# Package ‘phyloseq’

March 26, 2013

**Version** 1.2.1

**Date** 2013-01-23

**Title** Handling and analysis of high-throughput phylogenetic sequence data.

**Description** phyloseq provides a set of classes and tools to facilitate the import, storage, analysis, and graphical display of phylogenetic sequencing data.

**Maintainer** Paul J. McMurdie <mcmurdie@stanford.edu>

**Author** Paul J. McMurdie <mcmurdie@stanford.edu>, Susan Holmes <susan@stat.stanford.edu>

**License** AGPL-3

**Imports** ape (>= 3.0), foreach (>= 1.3), igraph0 (>= 0.5), multtest (>= 2.8), plyr (>= 1.7.1), reshape (>= 0.8.4), RJSONIO (>= 0.98),scales (>= 0.2.2), vegan (>= 2.0)

**Depends** R (>= 2.15.0), methods, ade4 (>= 1.4), ggplot2 (>= 0.9.2),picante (>= 1.3)

**Suggests** genefilter, testthat

**Enhances** doParallel (>= 1.0)

**biocViews** Clustering, Classification, MultipleComparisons,QualityControl, GeneticVariability, HighThroughputSequencing

**URL** http://joey711.github.com/phyloseq/

**BugReports** https://github.com/joey711/phyloseq/issues

**Collate** 'allClasses.R' 'allPackage.R' 'allData.R' 'as-methods.R''show-methods.R' 'plot-methods.R' 'extract-methods.R''almostAllAccessors.R' 'otuTable-class.R' 'phyloseq-class.R''taxonomyTable-class.R' 'IO-methods.R' 'merge-methods.R''multtest-wrapper.R' 'ordination-methods.R''transform_filter-methods.R' 'validity-methods.R''assignment-methods.R' 'sampleData-class.R' 'extend_vegan.R''network-methods.R' 'distance-methods.R''deprecated_functions.R'

## R topics documented:

---

phyloseq-package *Handling and analysis of high-throughput phylogenetic sequence data.*

---

## Description

There are already several ecology and phylogenetic packages available in R, including the adephylo, vegan, ade4, picante, ape, phangorn, phylobase, and OTUbase packages. These can already take advantage of many of the powerful statistical and graphics tools available in R. However, prior to *phyloseq* a user must devise their own methods for parsing the output of their favorite OTU clustering application, and, as a consequence, there is also no standard within Bioconductor (or R generally) for storing or sharing the suite of related data objects that describe a phylogenetic sequencing project. The phyloseq package seeks to address these issues by providing a related set of S4 classes that internally manage the handling tasks associated with organizing, linking, storing, and analyzing phylogenetic sequencing data. *phyloseq* additionally provides some convenience wrappers for input from common clustering applications, common analysis pipelines, and native implementation of methods that are not available in other R packages.

## Author(s)

Paul J. McMurdie II <mcmurdie@stanford.edu>

## References

www.stanford.edu/~mcmurdie

---

access *Universal slot accessor function for phyloseq-class.*

---

## Description

This function is used internally by many accessors and in many functions/methods that need to access a particular type of component data. If something is wrong, or the slot is missing, the expected behavior is that this function will return NULL. Thus, the output can be tested by is.null as verification of the presence of a particular data component. Unlike the component-specific accessors (e.g. otu_table, or phy_tree), the default behavior is not to stop with an error if the desired slot is empty. In all cases this is controlled by the errorIfNULL argument, which can be set to TRUE if an error is desired.

## Usage

```
access(physeq, slot, errorIfNULL=FALSE)
```

## Arguments

| | |
|---|---|
| physeq | (Required). phyloseq-class. |
| slot | (Required). A character string indicating the slot (not data class) of the component data type that is desired. |
| errorIfNULL | (Optional). Logical. Should the accessor stop with an error if the slot is empty (NULL)? Default FALSE. |

## Value

Returns the component object specified by the argument slot. Returns NULL if slot does not exist. Returns physeq as-is if it is a component class that already matches the slot name.

### See Also

getslots.phyloseq, merge_phyloseq

### Examples

```
#
## data(GlobalPatterns)
## access(GlobalPatterns, "tax_table")
## access(GlobalPatterns, "phy_tree")
## access(otu_table(GlobalPatterns), "otu_table")
## # Should return NULL:
## access(otu_table(GlobalPatterns), "sample_data")
## access(otuTree(GlobalPatterns), "sample_data")
## access(otuSam(GlobalPatterns), "phy_tree")
```

---

## data-enterotype *(Data) Enterotypes of the human gut microbiome (2011)*

---

### Description

Published in Nature in early 2011, this work compared (among other things), the faecal microbial communities from 22 subjects using complete shotgun DNA sequencing. Authors further compared these microbial communities with the faecal communities of subjects from other studies. A total of 280 faecal samples / subjects are represented in this dataset, and 553 genera. The authors claim that the data naturally clumps into three community-level clusters, or “enterotypes”, that are not immediately explained by sequencing technology or demographic features of the subjects, but with potential relevance to understanding human gut microbiota.

