac="Harvestmethod",
              taxon=rownames(its1)[1])

NB.model<-nb[[1]]
tax.meta<-nb[[2]]
taxon<-nb[[3]]

envis.NB(NB.model=NB.model, tax.meta=tax.meta,  taxon=taxon,
                x="Ergosterol_ppm", num.col="Ergosterol_ppm",
                group="Crop", group.order=NULL,
                xlab="Ergosterol (ppm)", ylab=NULL,
                fill="Harvestmethod", facet=c("City","Crop"))

## End(Not run)
```

`filter.META` *Select METADATA Variables*

Description

This function will remove metadata variables with only one level and /or remove variables with missing data or neither numeric nor factor/character (NNF).

Usage

```
filter.META(meta=meta, excl.na=TRUE, excl.NNF=TRUE, exclude=NULL)
```

Arguments

`meta`	the metadata table to be analyzed.
`excl.na`	logical. Whether or not remove variables that contain missing data.
`excl.NNF`	logical. Whether or not remove variables that neither are numeric nor factor/character.
`exclude`	optional. If is `NULL`, the function only removes variables with only one level or NNF. Otherwise, the variables in the `exclude` will also be removed from the metadata table.

Value

The value returned by this function is a data frame with the following metadata variables being removed: 1) with missing data; 2) NNF if `excl.NNF` is `TRUE`; and 3) in the `exclude` list.

Author(s)

Wen Chen

Examples

```
data(meta)
## Not run:
# add a new column with NA
meta.nw <- meta
meta.nw$na <- c(rep(NA, nrow(meta.nw)-3), c(1, 3, 5))
meta.nw$nf <- rep("Canada", nrow(meta.nw))

meta.fil <- filter.META(meta.nw)
meta.fil <- filter.META(meta.nw, excl.na=FALSE, excl.NNF=FALSE,
                        exclude=c("Province", "Latitude"))

## End(Not run)
```

`filter.OTU` *Filter OTU*

Description

This function filter OTU table by counts or relative abundance. If filter by counts, otus having total counts more than a threshhold will be kept. If filter by relative abundance, otus with the maximum relative abundance greater than a threshhold in at least one subject will be kept.

Usage

```
filter.OTU(data, percent=NULL, number=NULL)
```

Arguments

`data`	a list of otu tables to be processed. See also `RAM.input.formatting`
`percent`	a floating point greater than 0 and less or equals to 1. Filter OTU table by relative abundance.
`number`	an integer. Filter OTU table by counts.

Value

The value returned by this function is a list of filtered otu tables provided by the user

Author(s)

Wen Chen

Examples

```
data(ITS1, ITS2, meta)
data<-list(ITS1=ITS1, ITS2=ITS2)
## Not run:
otu.001 <- filter.OTU(data=data, percent=0.01)
length(otu.001)
names(otu.001)
otu.50 <- filter.OTU(data=data, number=50)

## End(Not run)
```

`filter.Taxa` *Filter Taxonomic Abundance Matrix by Total Counts Or Maximum Relative Abundance*

Description

This function filter taxa group by counts or relative abundance. If filter by counts, taxa having total counts more than a threshhold will be kept. If filter by relative abundance, taxa with the maximum relative abundance greater than a threshhold in at least one subject will be kept.

Usage

```
filter.Taxa(taxa, drop.unclassified=TRUE,
            percent=NULL, number=NULL)
```

Arguments

`taxa`	the taxonomy abundance matrix: sample x species data frame. See also `tax.abund`
`drop.unclassified`	logical, whether or not remove unclassified groups. See also `tax.abund`
`percent`	a floating point greater than 0 and less or equals to 1. Filter Taxa table by relative abundance.
`number`	an integer. FilterTaxa table by total sequence counts.

Value

The value returned by this function is a data frame with taxa met the filter requirement only.

Author(s)

Wen Chen

Examples

```
data(ITS1)
g1 <- tax.abund(ITS1, rank="g", drop.unclassified=TRUE)
taxa.fil <- filter.Taxa(g1, percent=0.01)
```

fread.meta *Load Metadata Table*

Description

This function is same as read.meta to read in data; except using fread for loading large data sets.

Usage

```
fread.meta(file, sep="auto")
```

Arguments

file	a character vector specifying the file path to your file.
sep	the character used to separate cells in the file.

Value

Returns a data frame with the information from the file. The first row and column are used for the names of the other rows and columns.

Author(s)

Wen Chen

See Also

read.meta, fread

Examples

```
## Not run:
  my.meta <- fread.meta("path/to/meta")

## End(Not run)
```

fread.OTU *Fast Load Large OTU Table*

Description

This function is same as read.OTU except using fread for loading large data sets.

Usage

```
fread.OTU(file, sep="auto")
```

Arguments

file	a character vector specifying the file path to your file.
sep	the character used to separate cells in the file.

Value

Returns a data frame with the information from the file. The first row and column are used for the names of the other rows and columns.

Note

The OTU table should only contain classifications for the seven major taxonomic ranks, additional ranks will break some functions in the package. To remove those other classifications, try running

```
sed -i.backup -e 's/s[a-z]__[^;]*; //g' -e 's/t__[^;]*; //g' $FILE
```

where `$FILE` is your OTU table. The letter `t` can be replaced in the second expression for any other letter which appears as a prefix. For example, adding `-e 's/u__[^;]*; //g'` before `$FILE` would remove any entries formatted like `u__test_classification; .`

Additionally, if your OTU table starts with the entry `#OTU ID`, that cell needs to be removed before the table can be read with `fread.OTU`.

Author(s)

Wen Chen.

See Also

`read.OTU`, `fread`

Examples

```
## Not run:
  my.OTU <- fread.OTU("path/to/otu")

## End(Not run)
```

---

get.rank *Get OTUs Classified at Taxonomic Rank(s)*

Description

This function returns the OTUs of the given OTU table(s) which are classified at the given taxonomic rank.

Usage

```
get.rank(otu1, otu2 = NULL, rank = NULL)
```

Arguments

otu1	the first OTU table to be used.
otu2	the second OTU table to be used.
rank	a character vector representing a rank. Must be in one of three specific formats (see RAM.rank.formatting for help). If omitted, the function will repeat for all seven major taxonomic ranks.

Value

The value returned by this function may become nested lists, so for convenience, any nested lists have been given descriptive items names to make accessing its elements simple (see Examples).

- If otu2 is given:
  - If rank is given: a list containing two data frames (otu1 and otu2 selected at the given rank).
  - If rank is not given: a list containing two lists. The first sublist represents otu1, the second otu2. The sublists contain seven data frames, which are the OTU tables selected at each taxonomic rank (see Examples).
- If otu2 is not given:
  - If rank is given: a single data frame (otu1 selected at the given rank).
  - If rank is not given: a list containing seven data frames (otu1 selected at each taxonomic rank).

Author(s)

Wen Chen and Joshua Simpson.

Examples

```
data(ITS1, ITS2)

# the following are equivalent:
ITS1.p <- get.rank(ITS1, rank="p")
# this list has get.rank(ITS1, rank="k"),
#               get.rank(ITS1, rank="p"), ...
lst <- get.rank(ITS1)
stopifnot(identical(ITS1.p, lst$phylum))
# true

# get a list of length 2: the item holds all ITS1 data, the
# second holds ITS2 data
lst.all <- get.rank(ITS1, ITS2)
stopifnot(identical(ITS1.p, lst.all$otu1$phylum))
```

group.abund.Taxa *Barplot Of Distribution Of Taxa In Groups*

Description

This function do a barplot to show the distribution of selected taxa in each level of a given metadata variable

Usage

```
group.abund.Taxa(data, is.OTU=TRUE, rank="g", taxa,
                 drop.unclassified=FALSE, bar.width=NULL,
                 meta, meta.factor="", RAM.theme=NULL,
                 col.pal=NULL, main="",
                 file=NULL, ext=NULL, height=8, width=16)
```

Arguments

data	a list of otu tables or taxonomic abundance matrices. See also RAM.input.formatting.
is.OTU	logical. If an OTU table was provided, is.OTU should be set as TRUE; otherwise, it should be set as FALSE.
rank	a single taxonomic rank. See also RAM.rank.formatting
taxa	a vector containing taxa names for plotting.
drop.unclassified	logical. Whether or not drop the unclassified taxon groups.
bar.width	width of bars
meta	the metadata table to be used (must have same samples as data.
meta.factor	a character string. Must be one of the metadata variables.
RAM.theme	customized ggplot_theme in RAM. See also ?theme_ggplot.
col.pal	color palettes to be used.
main	a character string. The title of the plot, default is an empty string.
file	filename to save the plot.
ext	filename extension, the type of image to be saved to.
width	an integer, width of the plot.
height	an integer, height of the plot.

Value

This function returns a Barplot of the distribution of seleted taxa within each level of a given metadata variable.

Note

This funtion provide an alternative view of taxa distribution as group.Taxa.bar.

Author(s)

Wen Chen.

Examples

```
data(ITS1, ITS2, meta)
taxa <- c("Fusarium", "Alternaria", "Cladosporium")
group.abund.Taxa(data=list(ITS1=ITS1, ITS2=ITS2), taxa=taxa,
                 meta=meta, meta.factor="Crop",
                 drop.unclassified=TRUE)
```

---

group.abundance *Plot the Abundance of OTUs by Classification at a Given Taxonomic Rank*

Description

This function consumes an OTU, and a rank, as well as various optional parameters. It creates a stacked bar plot showing the abundance of all classifications at the given taxonomic rank for each sample.

Usage

```
group.abundance(data, rank,
                top=NULL, count=FALSE, drop.unclassified=FALSE,
                cex.x=NULL, main=NULL, file=NULL, ext=NULL,
                height=8, width=16, bw=FALSE, ggplot2=TRUE)
```

Arguments

data	a list of OTU tables.
rank	the taxonomic rank to use. See RAM.rank.formatting.
top	the number of groups to select, starting with the most abundant. If NULL, all are selected.
count	logical. If TRUE, the numerical counts for each OTU will be shown; otherwise the relative abundance will be shown.
drop.unclassified	logical. Should unclassified samples be excluded from the data?
cex.x	optional. The size of x axis names.
main	the title of the plot
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).
bw	logical. Should the image be created in black and white?
ggplot2	logical. Should the ggplot2 package be used to produce the plot, or should the base graphics be used? (see ?RAM.plotting).

Author(s)

Wen Chen and Joshua Simpson

Examples

```
data(ITS1, ITS2)

# plot the relative abundance at the class level to the screen, ignoring the
# unclassified
group.abundance(data=list(ITS1=ITS1), rank="phylum",
                drop.unclassified=TRUE)

## Not run:
# plot the count abundance at the phylum level to path.tiff
group.abundance(data=list(ITS1=ITS1, ITS2=ITS2), rank="g",
                top=10, count=FALSE, drop.unclassified=TRUE,
                main="", file=NULL, ext=NULL,
                height=8, width=16, bw=FALSE, ggplot=TRUE)

## End(Not run)
```

---

group.diversity *Boxplot To Compare Diversity Indices Among Groups*

Description

This function first use OTU.diversity to calculate the diversity indices for each sample and then do a boxplot to compare the selected indices among different groups.

Usage

```
group.diversity(data, meta, factors="", indices="",
                diversity.info=FALSE,
                x.axis=NULL, compare=NULL,
                facet=NULL, facet.y=TRUE, facet.x.cex=NULL,
                facet.y.cex=NULL, scale.free=NULL,
                xlab=NULL, ylab=NULL,
                legend.title=NULL, legend.labels=NULL,
                file=NULL, ext=NULL, width=8, height=8)
```

Arguments

data	a list, containing otu tables. See also RAM.input.formatting
meta	the metadata table to be used (must have same samples as data.
factors	a character vector. Must be variables in the metadata
indices	a character vector. Must be one or more of the following: "spec", "sim", "invsim", "shan", "sim_even", "shan_even", "sim_trudiv", "shan_trudiv", "chao", "ACE". See also OTU.diversity, true.diversity, evenness, and diversity.

diversity.info	logical. Whether the diversity indices have calculated and included in the metadata table. The diversity indices should be processed by OTU.diversity for the same otu tables and metadata table.
x.axis	optional. If NULL, will use the first variable in factors; otherwise, must be one factor in the metadata or 'SampleID'
compare	optional. If NULL, will use the first variable in factors; otherwise, must be one factor in the metadata
facet	optional. If provided, must be one factor in the metadata or 'SampleID'
facet.y	logical, whether the facet being used as strip text of y axis or x axis.
facet.x.cex	optional, an integer, the font size of the stip.text.x in ggplot
facet.y.cex	optional, an integer, the font size of the strip.text.y in ggplot.
scale.free	optional. Whether use free scale for y axis.
xlab	optional. If not provided, the x.axis will be used as the title of the x axis, otherwise, will use the provided string.
ylab	optional. If not provided, "value" will be used as the title of the y axis, otherwise, will use the provided string.
legend.title	optional. If not provided, compare will be used as the title of the legend, otherwise, will use the provided string.
legend.labels	optional. If not provided, will use the levels of compare for the legends, otherwise, will use the provided vector of strings. The length of the provided vector of strings must equals to the levels of compare.
file	the filename to save the plot.
ext	the extention (file type) of the plot to saved.
width	the width of the plot to be saved.
height	the heigth of the plot to be saved.

Value

This function returns a boxplot to compared selected diversity indices among different groups.

Author(s)

Wen Chen.

See Also

OTU.diversity, true.diversity, evenness and diversity

Examples

```
data(ITS1, ITS2, meta)
## Not run:
RAM.theme<-RAM.color()
group.diversity(data=list(ITS1=ITS1, ITS2=ITS2), meta=meta,
                factors=c("Crop", "City"),
```

```
                indices=c("sim_trudiv", "shan_trudiv"),
                x.axis="Crop", compare="Harvestmethod",
                facet="City", facet.y=FALSE) + RAM.theme

## End(Not run)
```

---

group.heatmap *Plot OTU Abundance at a Given Rank with Metadata Annotation*

Description

This function plots the abundance for all taxon groups at a given rank. Additionally, it can display metadata for the samples as annotations along the rows of the heatmap.

Usage

```
group.heatmap(data, is.OTU=TRUE, meta, rank, factors,
              top=25, remove.unclassified=TRUE,
              stand.method=NULL,
              dist.method="bray",
              hclust.method="average",
              dendro.row.status="yes",
              dendro.col.status="hidden",
              row.labels=TRUE, row.cex=1,
              cut=NULL, file=NULL, ext=NULL,
              width=9, height=9)
```

Arguments

data	the OTU table to be used.
is.OTU	logical. Whether or not the data is an OTU table.
meta	the metadata table to be used.
rank	the taxonomic rank to use (see RAM.rank.formatting for formatting details).
factors	a named character vector indicating the columns of the metadata table to be used (see RAM.factors).
top	the number of groups to select, starting with the most abundant. If NULL, all are selected.
remove.unclassified	logical. Define whether OTUs labelled "unclassified" or missing classification at the given taxonomic rank should be excluded.
stand.method	optional. Data standardization methods specified in decostand.
dist.method	distance matrix calculation methods specified vegdist.
hclust.method	the agglomeration methods specified in hclust. This should be unambiguous abbreviation of one of the following: 'ward.D', 'ward.D2', 'single', 'complete', 'average', 'mcquitty', 'median' or 'centroid'.

`dendro.row.status`
: whether or not to use the dendrogram to re-order the rows. It should be one of the following: 'yes' that use the dendrogram to re-order the rows/columns and display the dendrogram; 'hidden' means re-order, but do not display; 'no' means do not use the dendrogram at all. See also `annHeatmap2`

`dendro.col.status`
: whether or not to use the dendrogram to re-order the columns. It should be one of the following: 'yes' that use the dendrogram to re-order the rows/columns and display the dendrogram; 'hidden' means re-order, but do not display; 'no' means do not use the dendrogram at all. See also `annHeatmap2`

`row.labels`
: logical. Whether or not to plot the row labels.

`row.cex`
: the size of row labels if `row.labels` is `TRUE`

`cut`
: the height at which to cut the sample tree, this will create distinct coloured groups. Currently this will allow for at most nine groups (see Details).

`file`
: the file path where the image should be created (see ?RAM.plotting).

`ext`
: the file type to be used; one of `"pdf"`, `"png"`, `"tiff"`, `"bmp"`, `"jpg"`, or `"svg"`.

`height`
: the height of the image to be created (in inches).

`width`
: the width of the image to be created (in inches).

Details

A warning from `brewer.pal` indicating "n too large, allowed maximum for palette Pastel1 is 9" means that the cut height is too low to allow for that many groups. This should be fixed in a future release.

Author(s)

Wen Chen and Joshua Simpson.

See Also

`decostand`, `annHeatmap2`

Examples

```
data(ITS1, meta)
library("Heatplus")
library("RColorBrewer")

group.heatmap(ITS1, is.OTU=TRUE, meta=meta, rank="c", factors=c(Crop="Crop",
              City="City"), stand.method="chi", dist.method="euc",
              hclust.method="mcquitty", cut=0.5)
## Not run:
```

`group.heatmap.simple` *Plot a Heatmap Showing OTU Abundance by Taxonomic Classification*

Description

This function consumes an OTU table and a rank, as well as some optional parameters, and creates a heatmap showing the abundance of the OTUs at the given taxonomic rank for each sample.

Usage

```
group.heatmap.simple(data, is.OTU=TRUE, meta=NULL, rank, row.factor=NULL,
                     top=NULL, count=FALSE, drop.unclassified=FALSE,
                     dendro="none", file=NULL, ext=NULL,
                     width=9, height=8, leg.x=-0.08, leg.y=0)
```

Arguments

`data`	the OTU table to be used.
`is.OTU`	logical. Whether or not the data is an OTU table.
`meta`	the metadata table to be used.
`rank`	the taxonomic rank to use (see ?RAM.rank.formatting for formatting details).
`row.factor`	a factor from the metadata to show along the rows of the heatmap. (see Details below).
`dendro`	a character vector specifying on which axes (if any) a dendrogram should be plotted. Must be one of "`none`", "`both`", "`column`", or "`row`".
`top`	the number of groups to select, starting with the most abundant. If `NULL`, all are selected.
`count`	logical. Should the actual count of each OTU be shown, or should the relative abundances be shown?
`drop.unclassified`	logical. Should OTUs labelled "unclassified" or missing classification at the given taxonomic rank be excluded?
`file`	the file path where the image should be created (see ?RAM.plotting).
`ext`	the file type to be used; one of "`pdf`", "`png`", "`tiff`", "`bmp`", "`jpg`", or "`svg`".
`height`	the height of the image to be created (in inches).
`width`	the width of the image to be created (in inches).
`leg.x`	how far the legend should be inset, on the x axis.
`leg.y`	how far the legend should be inset, on the y axis.

Details

`row.factor` should be a named character vector specifying the name of the row to be used from `meta` (see `RAM.factors`). It should also be a factor; if it is not, a warning is raised and it is coerced to a factor (see `factor`). A warning is also raised when a factor has more than 8 levels, as that is the most colours the current palettes support. The factor must also correspond to the OTU table; i.e. they should have the same samples. If not, an error is raised.

Note

This function creates the heatmap using the `heatmap.2` function from the `gplots` package. That function calls `layout` to set up the plotting environment, which currently prevents plotting two data sets side by side, or to automatically place the (metadata) legend. Unfortunately, this means that the `leg.x` and `leg.y` parameters must be used to adjust the legend by trial and error. It is possible to move the legend outside of the plotting area; if no legend appears, try with small `leg.x` and `leg.y` values.

Author(s)

Wen Chen and Joshua Simpson.

See Also

`factor`, `heatmap.2`, `RAM.factors`

Examples

```
data(ITS1, meta)

# plot the abundance of all observed classes for each sample, displaying
# it to the screen and adding a dendrogram on the column and the Collector
# on the row
group.heatmap.simple(ITS1, is.OTU=TRUE, meta=meta,
                     row.factor=c(Crop="Crop"), dendro="row",
                     rank="g", top=10, drop.unclassified=TRUE,
                     leg.x=-0.06)

## Not run:
# plot the genus for all OTUs, add a dendrogram to the row and column,
# and save the plot in path.tiff
group.heatmap.simple(ITS1, is.OTU=TRUE, meta=meta, rank="genus",
                     dendro="both", file="my/file/path")
## End(Not run)
```

group.indicators *Plot Indicator Taxon Groups for Metadata Trends*

Description

This function conumes one or two OTU tables, along with a metadata facotr, and creates a barplot showing the relative abundance of all groups which are statistical indicators of that factor.

Usage

```
group.indicators(data, is.OTU=TRUE, meta, factor, rank,
                 thresholds = c(A = 0.85,
                                B = 0.8,
                                stat = 0.8,
                                p.value = 0.05),
                  all.indicators=TRUE, cex.x=NULL, file = NULL,
                  ext = NULL, height = 12, width = 12)
```

Arguments

data	a list of OTUs or taxonomic abundance matrices. see also RAM.input.formatting
is.OTU	logical. Whether the input data are OTU tables.
meta	the metadata table to be used.
factor	a named character vector (of length 1) giving the name of the column in meta to be used when performing the analysis (see RAM.factors).
rank	the taxonomic rank to use (see ?RAM.rank.formatting for formatting details). if rank is NULL, will use otus as indicators which are annotated by the lca assigned to the otus, otherwise will use taxon names as indicators at the given taxonomic rank.
thresholds	a character vector of length 4 specifying the thresholds for the A, B, stat, and p values (see Details).
all.indicators	logical. Whether or not plot all identified indicators. If set as TRUE, will plot all indicators, otherwise, if the total indicators are less than 12, will plot them all; if the total indicators are more than 12, will plot the first 12 of them.
cex.x	optional. The size of x axis names.
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).

Details

This function uses `indicspecies::multipatt` to determine the indicators. After this analysis is performed, there will likely be some species determined to be 'significant,' but to varying degrees. To control how many groups are selected, you can adjust the `thresholds` argument. It consists of four components: (quotations taken from `vignette("indicspeciesTutorial")`, see References):

A the *specificity* of the indicator; "the probability that the surveyed site belongs to the target site group given the fact that the species has been found."

B the *fidelity* of the indicator; "the probability of finding the species in sites belonging to the site group."

stat the association strength for the combinations.

p.value "the degree of statistical significance of the association."

Only the species with `A`, `B`, and `stat` values above, and `p.value` below those given in `thresholds` will be kept.

Value

This function returns a stacked barplot and a vector of identified indicators, including the ones in the plot and the ones being excluded from the plot.

Author(s)

Wen Chen and Joshua Simpson.

References

De Caceres, M., Legendre, P. (2009). Associations between species and groups of sites: indices and statistical inference. Ecology, URL http://sites.google.com/site/miqueldecaceres/

See Also

`multipatt`

Examples