### Details

abstract from research article (quoted):

Our knowledge of species and functional composition of the human gut microbiome is rapidly increasing, but it is still based on very few cohorts and little is known about variation across the world. By combining 22 newly sequenced faecal metagenomes of individuals from four countries with previously published data sets, here we identify three robust clusters (referred to as enterotypes hereafter) that are not nation or continent specific. We also confirmed the enterotypes in two published, larger cohorts, indicating that intestinal microbiota variation is generally stratified, not continuous. This indicates further the existence of a limited number of well-balanced host-microbial symbiotic states that might respond differently to diet and drug intake. The enterotypes are mostly driven by species composition, but abundant molecular functions are not necessarily provided by abundant species, highlighting the importance of a functional analysis to understand microbial communities. Although individual host properties such as body mass index, age, or gender cannot explain the observed enterotypes, data-driven marker genes or functional modules can be identified for each of these host properties. For example, twelve genes significantly correlate with age and three functional modules with the body mass index, hinting at a diagnostic potential of microbial markers.

(end quote)

### Author(s)

Arumugam, M., Raes, J., et al.

## References

Arumugam, M., et al. (2011). Enterotypes of the human gut microbiome.

Nature, 473(7346), 174-180.

http://www.nature.com/doifinder/10.1038/nature09944 See supplemental information for subject data.

OTU-clustered data was downloaded from the publicly-accessible:

http://www.bork.embl.de/Docu/Arumugam_et_al_2011/downloads.html

## Examples

```
# Try simple network-analysis plot
data(enterotype)
ig <- make_network(enterotype, "samples", max.dist=0.3)
plot_network(ig, enterotype, color="SeqTech", shape="Enterotype", line_weight=0.3, label=NULL)
#
# Filter samples that don't have Enterotype
x <- subset_samples(enterotype, !is.na(Enterotype))
#
# Alternatively. . .
ent.cca <- ordinate(x ~ Enterotype, "CCA")
plot_ordination(x, ent.cca, color="Enterotype")
plot_ordination(x, ent.cca, "biplot")
plot_ordination(x, ent.cca, "split", color="Enterotype")
#
# # multiple testing of genera correlating with enterotype 2
# mt(x, data.frame(sample_data(x))[, "Enterotype"]==2)
# # Should return a data.frame, with the following head()
# # # # # index    teststat   rawp   adjp plower
# # # Prevotella                  207 11.469961374 0.0001 0.0088 0.0001
# # # Bacteroides                  203 -9.015717540 0.0001 0.0088 0.0001
# # # Holdemania                    201 -5.810081084 0.0001 0.0088 0.0001
# # # Acetivibrio                 156 -5.246137207 0.0001 0.0088 0.0001
```

---

data-esophagus *(Data) Small example dataset from a human esophageal community (2004)*

---

## Description

Includes just 3 samples, 1 each from 3 subjects. Although the research article mentions 4 subjects, only 3 are included in this dataset.

## Details

abstract from research article (quoted):

The esophagus, like other luminal organs of the digestive system, provides a potential environment for bacterial colonization, but little is known about the presence of a bacterial biota or its nature. By using broad-range 16S rDNA PCR, biopsies were examined from the normal esophagus of four human adults. The 900 PCR products cloned represented 833 unique sequences belonging to 41 genera, or 95 species-level operational taxonomic units (SLOTU); 59 SLOTU were homologous with culture-defined bacterial species, 34 with 16S rDNA clones, and two were not homologous with any known bacterial 16S rDNA. Members of six phyla, Firmicutes, Bacteroides, Actinobacteria, Proteobacteria, Fusobacteria, and TM7, were represented. A large majority of clones belong to 13 of the 41 genera (783/900, 87%), or 14 SLOTU (574/900, 64%) that were shared by all four persons. Streptococcus (39%), Prevotella (17%), and Veilonella (14%) were most prevalent. The present study identified 56-79% of SLOTU in this bacterial ecosystem. Most SLOTU of esophageal biota are similar or identical to residents of the upstream oral biota, but the major distinction is that a large majority (82%) of the esophageal bacteria are known and cultivable. These findings provide evidence for a complex but conserved bacterial population in the normal distal esophagus.

(end quote)

A description of the 16S rRNA sequence processing can be found on the mothur-wiki at the link below. A cutoff of 0.10 was used for OTU clustering in that example, and it is taken here as well to create example data, esophagus, which was easily imported with the import_mothur() function.