```
data(ITS1, ITS2, meta)
## Not run:
# inputs are OTU tables
group.indicators(data=list(ITS1=ITS1, ITS2=ITS2), is.OTU=TRUE, meta,
                    factor = c(Province="Province"),
                    rank="g")
group.indicators(data=list(ITS1=ITS1), is.OTU=TRUE, meta,
                    factor = c(Province="Province"),
                    rank=NULL, all.indicators=TRUE)
group.indicators(data=list(ITS1=ITS1), is.OTU=TRUE, meta,
                    factor = c(Province="Province"),
                    rank=NULL, all.indicators=FALSE)
# inputs are taxonomic abundance matrices
g1 <- tax.abund(ITS1, rank="g")
```

```
g2 <- tax.abund(ITS2, rank="g")
group.indicators(data=list(g1=g1, g2=g2), is.OTU=FALSE, meta,
                    factor = c(Province="Province"),
                    rank="g")


## End(Not run)
```

`group.OTU` *Plot Distribution of OTUs*

Description

This function plot the distributon of otus in each level of a given metadata variable. The plot can be boxplot of barplot. The boxplot shows the range of relative abundance of a given otuID in each level of metadata category. The barplot shows the relative abundance of the total counts of a given otuID in each level of metadata category.

Usage

```
group.OTU(otu, rank="g", otuIDs="", meta, meta.factor="",
         boxplot=TRUE, main="", file=NULL, ext=NULL,
         height=8, width=16)
```

Arguments

`otu`	the OTU table to be analyzed.
`rank`	the taxonomic rank(s) of otu classification (see ?RAM.rank.formatting for formatting details).
`otuIDs`	an vector of otuIDs in the OTU table
`meta`	the metadata table to be analyzed.
`meta.factor`	the metadata qualitative variable
`boxplot`	logical. If `TRUE`, generate boxplot; otherwise generate barplot.
`main`	title of the plot.
`file`	the file path where the image should be created (see ?RAM.plotting).
`ext`	the file type to be used; one of `"pdf"`, `"png"`, `"tiff"`, `"bmp"`, `"jpg"`, or `"svg"`.
`height`	the height of the image to be created (in inches).
`width`	the width of the image to be created (in inches).

Value

`group.OTU` returns boxplot or barplot for the distribution of a list of otuIDs.

Note

The OTUs are determined to be absent/present using the "pa" method from the function `decostand`.

Author(s)

Wen Chen

See Also

ggplot

Examples

```
data(ITS1, meta)

# otuIDs
otuIDs=rownames(ITS1)[1:10]

# names(meta)
theme <- RAM.color()
group.OTU(otu=ITS1, rank="g", otuIDs=otuIDs,
        meta=meta, meta.factor="City", boxplot=TRUE,
        file=NULL, ext=NULL) + theme

## Not run:
group.OTU(otu=ITS1, rank="g", otuIDs=otuIDs,
               meta=meta, meta.factor="City", boxplot=FALSE,
               file=NULL) + theme

## End(Not run)
```

---

group.rich *Barplot Of Richness For Each Level Of A Given Metadata Variable*

Description

This function first use specpool to estimate the extrapolated species richness in a species pool (levels of metadata variable), and the number of unobserved species, then do a barplot.

Usage

```
group.rich(otu, meta, factor, file=NULL, ext=NULL, width=8, height=8)
```

Arguments

otu	an OTU table.
meta	the metadata table to be used (must have same samples as data.
factor	a character string. Must be one of the metadata variables.
file	optional. Filename that the plot to be saved to.
ext	optional. Filename extension, type of image to be saved.
width	an integer. Filter OTU table by counts.
height	an integer. Filter OTU table by counts.

Value

This function returns a barplot of species richness for a given metadata variable.

Author(s)

Wen Chen.

See Also

specpool, specpool

Examples

```
data(ITS1, meta)

group.rich(ITS1, meta, "Crop")
```

---

group.spatial *Plot Spatial Collection Trends for Taxon Groups*

Description

This function consumes an OTU table and its associated metadata table, and uses that information to produce a choropleth (essentially a heatmap, but superimposed imposed on an actual, cartographic map) to show how many counts of each taxon group were detected in each Canadian province/territory.

Usage

```
group.spatial(data, meta, date.col, province.col, rank, group,
              breaks = "year", file = NULL, ext = NULL,
              height = 8, width = 10)
```

Arguments

data	the OTU table to be used.
meta	the metadata table to be used.
date.col	a character vector specifying which column in metadata contains the date information (see RAM.dates).
province.col	a character vector specifying which column in metadata contains the province information (see Details).
rank	a character vector specifying the rank of the desired taxon groups. Note that all groups should come the same rank. (see RAM.rank.formatting).
group	a character vector giving the names of the groups to be plotted.
breaks	how many time segments should be plotted; see Details.

file	the file path where the image should be created. (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).

Details

This function currently only supports Canadian data. The entries in meta$province.col should be specified as provinces, using the standard postal abbreviations (e.g. "Ontario" would be "ON").

The breaks argument is slightly buggy at the moment, possibly due to how R tries to split Date objects. breaks can be either an integer, in which case it will attempt to create that many levels (i.e. setting breaks=3 should split the data into three date 'blocks'.) breaks can also be a character vectors, such as "quarter" or "year" which attempts to split the date information accordingly. See cut.Date for more details and a complete specification of what is allowed for breaks.

Author(s)

Wen Chen and Joshua Simpson.

References

The file used to create the map of Canada is from GeoBase and is licensed under the Open Government License - Canada.

Examples

```
data(ITS1, meta)

## Not run:
group.spatial(ITS1, meta, date.col="Harvestdate",
              province.col="Province", rank="p",
              group=c("Ascomycota", "Basidiomycota"),
              breaks=2)

## End(Not run)
```

---

group.spec *Boxplot Of Richness For Each Level Of A Given Metadata Variable*

Description

This function first use specpool to estimate the extrapolated species richness in a species pool (levels of metadata variable), and the number of unobserved species, then do a boxplot for percent of observed richness.

Usage

```
group.spec(otu, meta, factor, file=NULL, ext=NULL, width=8,height=8)
```

Arguments

otu	an OTU table.
meta	the metadata table to be used (must have same samples as data.
factor	a character string. Must be one of the metadata variables.
file	optional. Filename that the plot to be saved to.
ext	optional. Filename extension, type of image to be saved.
width	an integer. Filter OTU table by counts.
height	an integer. Filter OTU table by counts.

Value

This function returns a boxplot of species richness for a given metadata variable.

Author(s)

Wen Chen.

See Also

specpool, specpool

Examples

```
data(ITS1, meta)

group.spec(ITS1, meta, "Crop")
```

---

group.Taxa.bar *Barplot Of Taxa Distribution In Groups*

Description

This function do a barplot to show the distribution of selected taxa in each level of a given metadata variable

Usage

```
group.Taxa.bar(data, is.OTU=TRUE, rank="g", taxa="",
               meta, meta.factor="", cex.y=5, cex.x=5,
               bar.width=NULL, RAM.theme=NULL,
               col.pal=NULL, main="", file=NULL, ext=NULL,
               height=8, width=16)
```

Arguments

data	a list of otu tables or taxonomic abundance matrices. See also RAM.input.formatting.
is.OTU	logical. If an OTU table was provided, is.OTU should be set as TRUE; otherwise, it should be set as FALSE.
rank	a single taxonomic rank. See also RAM.rank.formatting
taxa	a vector containing taxa names for plotting.
meta	the metadata table to be used (must have same samples as data.
meta.factor	a character string. Must be one of the metadata variables.
cex.y	size of y axis tick labels.
cex.x	size of x axis tick labels.
bar.width	width of bars
RAM.theme	customized ggplot_theme in RAM. See also ?theme_ggplot.
col.pal	color palettes to be used.
main	a character string. The title of the plot, default is an empty string.
file	filename to save the plot.
ext	filename extension, the type of image to be saved to.
width	an integer, width of the plot.
height	an integer, height of the plot.

Details

To use customized color palettes, must pass a vector of color names or hexadecimals. See examples for detail.

Value

This function returns a barplot.

Author(s)

Wen Chen

Examples

```
data(ITS1, ITS2, meta)
taxa <- c("Fusarium", "Alternaria", "Cladosporium")
group.Taxa.bar(data=list(ITS1=ITS1, ITS2=ITS2), is.OTU=TRUE,
               rank="g", taxa=taxa, meta=meta, meta.factor="City",
               cex.y=5, cex.x=5, bar.width=0.5, RAM.theme=RAM.color())
## Not run:
# change default color schemes
col <- c("dodgerblue1", "darkcyan")
taxa.1 <- c("Fusarium", "Alternaria", "Cladosporium", "Verticillium",
          "Kondoa")
group.Taxa.bar(data=list(ITS1=ITS1, ITS2=ITS2), is.OTU=TRUE,
```

```
                rank="g", taxa=taxa.1, meta=meta, meta.factor="City",
                cex.y=5, cex.x=5, bar.width=0.5, RAM.theme=NULL,
                col.pal=col)
# change ggplot theme
group.Taxa.bar(data=list(ITS1=ITS1, ITS2=ITS2), is.OTU=TRUE,
                rank="g", taxa=taxa.1, meta=meta, meta.factor="City",
                cex.y=5, cex.x=5, bar.width=0.5,
                col.pal=col, RAM.theme=RAM.border())
# save the plot
group.Taxa.bar(data=list(ITS1=ITS1, ITS2=ITS2), is.OTU=TRUE,
                rank="g", taxa=taxa.1, meta=meta, meta.factor="City",
                cex.y=5, cex.x=5, bar.width=0.5, RAM.theme=NULL,
                col.pal=col,main="", file="path/to/filename.pdf",
                ext="pdf", height=8, width=16)

## End(Not run)
```

`group.Taxa.box` *Boxplot Of Distribution Of Taxa In Each Level of A Metadata Variable*

Description

This function do a boxplot to show the distribution of selected taxa in each level of a given metadata variable

Usage

```
group.Taxa.box(data, is.OTU=TRUE, rank="g",
                taxa="", meta, meta.factor="",
                cex.y=5, cex.x=5,
                RAM.theme=NULL, col.pal=NULL, main="",
                file=NULL, ext=NULL, height=8, width=16)
```

Arguments

`data`	a list of otu tables or taxonomic abundance matrices. See also `RAM.input.formatting`.
`is.OTU`	logical. If an OTU table was provided, `is.OTU` should be set as `TRUE`; otherwise, it should be set as `FALSE`.
`rank`	a single taxonomic rank. See also `RAM.rank.formatting`
`taxa`	a vector containing taxa names for plotting.
`meta`	the metadata table to be used (must have same samples as `data`.
`meta.factor`	a character string. Must be one of the metadata variables.
`cex.y`	size of y axis tick labels.
`cex.x`	size of x axis tick labels.
`RAM.theme`	customized ggplot_theme in RAM. See also ?theme_ggplot.
`col.pal`	color palettes to be used.

main	a character string. The title of the plot, default is an empty string.
file	filename to save the plot.
ext	filename extension, the type of image to be saved to.
width	an integer, width of the plot.
height	an integer, height of the plot.

Value

This function returns a boxplot of the distribution of seleted taxa within each level of a given metadata variable.

Author(s)

Wen Chen.

Examples

```
data(ITS1, ITS2, meta)
taxa <- c("Fusarium", "Alternaria", "Cladosporium")
group.Taxa.box(data=list(ITS1=ITS1, ITS2=ITS2), is.OTU=TRUE, rank="g",
          taxa=taxa, meta=meta, meta.factor="City")
## Not run:
taxa.1 <- c("Fusarium", "Alternaria", "Cladosporium", "Verticillium",
          "Kondoa")
group.Taxa.box(data=list(ITS1=ITS1, ITS2=ITS2), is.OTU=TRUE, rank="g",
          taxa=taxa.1, meta=meta, meta.factor="City")

## End(Not run)
```

---

group.temporal *Plot Temporal Trends for Metadata and Taxon Groups*

Description

This function consumes an OTU table and its associated metadata, and creates a plot showing how the collections of taxonomic groups, as well as metadata factors, evolve over time.

Usage

```
group.temporal(data, meta, date.col, factors, rank, group,
               file = NULL, ext = NULL, height = 8, width = 12)
```

Arguments

data	the OTU table to be used.
meta	the metadata table to be used.
date.col	a character vector specifying which column of the metadata has date information (see RAM.dates).
factors	a named character vector specifying the names of the metadata columns to be plotted with the taxon group data. (see RAM.factors). NOTE: these factors must be *numeric* variables.
rank	a character vector specifying the rank of the desired taxon groups. Note that all groups should come the same rank. (see RAM.rank.formatting).
group	a character vector giving the names of the groups to be plotted.
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).

Details

The image created will contain several plots. It will always contain a large panel showing the counts collected for the specified taxon groups over time, and above that panel (on a common x-axis) will be a line graph for each metadata factor specified.

Note

If your data has collections being taken roughly annually, you may have a large amount of "empty space" in the middle of your plot. Consider subsetting the data by year, and plotting each year individually using this function.

Author(s)

Wen Chen and Joshua Simpson

Examples

```
data(ITS1, meta)

group.temporal(ITS1, meta, date.col="Harvestdate",
               factors=c(Ergosterol="Ergosterol_ppm"),
               rank="p", group=c("Ascomycota", "Basidiomycota"))
```

group.venn *Plot Venn Diagram For Two To Five Sets With Item Labels*

Description

This function use draw.pairwise.venn to creates a venn diagram for two vectors

Usage

```
group.venn(vectors, cat.cex=1.5, cex=1,
           cat.pos=NULL, cat.dist=NULL,
           label=TRUE, lab.cex=1,
           lab.col= "black", fill=NULL,
           file=NULL, ext=NULL, width=8, height=8)
```

Arguments

vectors	a list of vectors with names. See also RAM.input.formatting.
cat.cex	size of the category names. (see venn.diagram for details).
cex	size of the label of the circles. (see venn.diagram for details).
cat.pos	optional. Location of the category names along the circles. (see venn.diagram for details).
cat.dist	optional. Distance of the category names to the circles. (see venn.diagram for details).
label	logical. If TRUE, will plot the item labels for 2 data sets. For more than 2 datasets or this is set as FALSE, the labels will be numbers for each circle. (see venn.diagram for details).
lab.cex	size of the labels.
lab.col	color of the labels.
fill	optional, color of the circles. (see venn.diagram for details).
file	the file path where the image should be created (see ?RAM.plotting).
ext	filename extension, the type of image to be saved to.
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).

Value

group.venn returns a venn diagram for 2 to 5 sets. The user can choose to place item labels for 2 sets of data, however, the label locations can be wrong if the the smaller data set is part of the bigger data set, in this case, set label as FALSE. If the input datasets is more than 2, label will be ignored.

Author(s)

Wen Chen

See Also

see venn.diagram

Examples

```
data(ITS1, meta)
# core OTUs
core <- core.OTU.rank(data=list(ITS1=ITS1), meta=meta, rank="g",
                      meta.factor="Crop", percent=1)

# taxa that core OTUs assigned to
core.Crop1 <- core$ITS1$Crop1$taxa
core.Crop2 <- core$ITS1$Crop2$taxa

# venn plot
vectors <- list(Core_Crop1=core.Crop1, Core_Crop2=core.Crop2)
group.venn(vectors=vectors, label=TRUE, cat.pos=c(330, 150),
           lab.cex=0.7)
## Not run:
group.venn(vectors=vectors, label=FALSE, cat.pos=c(330, 150),
           lab.cex=0.7, cex=3)

## End(Not run)
```

ITS1/ITS2 *Sample ITS1 and ITS2 Data*

Description

Sample ITS1 and ITS2 OTU tables.

Usage

```
data(ITS1)
data(ITS2)
```

Format

A data frame with 4704 observations on the following 17 variables.

`P1001.1M1, P1001.1M2, P1001.1M3, P1001.1M4, P1001.1M5, P1001.1M6, P1001.1M7, P1001.1M8, P1001.1M9,`
Collection samples.

`taxonomy` the taxonomic classification of the OTU.

Source

Wen Chen, AAFC-AAC

Examples

```
data(ITS1, ITS2)

str(ITS1)
str(ITS2)
```

LCA.OTU *Lowest Common Ancestor (LCA) OF EACH OTU*

Description

This function consumes an OTU table and extract the LCA (lowest common ancestor) that each otu assigned to. See also tax.split.

Usage

```
LCA.OTU(otu, strip.format=FALSE, drop=TRUE)
```

Arguments

otu	the OTU table to be used.
strip.format	logical. Whether or not to remove the prefix of the taxonomy assignment at each rank. see
drop	logical. Whether or not drop taxonomic columns other than LCA.

Value

This function return a data frame same as the input OTU table, except the last column is the LCA of each otu, not the lineage. The taxonomy column can be kept, by using drop.

Note

tax.split returns the same OTU table with classification at a given taxonomic rank, which can be missing if an otu was not classified a that that taxonomic level. LCA.OTU, guaranteed that all OTUs will be preserved in the returned data table and provide the LCA for each OTU, although only higher taxonomic ranks were available.

Author(s)

Wen Chen

See Also

tax.split

Examples

```
data(ITS1)
## Not run:
# compare the following 2 commands:
# keep the rank prefix of the LCA column
ITS1.LCA <- LCA.OTU(ITS1, strip.format=TRUE, drop=TRUE)
# remove the rank prefix of the LCA column
ITS1.LCA <- LCA.OTU(ITS1, strip.format=FALSE, drop=TRUE)

## End(Not run)
```

---

location.formatting *Location Formatting*

Description

Some functions in RAM expect to find a column with provincial/territorial data in the metadata. This data should use the standard Canadian provincial/territorial abbreviations:

- Alberta - AB
- British Columbia - BC
- Manitoba - MB
- New Brunswick - NB
- Newfoundland and Labrador - NL
- Novia Scotia - NS
- Northwest Territories - NT
- Nunavut - NU
- Ontario - ON
- Prince Edward Island - PE
- Quebec - QC
- Saskatchewan - SK
- Yukon - YT

Support for other locations is not available at this time.

Author(s)

Wen Chen and Joshua Simpson.

match.data *Match Samples In Ecology Data Sets and Metadata*

Description

This function will match samples in ecology data sets, either OTU tables or taxonomy abundance matrices, and those in metadata. It makes sure that datasets contains same samples in the same order.

Usage

```
match.data(data, is.OTU=TRUE, meta)
```

Arguments

data	a list of ecology data sets. if is.OTU is TRUE, they should be OTU tables, otherwise should be taxonomy abundance matrices. See also RAM.input.formatting.
is.OTU	logical, whether or not the ecology data sets are OTU tables.
meta	metadata associated with input ecology data sets.

Author(s)

Wen Chen

See Also

RAM.input.formatting

Examples

```
## Not run:
data(ITS1, ITS2, meta)
meta <- meta[1:8, ]
# use otu tables
matched <- match.data(data=list(otu_ITS1=ITS1, otu_ITS2=ITS2),
           is.OTU=TRUE, meta=meta)

# use taxonomy abundance matrices
g1 <- tax.abund(ITS1, rank="g")
g2 <- tax.abund(ITS2, rank="g")
matched <- match.data(data=list(genus_ITS1=g1, genus_ITS2=g2),
           is.OTU=FALSE, meta=meta)
# class(matched)
# names(matched)

## End(Not run)
```

meta *Sample Metadata for ITS1/ITS2*

Description

This data frame provides sample metadata for the ITS1/ITS2 data included in this package.

Usage

```
data(meta)
```

Format

A data frame with 16 observations on the following 10 variables.

`Sample` a factor with levels `Sample1` `Sample2` `Sample3` `Sample4` `Sample5` `Sample6` `Sample7` `Sample8`

`City` a factor with levels `City1` `City2`

`Crop` a factor with levels `Crop1` `Crop2`

`Plots` a factor with levels `1` `2`

`Harvestmethod` a factor with levels `Method1` `Method2`

`Harvestdate` a Date

`Ergosterol_ppm` a numeric vector

`Province` a character vector

`Latitude` a numeric vector

`Longitude` a numeric vector

Source

Wen Chen and Joshua Simpson.

Examples

```
data(meta)

str(meta)
```

`META.clust` *Plot Hierarchical Cluster Of Metadata*

Description

This function plot hierarchical cluster Of metadata.

Usage

```
META.clust(meta, group=4, data.trans=NULL, dist=NULL, clust=NULL,
          type=NULL, main="", file=NULL, ext=NULL, width=8, height=8)
```

Arguments

`meta`	the metadata table to be clustered.
`group`	an integer or a metadata variable. If an integar, will cut tree into corresponding groups and color them accordingly; if a metadata variable was provided, tree leaves (sampleIDs) will be colored by each level.
`data.trans`	optional. If was provided, numeric data will be transformed. See also `decostand`
`dist`	optional. If was provided, distance matrix will be calculated using the give method for numeric variables; otherwise use `vegdist` default Bray-Curtis method. If metadata include qualitative variables, distance matrix will be calculated by `gowdis`.
`clust`	optional. If was not provided, will use the default agglomeration method used by hclust, i.e. "complete". Otherwise, will used user defined method for clustering. See also `hclust`.
`type`	optional. Can be one of the following: "triangle", "rectangle", "phylogram", "cladogram", "fan", "unrooted", "radial".
`main`	The title of the plot.
`file`	optional. Filename that the plot to be saved to.
`ext`	optional. File type that the plot to be saved to.
`width`	an integer, width of the plot.
`height`	an integer, height of the plot.

Value

This function return a plot of the hierarchical cluster analysis on a set of metadata.

Author(s)

Wen Chen

See Also

`vegdist` and `gowdis`.

Examples

```
data(meta)

META.clust(meta=meta, type="fan")
META.clust(meta=meta, type="fan", group="City")
```

`OTU.diversity` *Summarize Diversity Indices for OTU Tables*

Description

These functions calculate diversity indices for all samples and append outputs as new columns to metadata table.

Usage

```
OTU.diversity(data, meta)
```

Arguments

`data`	a list of OTU tables.
`meta`	the metadata to append the outputs.

Details

This function summarize the following diversity indices: specnumber, shannon, simpson, invsimpson, true diversity, evenness, chao and ACE indices, for a given otu table. See `diversity.indices`)

Value

This function return vectors of diversity indices for each sample, which are appended to a given metadata table.

Note

Credit goes to package `vegan` for the partial argument matching (see References), and for the calculation of all diversity indices except for true diversity and evenness.

Author(s)

Wen Chen.

References

Jari Oksanen, F. Guillaume Blanchet, Roeland Kindt, Pierre Legendre, Peter R. Minchin, R. B. O'Hara, Gavin L. Simpson, Peter Solymos, M. Henry H. Stevens and Helene Wagner (2013). vegan: Community Ecology Package. R package version 2.0-10. http://CRAN.R-project.org/package=vegan

Diversity index. (2014, May 7). In Wikipedia, The Free Encyclopedia. Retrieved 14:57, May 28, 2014, from http://en.wikipedia.org/w/index.php?title=Diversity_index&oldid=607510424

Blackwood, C. B., Hudleston, D., Zak, D. R., & Buyer, J. S. (2007). Interpreting ecological OTU.diversity applied to terminal restriction fragment length polymorphism data: insights from simulated microbial communities. Applied and Environmental Microbiology, 73(16), 5276-5283.

Examples

```
data(ITS1, ITS2, meta)
data=list(ITS1=ITS1, ITS2=ITS2)
## Not run:
meta.diversity<-OTU.diversity(data=data, meta=meta)

## End(Not run)
```

`OTU.ord` *Ordination Plot For OTUs Using CCA or RDA Analysis*

Description

This function consumes an OTU table, metadata factors, and graphing options, then produces a plot showing the `cca` or `rda` analysis of the OTU table.

Usage

```
OTU.ord(otu, meta=meta, factors=NULL, group=NULL,
        na.action=c("na.fail", "na.omit", "na.exclude"),
        rank="g", taxa=NULL, data.trans="total",
        plot.species=TRUE, plot.scaling=-1,
        biplot.scale=NULL, biplot.sig=NULL, biplot.label= TRUE,
        mode=c("rda", "cca"), choice=c(1,2),
        main="", cex.point=3, cex.leg=12, cex.bp=3,
        file=NULL, ext=NULL, width=10, height=10)
```

Arguments

`otu`	the OTU table to be used.
`meta`	the metadata table to be used.
`factors`	a named character vector of length 1 or 2 specifying metadata factors for the samples in the OTU table (see Details).
`group`	a named character vector of length 1 or 2 specifying metadata factors for the samples in the OTU table (see Details).

na.action	choice of one of the following: "na.fail", "na.omit" or "na.exclude", see na.action in cca for detail.
rank	the rank to select the taxon groups at.
taxa	an integer or a character vector of taxa names at the given rank. If it's an integer, will display the top most abundant taxa; if it's a vector of taxa names, will plot the selected taxa.
data.trans	a method used to standardize the OTU table. One of "total", "max", "freq", "normalize", "range", "standardize", "pa", "chi.square", "hellinger" or "log" (see ?decostand).
plot.species	whether plot sites or taxa, should be reflex to plot.scaling
plot.scaling	one of the following: 1, 2, 3, or -1. See scaling in plot.cca for detail. See also ordiplot
biplot.scale	a numeric number, length of the biplot arrows
biplot.sig	significance cutoff for biplot to be displayed. Currently disabled because in the function, calculated ordination model cannot be passed to anova test.
biplot.label	whether or not to plot biplot
mode	one of the following: "cca" or "rda".
choice	the chosen axes
main	title of the plot
cex.point	size of points
cex.leg	size of lengend name
cex.bp	size of biplot labels
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
width	the width of the image to be created (in inches).
height	the height of the image to be created (in inches).

Details

group should be a named character vector specifying the names of the columns to be used from meta (see RAM.factors). The values on the axes denote what fraction of the sum of all eigenvalues (i.e. from all axes) is explained by that (single) axis.

Value

return a list of following: 1) ggplot object; 2) ordination model; 3) commodity data and 4) metadata used for the ordination model.

Author(s)

Wen Chen.

See Also

decostand, Taxa.ord, pcoa.plot

Examples

```
data(ITS1, meta)
its1<- filter.OTU(data=list(ITS1=ITS1), percent=0.001)[[1]]
factors=c("City", "Crop", "Harvestmethod", "Ergosterol_ppm")
## Not run:
# plot sites
ord1 <- OTU.ord(its1,  meta=meta, data.trans="total",
        factors=factors, mode="cca", biplot.sig=0.1,
        taxa=20, biplot.scale=1.5, cex.point=5,
        plot.species=FALSE, rank="f", plot.scaling=3,
        group=c(City="City", Crop="Crop"))
# plot taxa
ord2 <- OTU.ord(its1,  meta=meta, data.trans="total", plot.scaling=-1,
        factors=factors, mode="cca", biplot.sig=0.1,
        taxa=20, biplot.scale=3, cex.point=3,
        plot.species=TRUE, rank="g")

## End(Not run)
```

OTU.rarefy *Create Rarefied OTU Tables*

Description

This function output rarefied OTU tables using `rrarefy`. This function may take long time for large dataset, e.g. over 100k otus x 45 samples.

Usage

```
OTU.rarefy(data, sample=NULL)
```

Arguments

`data`	a list of otu tables. See also `RAM.input.formatting`.
`sample`	an integer represent the sampling size.

Value

This function returns a list of rarefied otu tables.

Note

See also `rrarefy`

Author(s)

Wen Chen

See Also

RAM.input.formatting.

Examples

```
## Not run:
data(ITS1, ITS2)
otus.rf <- OTU.rarefy(data=list(ITS1=ITS1, ITS2=ITS2),
                      sample=NULL)

## End(Not run)
```

OTU.recap *Summarize OTU*

Description

This function summarize OTU table at each given taxonomic ranks.

Usage

```
OTU.recap(data, ranks=c("p", "c", "o", "f", "g"),
          brewer.pal="Pastel1", file=NULL, ext="pdf",
          width=12, height=8)
```

Arguments

data	a list of otu tables. See also RAM.input.formatting.
ranks	a vector of taxonomic ranks. See also RAM.rank.formatting
brewer.pal	one of the color patterns available in RColorBrewer. See brewer.pal for available selections.
file	filename to save the plot.
ext	extention of the filename to save the plot.
width	width of the plot
height	heigth of the plot

Value

This function returns either a data frame or a list of data frames. If a single otu was provided, it returns the a dataframe with information of how many otuIDs and sequences being classified at selected taxonomic ranks. If more than 1 otu tables being provided, it returns a list, with the first a few are data frames of classification summary of each otu table, the last is a list showing taxa found only in one of the otu data set.

This function also generates a barplot for the percent classified otus and sequences at each given rank.

Note

warning is raised when run strsplit() and can be ignored.

Author(s)

Wen Chen

See Also

RAM.rank.formatting and RAM.input.formatting.

Examples

```
data(ITS1, ITS2)

ranks <- c("p", "c", "o", "f", "g")
df <- OTU.recap(data=list(ITS1=ITS1, ITS2=ITS2), ranks=ranks)
class(df)
```

---

pcoa.plot *Create a PCoA plot for an OTU Table*

Description

This function consumes an OTU table, metadata factors, and graphing options, then produces a plot showing the PCoA analysis of the OTU table.

Usage

```
pcoa.plot(data, is.OTU=TRUE, meta, factors, rank, stand.method = NULL,
          dist.method = "morisita", sample.labels = TRUE, top = 20,
          ellipse = FALSE, main = NULL, file = NULL, ext = NULL,
          height = 8, width = 10, ggplot2 = TRUE, bw = FALSE)
```

Arguments

data	an OTU table or taxonomic abundance matrix to be used.
is.OTU	logical. Whether or not the input data an OTU table.
meta	the metadata table to be used.
factors	a named character vector of length 1 or 2 specifying metadata factors for the samples in the OTU table (see Details).
rank	the rank to select the taxon groups at. For an OTU table, if rank is set NULL, distance matrix will be calcuated using all OTUs, otherwise, the OTU table will be transformed to taxonomic abundance matrix before the calculation of the distance matrix. If a taxonomic abundance matrix is provided, i.e. is.OTU is set TRUE, then the rank will be ignored.

stand.method	a method used to standardize the OTU table. One of "total", "max", "freq", "normalize", "range", "standardize", "pa", "chi.square", "hellinger" or "log" (see ?decostand).
dist.method	the dissimilarity index to be used; one of "manhattan", "euclidean", "canberra", "bray", "kulczynski", "jaccard", "gower", "altGower","morisita", "horn", "mountford", "raup", "binomial", "chao", or "cao" (see vegdist).
sample.labels	logical. Should the labels for the samples be displayed?
top	how many taxon groups should be displayed, starting from the most abundant.
ellipse	which of the metadata factors (if any) should have ellipses plotted around them. Must be one of 1, 2, or FALSE (see Details).
main	The title of the plot.
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).
ggplot2	logical. Should the ggplot2 package be used to produce the plot, or should the base graphics be used? (see ?RAM.plotting).
bw	logical. Should the image be created in black and white?

Details

This function uses pco in the labdsv package for the Principal coordinates analysis (PCoA). The distance matrix was square rooted before being passed to pco to avoid negative eigenvalues. factors should be a named character vector specifying the names of the columns to be used from meta (see RAM.factors). Those columns should be factors; if they are not, a warning is raised and they are coerced to factors (see factor). A warning is also raised when a factor has more than 9 levels, as that is the most colours the current palettes support.

The values on the axes denote what fraction of the sum of all eigenvalues (i.e. from all axes) is explained by that (single) axis.

When ellipse = FALSE, no ellipses will be plotted. When ellipse is a number, that 'number' metadata factor will have ellipses plotted. For example, if factors = c(Crop="Crop", City="City") and ellipse = 1, ellipses will be plotted for the different crops, but NOT the cities. Setting factors = c(City="City") and ellipse = 2 is invalid, since there is no second metadata factor given. Ellipses can only be plotted for one factor currently. Furthermore, there need to be at least 3 samples for every level in every item in factors, otherwise ellipses cannot be plotted.

Value

When ggplot2 = TRUE, a ggplot object is returned; otherwise nothing ism returned (but the plot is shown on screen).

Note

The labels for the sample points are placed above, below, or next to the point itself at random. If labels are outside of the plotting area, or overlapping with each other, run your command again (without changing any arguments!) and the labels should move to new positions. Repeat until they are placed appropriately. This is done to ensure even tightly-grouped samples, or samples near the edge of the plot, have their labels shown. If the labels are too distracting, remember that they can be turned off by setting `sample.labels = FALSE`.

Author(s)

Wen Chen and Joshua Simpson.

See Also

vegdist

Examples

```
data(ITS1, meta)

# The argument for factors is a vector of length two; the first
# item is # Crop, which is a column from meta, and the second item
# is City, another # column from meta.

pcoa.plot(ITS1, meta=meta, rank="c",
          factors=c(Crop="Crop", City="City"))

## Not run:
# If you want to customize legend labels and plot the top 20 taxon
# groups at genus:
pcoa.plot(ITS1, meta=meta, rank="g", main="PCoA plot",
          factors=c(Place="City", Harvest_Method="Harvestmethod"))

# In black & white, using base graphics:
pcoa.plot(ITS1, meta=meta, rank="c", factors=c(Plot="Plots"),
          ggplot=F, bw=T)

pcoa.plot(ITS1, meta=meta, rank="c", factors=c(Plot="Plots"),
          ggplot=F, bw=T, dist.method="euc", stand.method="hell")

# Focus on the samples: hide all groups and plot ellipses for Crop:
pcoa.plot(ITS1, meta=meta, rank="g",
          factors=c(Crop="Crop", City="City"),
          ellipse=1, sample.labels=FALSE, top=0)

# Standardize the data before calculating distances:
pcoa.plot(ITS1, meta=meta, rank="g", factors=c(City="City"),
          stand.method="chi.square",
          dist.method="euclidean")
```

```
## End(Not run)
```

`percent.classified` *Calculate Percent of OTUs Classified at a Given Taxonomic Rank*

Description

This function consumes an OTU table, and a vector containing taxnomic ranks, then returns what percent of OTUs in the given table are classified at each taxonomic rank.

Usage

```
percent.classified(data, ranks=c("f","g"))
```

Arguments

`data`	a list of OTU tables to be processed. See also `RAM.input.formatting`
`ranks`	a vector containing the taxonomic ranks you are interested in (see ?RAM.rank.formatting for formatting details).

Value

A list of numeric vectors, containing the result for each taxonomic rank.

Author(s)

Wen Chen and Joshua Simpson.

Examples

```
data(ITS1, ITS2)
data <- list(ITS1=ITS1, ITS2=ITS2)
# find what percent of OTUs classified at family and genus levels
percent.classified(data=data, ranks=c("f","g"))
```

`RAM.dates` *Date Formatting for RAM*

Description

This help page details the expected format for dates in RAM.

Details

For all functions expecting some type of date data, you will need to specify which column of the metadata table contains that information. The date information will likely be encoded as a character vector from read.meta, so these functions will try to coerce it to a Date object (see `Date` and `as.Date`), with a warning. These functions are expecting the date information to be in YYYY-MM-DD format.

RAM.factors *Factor Formatting for RAM*

Description

This help page details how to pass metadata arguments in RAM.

Details

Many functions will expect arguments such as `meta` and `factors` (possibly `row.factor` or `col.factor`). These functions are expecting the full metadata table for `meta` (which you probably read into R using `read.meta`). The other argument, `factor`, should be a *named* character vector. The values of this vector should be the columns to be used from `meta`, while the names of the vector should be the labels you wish to have displayed in the plots. There are several ways to name a character vector:

```
> my.vec <- c(This = "is", a = "named", character = "vector")
> names(my.vec)
[1] "This"      "a"         "character"
> cat(my.vec)
is named vector
```

Notice that `myvec` has *names* `"This"`, `"a"`, `"character"`, but has *values* `"is"`, `"named"`, `"vector"`. It is the names that will be used to label graphs in `RAM`, but the values that will be used to extract the actual data. This is useful if you have more complicated names; say we want the data from the column named "Precip_14d_before_harvest", but we want a nicer label for the plot. We can do the following:

```
> my.vec <- "Precip_14d_before_harvest"
> names(my.vec) <- "Precipitation (14 d. prior to Harvest, mm)"
```

Now we will be able to plot the value of the `"Precip_14d_before_harvest"` column, but we will have the (much nicer!) label "Precipitation (14 d. prior to Harvest, mm)" appear in our plots.

RAM.input.formatting *Data Input Formatting*

Description

When use some `RAM` functions for the comparison of multiple OTU tables or taxonomic abundance matrices, the user needs to privde all input data sets as list with names being provided.

one data set: data=list(data=otu)

multiple data sets: data=list(data1=otu1, data2=otu2, data3=otu3)

an OTU table: a data frame of otuIDs x sampleIDs with the last column named "taxonomy"

a taxonomy abundance matrix: a data frame of sampleIDs x taxa (e.g. species)

is.OTU: logical, many functions in RAM require the user to set `is.OTU` to be `TRUE` for OTU tables or `FALSE` for a taxonomy abundance matrices.

Author(s)

Wen Chen.

Examples

```
data(ITS1, ITS2, meta)
# use otu tables
matched <- match.data(data=list(otu_ITS1=ITS1, otu_ITS2=ITS2),
           is.OTU=TRUE, meta=meta)
# taxonomy abundance matrices
g1 <- tax.abund(ITS1, rank="g")
g2 <- tax.abund(ITS2, rank="g")
matched <- match.data(data=list(genus_ITS1=g1, genus_ITS2=g2),
           is.OTU=FALSE, meta=meta)
```

RAM.pal *Creat Color Palette*

Description

This function creates color palette, especially if the number of colors required is more than 12.

Usage

```
RAM.pal(cols.needed=20)
```

Arguments

cols.needed	an integer.

Author(s)

Wen Chen

Examples

```
col <- RAM.pal(40)
```

RAM.plotting *Creating Plots with RAM*

Description

This help page details the standards for RAM plotting functions.

Overview

The RAM package contains many functions to produce plots and visualizations for metagenomic data. Currently, the plotting functions are grouped into 'casual' and 'publication' categories. The 'casual' plotting functions only accept a `file` argument and produce a `.tiff` file automatically. They are meant to quickly highlight certain aspects of the data, but are not meant to be published. The 'publication' quality plots accept many more graphing parameters, and should be of suitable quality for future publication. All 'publication' plots should accept the following parameters, in addition to those required to produce the plot:

- `"file"` the file name for the plot.
- `"ext"` the file type for the plot (see below).
- `"height"` the height of the plot, in inches.
- `"width"` the width of the plot, in inches.

Additionally, the following parameters are accepted by some functions:

- `"bw"` should the plot be in black and white?

For 'casual' plots, if `file` is not provided, the plot is displayed to the default graphics device (usually a new window), otherwise a `.tiff` file is created.

For 'publication' plots, if neither `file` nor `ext` are provided, the plot is displayed to the default graphics device (usually a new window). If both `file` and `ext` are provided, a file of type `ext` is created at `file`. If only one of `file` or `ext` is given, an error is raised.

In either case, the file extension will automatically be appended to the end of `file`, if `file` does not already end in the appropriate extension. For example, if `file = ~/my/path.tiff` and `ext = png`, the file will be called `~/my/path.tiff.png`; but if `file = ~/my/path.png`, the file will just be called `~/my/path.png`. Finally, if `file = ~/my/path`, the file will be called `~/my/path.png`.

ggplot2

Furthermore, some of the 'publication' quality plots allow for a `ggplot2` argument. If `ggplot2` is `TRUE`, then the plot will be produced using the `ggplot2` package, and a `ggplot` object will be returned. This allows for additional, personal customization of the plot for those who have used the package before. If `ggplot2` is `FALSE`, then the plot will be created using the base plotting functions.

File Types

For 'publication' quality plots, the following file types are supported (use any of the following values for the `ext` argument): `"pdf"`, `"png"`, `"tiff"`, `"svg"`, `"bmp"`, `"jpeg"`.

Note

If `file` is given without a directrory (e.g. `file` = my_fancy_file), then that file will be created in the current working directory (see `?getwd` and `?setwd` for more information).

The current default resolution is 1000 dpi for all plots.

See Also

`ggplot`

Author(s)

Wen Chen and Joshua Simpson.

`RAM.rank.formatting` *Taxonomic Rank Formatting*

Description

In all `RAM` functions requiring the user to input a taxonomic rank, three different formats for this taxon are accepted. All of these formats are simple character vectors (strings), and are provided for readability and convenience. The user only needs to specify any single element from any of the formats below. The formats are as follows:

Format 1: "kingdom", "phylum", "class", "order", "family", "genus", "species"

Format 2: "k", "p", "c", "o", "f", "g", "s"

Format 3: "k__", "p__", "c__", "o__", "f__", "g__", "s__"

Author(s)

Wen Chen and Joshua Simpson.

See Also

`get.rank`, `tax.abund`

`read.meta` *Open Metadata Table*

Description

Opens the given file and return a data frame representing the metadata. This function use `read.table` to read in data; for large data sets, we recommend `read.meta`.

Usage

```
read.meta(file, sep=",")
```

Arguments

`file`	a character vector specifying the file path to your file.
`sep`	the character used to separate cells in the file.

Value

Returns a data frame with the information from the file. The first row and column are used for the names of the other rows and columns.

Author(s)

Wen Chen and Joshua Simpson

See Also

`read.meta`, `read.table`

Examples

```
  ## Not run:
    my.meta <- read.meta("path/to/meta")

## End(Not run)
```

read.OTU *Open OTU Table*

Description

Opens the given file and returns a data frame representing the OTU table. This function use `read.table` function so is quite slow for large data sets, for which we recommend to use `fread.OTU` instead.

Usage

```
read.OTU(file, sep=",")
```

Arguments

`file`	a character vector specifying the file path to your file.
`sep`	the character used to separate cells in the file.

Value

Returns a data frame with the information from the file. The first row and column are used for the names of the other rows and columns.

Note

The OTU table should only contain classifications for the seven major taxonomic ranks, additional ranks will break some functions in the package. To remove those other classifications, try running

```
sed -i.backup -e 's/s[a-z]__[^;]*; //g' -e 's/t__[^;]*; //g' $FILE
```

where `$FILE` is your OTU table. The letter `t` can be replaced in the second expression for any other letter which appears as a prefix. For example, adding `-e 's/u__[^;]*; //g'` before `$FILE` would remove any entries formatted like `u__test_classification; .`

Additionally, if your OTU table starts with the entry `#OTU ID`, that cell needs to be removed before the table can be read with `read.OTU`.

Author(s)

Wen Chen and Joshua Simpson.

See Also

`getwd`, `setwd`, `read.table`

Examples

```
## Not run:
my.OTU <- read.OTU("path/to/otu", sep=",")
## End(Not run)
```

reset.META *Reset OTU*

Description

This function reset data type of metadata variables.

Usage

```
reset.META(meta, factor=NULL, numeric=NULL, date=NULL)
```

Arguments

meta	data frame. The metadata table to reset variable data type.
factor	a string or charactor vector, containing the column names of metadata variables to be set as factor. reset.META
numeric	a string or charactor vector, containing the column names of metadata variables to be set as numeric.
date	a string or charactor vector containing the column names of metadata variables to be set as date.

Value

This function returns the same metadata with variables being reset to desired data type. Warnings or errors may be raise if the format of the original data cannot be recognized by R.

Author(s)

Wen Chen

Examples

```
data(meta)
str(meta)
## Not run:
# for demonstration purpose only
meta.new <- reset.META(meta, factor=c("Plots"),
                  numeric=c("City", "Province"))
str(meta.new)

## End(Not run)
```

`sample.locations` *Plot the Geographic Location of Samples*

Description

This function consumes an OTU table, along with its associated metadata, and plots all the samples from that data as points on a map. The size of a point indicates the number of counts collected from that sample, while the colour of the point (optionally) shows a metadata factor for that sample.

Usage

```
sample.locations(otu1, otu2=NULL, meta, factor = NULL, zoom = 5,
                 source = "google", labels = c("ITS1", "ITS2"),
                 lat.col = "Latitude", long.col = "Longitude",
                 file = NULL, ext = NULL, height = 10, width = 12)
```

Arguments

`otu1`	the OTU table to be used.
`otu2`	the (optional) second OTU table to be used.
`meta`	the metadata table to be used.
`factor`	(optional) a named character vector of length one specifying a column from the metadata table to be used to colour the points.
`zoom`	a positive integer in the range 3-21 (if `source == "google"`) or 3-18 (if `source == "osm"`) specifying the zoom for the map (low number means zoomed out).
`source`	the source to be used for the map; either "`google`" or "`osm`".
`labels`	a character vector giving one label per OTU.
`lat.col`	the name of the column in `meta` containing the latitude information.
`long.col`	the name of the column in `meta` containing the longitude information.
`file`	the file path where the image should be created (see ?RAM.plotting).
`ext`	the file type to be used; one of "`pdf`", "`png`", "`tiff`", "`bmp`", "`jpg`", or "`svg`".
`height`	the height of the image to be created (in inches).
`width`	the width of the image to be created (in inches).

Details

Please note that this function is getting map information using either the Google Maps API or the OpenStreetMap API, and your usage is subject to the terms of those APIs.

Note

If you are getting a 403/503 error, that likely means that the current map provider is currently unavailable. This can be for a variety of reasons: if `source == "google"`, you have likely maxed out your API call limit (this can be due to multiple users sharing an IP address; contact your system administrator for further details). If `source == "osm"`, the server is likely under heavy load and unable to process your request. You can check the server load online. In either case, the issue will likely resolve itself. Try calling the function again in a few hours.

If you get a warning message of the form "`Removed X rows containing missing values (geom_point).`", this means that the current zoom level is too high to display some or all of the points. Try using a lower value for`zoom`.

Author(s)

Wen Chen and Joshua Simpson.

See Also

`RAM.factors`

Examples

```
data(ITS1, meta)

## Not run:
sample.locations(otu1=ITS1, otu2=ITS2, meta=meta, factor=c(Crop="Crop"))
## End(Not run)
```

---

sample.map *Plot The Geographic Location of Samples*

Description

This function plot the number of samples collected from different locations that are DISTANT from each other, e.g. samples that collected from distant cities. This function is similar but not the same as `sample.locations` and `sample.sites`. The plot will also show the sample size of each location.

Usage

```
sample.map(meta, siteID="City", maptype="roadmap",
           lat="Latitude", lon="Longitude", zoom=3,
           file=NULL, ext=NULL, width=10, height=10)
```

Arguments

meta	the OTU table to be used.
siteID	IDs of sampling sites for each unique pair of longitude and latitude.
maptype	maptype to use, see also get_map.
lat	latitude of each sampling location
lon	longitude of each sampling location
zoom	map zoom, an integer from 3 (continent) to 21 (building). see also get_map.
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).

Details

Please note that this function is getting map information using either the Google Maps API or the OpenStreetMap API, and your usage is subject to the terms of those APIs.

Note

If you are getting a 403/503 error, that likely means that the current map provider is currently unavailable. Try calling the function again in a few hours. If you get a warning message of the form "Removed X rows containing missing values (geom_point).", this means that the current zoom level is too high to display some or all of the points. Try using a lower value forzoom.

Author(s)

Wen Chen.

See Also

sample.locations, sample.sites

Examples

```
data(meta)

## Not run:
sample.map(meta=meta, zoom=8)

## End(Not run)
```

`sample.sites` *Plot The Geographic Location of Samples*

Description

This function plot the number of samples collected from different locations that are close to each other. This function is similar but not the same as `sample.locations` and `sample.map`.

Usage

```
sample.sites(meta, siteID="City", marker.size="small",
             lat="Latitude", lon="Longitude", maptype="hybrid",
             zoom=5, file=NULL, ext=NULL, width=8, height=8)
```

Arguments

`meta`	the OTU table to be used.
`siteID`	IDs of sampling sites for each unique pair of longitude and latitude.
`marker.size`	maptype to use, see also `get_map`.
`lat`	latitude of each sampling location
`lon`	longitude of each sampling location
`maptype`	maptype to use, see also `get_map`.
`zoom`	map zoom, an integer from 3 (continent) to 21 (building). see also `get_map`.
`file`	the file path where the image should be created (see ?RAM.plotting).
`ext`	the file type to be used; one of `"pdf"`, `"png"`, `"tiff"`, `"bmp"`, `"jpg"`, or `"svg"`.
`height`	the height of the image to be created (in inches).
`width`	the width of the image to be created (in inches).

Note

This function is more suitable for plot sampling sites that are close to each other. If you are getting a 403/503 error, that likely means that the current map provider is currently unavailable. Try calling the function again in a few hours. If you get a warning message of the form `"Removed X rows containing missing values (geom_point)."`, this means that the current zoom level is too high to display some or all of the points. Try using a lower value for`zoom`.

Author(s)

Wen Chen.

See Also

`sample.locations`, `sample.map`

Examples

```
data(meta)
## Not run:
sample.sites(meta=meta, zoom=8)

## End(Not run)
```

---

shared.OTU *Summary of Shared OTUs Across ALL Subjects*

Description

This function consumes OTU tables and returns a list summarizing information about the presence of the OTUs in samples.

Usage

```
shared.OTU(data)
```

Arguments

data	a list of OTU tables to be analyzed.

Value

shared.OTU returns a list containing the information calculated. The names associated with the list describe what that number represents; i.e. "#_of_OTUs_in_all_samples" shows how many OTUs in the given table were found to be present in all samples. The last item in the list is a character vector, containing the OTU number and taxonomic information of each OTU which was present in all samples. All entries in that column are of the form "OTU-taxonomic_classification".

Note

The OTUs are determined to be absent/present using the "pa" method from the function decostand.

Author(s)

Wen Chen and Joshua Simpson.

See Also

decostand

Examples

```
data(ITS1)
## Not run:
shared <- shared.OTU(data=list(ITS1=ITS1))
shared <- shared.OTU(data=list(ITS1=ITS1, ITS2=ITS2))

## End(Not run)
```

`shared.Taxa` *Summary of Shared Taxa Across ALL Subjects*

Description

This function consumes OTU tables or a taxonomy matrices and returns a list summarizing information about the presence of the taxa in that table at a given taxonomic rank.

Usage

```
shared.Taxa(data, is.OTU=TRUE, rank="g")
```

Arguments

`data`	a list of OTU tables or taxonomy abundance matrices to be analyzed.
`is.OTU`	whether or not the input data are otu tables
`rank`	the taxonomic rank to be investigated

Value

`shared.Taxa` returns a list containing the information calculated. The names associated with the list describe what that number represents; i.e. "#_of_families_in_all_samples" shows how many taxa at the family level were found to be present in all samples. The last item in the list is a character vector, containing the taxon names of which were present in all samples.

Note

The taxa are determined to be absent/present using the "pa" method from the function `decostand`.

Author(s)

Wen Chen.

See Also

`decostand`

Examples

```
data(ITS1)
shared.Taxa(data=list(ITS1=ITS1))
## Not run:
g1 <- tax.abund(ITS1, rank="g", drop.unclassified=TRUE)
shared.Taxa(data=list(genus_ITS1=g1), rank="g", is.OTU=FALSE)

## End(Not run)
```

---

tax.abund *Aggregate OTU Data Based on Taxonomy*

Description

This function consumes OTU table(s) and (optionally) a taxonomic rank, then extracts the classification of each OTU at the given taxonomic rank, groups by classification at the given rank, removes all groups with only 0 counts, optionally removes all unclassified groups, sorts groups based on abundance, and then returns the transpose.

Usage

```
tax.abund(otu1, otu2=NULL, rank=NULL, drop.unclassified=FALSE,
          top=NULL, count=TRUE, mode="number")
```

Arguments

otu1	the first OTU table to be used.
otu2	the second OTU table to be used.
rank	a character vector representing a rank. Must be in one of three specific formats (see ?RAM.rank.formatting for help). If omitted, the function will repeat for all seven major taxonomic ranks.
drop.unclassified	logical. Determine whether or not the OTUs labelled "unclassified" or missing classification at the given taxonomic rank should be excluded.
top	the number of groups to select, starting with the most abundant. If NULL, all are selected.
count	logical. Should the actual count of each OTU be shown, or should the relative abundances be shown?
mode	a character vector, one of "percent" or "number". If number, then top many groups will be selected. If percent, then all groups with relative abundance in at least one sample above top will be selected.

Value

The value returned by this function may become nested lists, so for convenience, any nested lists have been given descriptive items names to make accessing its elements simple (see Examples).

- If `otu2` is given:
  - If `rank` is given: a list containing two data frames (otu1 and otu2 aggregated at the given rank).
  - If `rank` is not given: a list containing two lists. The first sublist represents `otu1`, the second `otu2`. The sublists contain seven data frames, the aggregation of the data at each taxonomic rank (see Examples).
- If `otu2` is not given:
  - If `rank` is given: a single data frame (otu1 aggregated at the given rank).
  - If `rank` is not given: a list containing seven data frames (otu1 aggregated at each taxonomic rank).

Author(s)

Wen Chen and Joshua Simpson.

See Also

RAM.rank.formatting

Examples

```
data(ITS1, ITS2)
# aggregate based on phylum
ITS1.p <- tax.abund(ITS1, rank="p")

# aggregate based on all ranks; ignoring all unclassified OTUs
ITS1.taxa <- tax.abund(ITS1, drop.unclassified=FALSE)

# aggregate for one rank for both ITS1 and ITS2
lst <- tax.abund(ITS1, ITS2, rank="class")

# aggregate for all ranks for both ITS1 and ITS2
lst.all <- tax.abund(ITS1, ITS2)

stopifnot(identical(lst.all$otu1$phylum, ITS1.p))

# get the counts for all genera with relative abundance > 25%
tax.abund(ITS1, rank="g", top=25, mode="percent", count=TRUE)
```

`tax.fill` *Fill Missing Taxonomic Information*

Description

This function consumes an OTU table and returns a new OTU table where the taxonomic classifications which are unidentified, unclassified, incertae sedis, or simply missing, are replaced with a more descriptive entry.

Usage

```
tax.fill(data, downstream = TRUE)
```

Arguments

`data`	the OTU table to be used.
`downstream`	logical. Should the replacement occur downstream, or upstream? (see Details)

Details

If `downstream == TRUE`, the function will start at the kingdom level and work its way down. Whenever an invalid group is encountered (i.e. one of "unclassified", "unidentified", "incertae_sedis", or simply missing, ignoring capitalization), the last known 'good' group will be substituted in the form "`p__belongs_to_k_Fungi`." If `downstream == FALSE`, the function will begin at the species level and work up.

This example should help clarify: given the taxonomy "k__Fungi; p__unidentified; c__Tremellomycetes", the new taxonomy is as follows (recall that an OTU table is required as input, and will be returned as output; this example simply shows the effect on the taxonomy):

- Downstream (Kingdom -> Species): "k__Fungi; p__belongs_to_k_Fungi; c__Tremellomycetes; o__belongs_to_c_Tremellomycetes; f__belongs_to_c_Tremellomycetes; g__belongs_to_c_Tremellomycetes; s__belongs_to_c_Tremellomycetes"
- Upstream (Species -> Kingdom): "k__Fungi; p__belongs_to_c_Tremellomycetes; c__Tremellomycetes; o__belongs_to_no_taxonomy; f__belongs_to_no_taxonomy; g__belongs_to_no_taxonomy; s__belongs_to_no_taxonomy"

Usually, `downstream = TRUE` will provide a more useful output, however if the species is often known for your data, but other ranks are unknown, `downstream = FALSE` will provide a more descriptive taxonomy.

Value

Returns an OTU table as a data frame with the exact same numerical counts as `data`, but an updated `taxonomy` column.

Author(s)

Wen Chen and Joshua Simpson.

See Also

RAM.rank.formatting

Examples

```
data(ITS1)

#\code{filter.OTU} returns a list
otu <- filter.OTU(data=list(ITS1=ITS1), percent=0.001)[[1]]

tax.fill(otu)
```

---

tax.split *Split OTU Tables By Taxonomic Rank*

Description

This function consumes an OTU table and splits its taxonomy columns into the seven major taxonomic ranks. It returns a data frame preserving all numerical data, but changing the 'taxonomy' column to the name of the appropriate rank, and preserving only the classifications at that rank.

Usage

```
tax.split(otu1, otu2 = NULL, rank = NULL)
```

Arguments

otu1	the first OTU table to be used.
otu2	the second OTU table to be used.
rank	the (optional) rank to split at and return (see ?RAM.rank.formatting for formatting details).

Value

The value returned by this function may become nested lists, so for convenience, any nested lists have been given descriptive items names to make accessing its elements simple (see Examples).

- If otu2 is given:
  - If rank is given: a list containing two data frames (otu1 and otu2 split at the given rank).
  - If rank is not given: a list containing two lists. The first sublist represents otu1, the second otu2. The sublists contain seven data frames, which are the data split at each taxonomic rank (see Examples).
- If otu2 is not given:
  - If rank is given: a single data frame (otu1 split at the given rank).
  - If rank is not given: a list containing seven data frames (otu1 split at each taxonomic rank).

Note

This function may seem similar to `get.rank`, but they are distinct. `get.rank` only returns the entries classified at that taxonomic rank, and so some OTUs might be omitted in the returned data frame. With `tax.split`, it is guaranteed that all OTUs will be preserved in the returned data table (although they may be missing taxonomic classification at that rank).

If no OTUs are classified at the given rank, the taxonomy column for that rank will be filled with empty strings.

Author(s)

Wen Chen and Joshua Simpson.

See Also

`get.rank`

Examples

```
data(ITS1, ITS2)

# split only ITS1 data at a single rank
ITS1.tax.p <- tax.split(ITS1, rank="phylum")

# split only ITS1 data at all ranks
ITS1.tax.all <- tax.split(ITS1)

# split ITS1 and IST2 data at a given rank
lst <- tax.split(ITS1, ITS2, rank="c")

# split ITS1 and ITS2 at every rank
lst.all <- tax.split(ITS1, ITS2)

stopifnot(identical(lst.all$otu1$phylum, ITS1.tax.p))
```

Taxa.ord *Ordination Plot For Taxa Groups Using CCA or RDA Analysis*

Description

This function consumes an ecology data set, metadata factors, and graphing options, then produces a plot showing the vegan::cca or vegan::rda analysis.

Usage

```
Taxa.ord(data, is.OTU=TRUE, meta=meta, factors=NULL,
         group=NULL, rank="g", taxa=10, data.trans="total",
         plot.species=TRUE, plot.scaling=-1,
         biplot.scale=NULL, biplot.sig=NULL, biplot.label= TRUE,
```

```
mode=c("rda", "cca"), choice=c(1,2), main="",
cex.point=3, cex.label=1, cex.leg=12, cex.bp=3, cex.text=3,
file=NULL, ext=NULL, width=10, height=10)
```

Arguments

data	an ecology data set, either an otu table or a taxonomy abundance matrix.
is.OTU	whether or not the data an otu table
meta	the metadata table to be used.
factors	a named character vector of length 1 or 2 specifying metadata factors for the samples in the OTU table (see Details).
group	a named character vector of length 1 or 2 specifying metadata factors for the samples in the OTU table (see Details).
rank	the rank to select the taxon groups at.
taxa	an integer or a character vector of taxa names at the given rank. if integer, plot the top most abundant taxa, otherwise plot the taxa in the vector.
data.trans	a method used to standardize the OTU table. One of "total", "max", "freq", "normalize", "range", "standardize", "pa", "chi.square", "hellinger" or "log" (see ?decostand).
plot.species	whether plot sites or taxa, should be reflex to plot.scaling
plot.scaling	one of the following: 1, 2, 3, or -1. See scaling in plot.cca for detail. See also ordiplot
biplot.scale	a numeric number, length of the biplot arrows
biplot.sig	significance cutoff for biplot to be displayed. Currently disabled because in the function, ordination model calcuated cannot be passed to anova test.
biplot.label	whether or not to plot biplot
mode	one of the following: "cca" or "rda".
choice	the chosen axes
main	title of the plot
cex.point	size of points
cex.label	size of taxa lables
cex.leg	size of lengend name
cex.text	size of taxon names if plot.species is set TRUE
cex.bp	size of biplot labels
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
width	the width of the image to be created (in inches).
height	the height of the image to be created (in inches).

Details

group should be a named character vector specifying the names of the columns to be used from meta (see RAM.factors). The values on the axes denote what fraction of the sum of all eigenvalues (i.e. from all axes) is explained by that (single) axis.

Value

return a list of following: 1) ggplot object; 2) ordination model; 3) commodity data and 4) metadata used for the ordination model.

Note

The labels for the taxa points are placed above, below, or next to the point itself at random. If labels are outside of the plotting area, or overlapping with each other, run your command again (without changing any arguments!) and the labels should move to new positions. Repeat until they are placed appropriately. This is done to ensure even tightly-grouped samples, or samples near the edge of the plot, have their labels shown. If the labels are too distracting, remember that they can be turned off by setting plot.species = FALSE.

Author(s)

Wen Chen.

See Also

decostand, OTU.ord, pcoa.plot

Examples

```
data(ITS1, meta)
its1<- filter.OTU(data=list(ITS1=ITS1), percent=0.001)[[1]]
factors=c("City", "Crop", "Harvestmethod", "Ergosterol_ppm")
## Not run:
ord <- Taxa.ord(its1,  meta=meta, data.trans="total",
        factors=factors, mode="cca", biplot.sig=0.1,
        taxa=20, biplot.scale=1.5, cex.point=5, cex.label=1,
        plot.species=TRUE, rank="g", plot.scaling=3,
        group=c(City="City", Crop="Crop"), biplot.label=FALSE)
names(ord)

## End(Not run)
```

theme_ggplot *Customized Themes For GGPLOT*

Description

RAM provides some customized ggplot themes to spice up your plots for presentations, but some of these addtional features might be distractive and not be ideal for publications

Author(s)

Wen Chen

See Also

ggplot

Examples

```
## Not run:
data(ITS1, ITS2, meta)
data <- list(ITS1=ITS1, ITS2=ITS2)
# dissim.alleig.plot returns a ggplot2 object:
my.eig.plot <- dissim.alleig.plot(data)
my.eig.plot # view the plot

# update ggplot theme
require("grid")
new_theme <-RAM.color()
my.eig.plot <- my.eig.plot + new_theme
my.eig.plot

## End(Not run)
```

top.groups.plot *Plot the Top Taxon Groups*

Description

These functions consume two OTU tables, along with (optionally) a file name and a parameter `top`. They create a box plot of the `top` number of OTUs (for `plot.top.number`), or all OTUs with relative abundance above `top` percent (for `plot.top.percent`) at the taxonomic ranks phylum, class, order, family, and genus.

Usage

```
group.top.number(data, top=10, ranks=c("p","c","o","f","g"),
                 drop.unclassified=FALSE, cex.x=NULL,
                 main=NULL,file=NULL, ext=NULL, height=8, width=16,
                 bw=FALSE, ggplot2=TRUE)

group.top.percent(data, top=10, ranks=c("p","c","o","f","g"),
                  drop.unclassified=FALSE, cex.x=NULL,
                  main=NULL, file=NULL, ext=NULL, height=8,
                  width=16, bw=FALSE, ggplot2=TRUE)
```

Arguments

data	a list of OTU tables.
top	the number of OTUs to select (for top.number), or the minimum relative abundance threshold to use for selecting OTUs (for top.percent).
ranks	a vector of the taxonomic ranks. See also RAM.rank.formatting
drop.unclassified	logical. Should OTUs labelled "unclassified" or missing classification at the given taxonomic rank be excluded?
cex.x	optional. The size of the x axis names.
main	the title of the plot
file	the file path where the image should be created (see ?RAM.plotting).
ext	the file type to be used; one of "pdf", "png", "tiff", "bmp", "jpg", or "svg".
height	the height of the image to be created (in inches).
width	the width of the image to be created (in inches).
bw	logical. Should the image be created in black and white?
ggplot2	logical. Should the ggplot2 package be used to produce the plot, or should the base graphics be used? (see ?RAM.plotting).

Note

Please be aware that the 'whiskers' in the plot may differ depending on the setting of ggplot2. Please see geom_boxplot boxplot, and boxplot.stats for more information on how the whiskers are calculated.

Author(s)

Wen Chen and Joshua Simpson.

See Also

RAM.plotting

Examples

```
## Not run:
data(ITS1, ITS2)

# plots the top 10 OTUs (by default) at five ranks
group.top.percent(data=list(ITS1=ITS1, ITS2=ITS2), top=10)

# plots all OTUs w/ relative abundance > 10% (as specified) at
# five ranks and saves the result as a .tiff file
# (only using ITS1 data)
group.top.percent(data=list(ITS1=ITS1, ITS2=ITS2), top=10,
                  file="my/file/path", ext="tiff")
## End(Not run)
```

`transpose.LCA` *Transpose OTU Tables With LCA Annotation For Each OTU*

Description

Similar to `transpose.OTU`, but each OTU is annotated by the lowest common ancestor it was assigned to.

Usage

```
transpose.LCA(data)
```

Arguments

`data`	The OTU tables to be transposed. See also `RAM.input.formatting`.

Value

Returns a transposed OTU table, but the colname is formatted as: LCA_otuID.

Author(s)

Wen Chen.

Examples

```
data(ITS1, ITS2)
## Not run:
lca.t <- transpose.LCA(data=list(ITS1=ITS1, ITS2=ITS2))

## End(Not run)
```

transpose.OTU *Take the Transpose of an OTU Table*

Description

Returns the transpose of the given OTU table, excluding the last column (which should contain taxonomic information).

Usage

```
transpose.OTU(data)
```

Arguments

data	The OTU table to be transposed.

Value

Returns a data frame with rows equal to the columns of the original OTU, and columns equal to the rows of the original OTU. (Excluding the taxonomy column).

Author(s)

Wen Chen and Joshua Simpson.

Examples

```
data(ITS1)

ITS1.t <- transpose.OTU(ITS1)
```

valid.OTU *Validate an OTU Table*

Description

This function consumes one or two OTU tables and checks if they are formatted properly and contain valid data.

Usage

```
valid.OTU(otu1, otu2 = NULL)
```

Arguments

otu1	the first OTU table to check.
otu2	the second OTU table to check.

Value

If the table is not valid, an error will be raised with a description explaining the problem. If the table is valid, NULL will be returned invisibly.

Author(s)

Dr. Chen Wen and Joshua Simpson.

Examples

```
data(ITS1, ITS2)

valid.OTU(ITS1)
valid.OTU(ITS1, ITS2)
```

---

valid.taxonomy *Validate And Reformat The OTU Taxonomy Column*

Description

A properly formatted taxonomy column of an otu table is critical for RAM functions to run properly. The taxonomy column of an otu table is composed of taxonomic lineages for otuIDs. RAM accept 7 ranks, including kingdom, phylum, class, order, family, genus and species, sub ranks are not supported. Taxa names at each rank should have prefix as "k__", "p__", "c__", "o__", __", "g__", and "s__", each rank should be separated by "; ", i.e. a semi colon and a white space, NOT just ";".

This function will check the format of the taxonomy column of the input otu table and give suggetions that whether or not it needs to be reformatted using reformat.taxonomy of RAM.

However, RAM does accept missing ranks in lineages, as long as each rank is separated by "; " with proper prefix.

Usage

```
valid.taxonomy(data)
reformat.taxonomy(data, input.ranks, sep="; ")
```

Arguments

data	a list of otu tables, see RAM.input.formatting.
input.ranks	the ranks of the taxonomic lineages in the input otu tables.
sep	the separator between each taxonomic rank in the lineage.

Author(s)

Wen Chen.

See Also

get.rank, tax.abund

Examples

```
data(ITS1)
## Not run:
# for demonstration purpose only
# the ITS1 dataset missing species rank, but it's ok
# the problematic taxonomy lineages are those missing proper prefix
# at each rank
# see ?RAM.rank.formatting
valid.taxonomy(data=list(ITS1=ITS1))
input.ranks <- c("kingdom", "phylum", "class", "order", "family",
                 "genus")
reform.ITS1 <- reformat.taxonomy(list(ITS1=ITS1),
                                 input.ranks=input.ranks,
                                 sep="; ")[[1]]


## End(Not run)
```

---

write.data *Write Data To CSV File*

Description

Creates a .csv-formatted file with the data. The file will be created in your current working directory (see ?getwd and ?setwd), unless specified otherwise by `file`. Note that if the `file` field does not end in .csv, ".csv" will be appended to the end of `file`.

Usage

```
write.data(data, file)
```

Arguments

`data`	a data frame or matrix etc.
`file`	The name of the .csv file to be created.

Author(s)

Wen Chen and Joshua Simpson.

See Also

write.csv, getwd, setwd

Examples

```
data(ITS1)
## Not run:
write.data(ITS1, "my_file_name.csv")

## End(Not run)
```

Index