### Author(s)

Pei et al. <zhiheng.pei@med.nyu.edu>

### References

Pei, Z., Bini, E. J., Yang, L., Zhou, M., Francois, F., & Blaser, M. J. (2004). Bacterial biota in the human distal esophagus. Proceedings of the National Academy of Sciences of the United States of America, 101(12), 4250-4255. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC384727

mothur-processed files and the sequence data can be downloaded from a zip-file, along with additional description, from the following URL: http://www.mothur.org/wiki/Esophageal_community_analysis

### Examples

```
# Example using esophagus-data in a UniFrac calculation.
data(esophagus)
UniFrac(esophagus, weighted=TRUE)
UniFrac(esophagus, weighted=FALSE)
#
# How to re-create the esophagus dataset using import_mothur function
mothlist  <- system.file("extdata", "esophagus.fn.list.gz", package="phyloseq")
mothgroup <- system.file("extdata", "esophagus.good.groups.gz", package="phyloseq")
mothtree  <- system.file("extdata", "esophagus.tree.gz", package="phyloseq")
show_mothur_list_cutoffs(mothlist)
cutoff    <- "0.10"
esophman  <- import_mothur(mothlist, mothgroup, mothtree, cutoff)
identical(esophagus, esophman)
```

---

## data-GlobalPatterns *(Data) Global patterns of 16S rRNA diversity at a depth of millions of sequences per sample (2011)*

---

### Description

Published in PNAS in early 2011. This work compared the microbial communities from 25 environmental samples and three known “mock communities” – a total of 9 sample types – at a depth averaging 3.1 million reads per sample. Authors were able to reproduce diversity patterns seen in many other published studies, while also invesitigating technical issues/bias by applying the same techniques to simulated microbial communities of known composition.

## Details

abstract from research article (quoted):

The ongoing revolution in high-throughput sequencing continues to democratize the ability of small groups of investigators to map the microbial component of the biosphere. In particular, the coevolution of new sequencing platforms and new software tools allows data acquisition and analysis on an unprecedented scale. Here we report the next stage in this coevolutionary arms race, using the Illumina GAIIx platform to sequence a diverse array of 25 environmental samples and three known “mock communities” at a depth averaging 3.1 million reads per sample. We demonstrate excellent consistency in taxonomic recovery and recapture diversity patterns that were previously reported on the basis of metaanalysis of many studies from the literature (notably, the saline/nonsaline split in environmental samples and the split between host-associated and free-living communities). We also demonstrate that 2,000 Illumina single-end reads are sufficient to recapture the same relationships among samples that we observe with the full dataset. The results thus open up the possibility of conducting large-scale studies analyzing thousands of samples simultaneously to survey microbial communities at an unprecedented spatial and temporal resolution.

(end quote)

Many thanks to J. Gregory Caporaso for directly providing the OTU-clustered data files for inclusion in this package.

## Author(s)

Caporaso, J. G., et al.

## References

Caporaso, J. G., et al. (2011). Global patterns of 16S rRNA diversity at a depth of millions of sequences per sample. PNAS, 108, 4516-4522. PMCID: PMC3063599

The primary article can be viewed/downloaded at: http://www.pnas.org/content/108/suppl.1/4516.short

## See Also

The examples on the phyloseq wiki page for plot_ordination show many more examples:

https://github.com/joey711/phyloseq/wiki/plot_ordination

## Examples

```
data(GlobalPatterns)
# Remove unobserved taxa
GP0   <- prune_species(taxa_sums(GlobalPatterns)>0, GlobalPatterns)
# Perform ordination (in this case, detrended correspondence analysis)
gpdca <- ordinate(GP0, "DCA")
# Create plot of samples
plot_ordination(GP0, gpdca, color="SampleType", title="DCA on abundances, first two axes")
# # More complicated plot facetting by phylum.
# library("ggplot2")
# plot_ordination(GP0, gpdca, color="SampleType", title="DCA on abundances, first two axes") + geom_line()
# plot_ordination(GP0, gpdca, "taxa", color="Kingdom") + facet_wrap(~Phylum, 8)
```

data-soilrep *(Data) Reproducibility of soil microbiome data (2011)*

## Description

Published in early 2011, this work compared 24 separate soil microbial communities under four treatment conditions via multiplexed/barcoded 454-pyrosequencing of PCR-amplified 16S rRNA gene fragments. The authors found differences in the composition and structure of microbial communities between soil treatments. As expected, the soil microbial communities were highly diverse, with a staggering 16,825 different OTUs (species) observed in the included dataset. Interestingly, this study used a larger number of replicates than previous studies of this type, for a total of 56 samples, and the putatively low resampling rate of species between replicated sequencing trials ("OTU overlap") was a major concern by the authors.

## Details

This dataset contains an experiment-level (phyloseq-class) object, which in turn contains the taxa-contingency table and soil-treatment table as otu_table-class and sample_data-class components, respectively.

This data was imported from raw files supplied directly by the authors via personal communication for the purposes of including as an example in the phyloseq-package. As this data is sensitive to choices in OTU-clustering parameters, attempts to recreate the otu_table from the raw sequencing data may give slightly different results than the table provided here.

abstract from research article (quoted):

To determine the reproducibility and quantitation of the amplicon sequencing-based detection approach for analyzing microbial community structure, a total of 24 microbial communities from a long-term global change experimental site were examined. Genomic DNA obtained from each community was used to amplify 16S rRNA genes with two or three barcode tags as technical replicates in the presence of a small quantity (0.1% wt/wt) of genomic DNA from Shewanella oneidensis MR-1 as the control. The technical reproducibility of the amplicon sequencing-based detection approach is quite low, with an average operational taxonomic unit (OTU) overlap of 17.2%+/-2.3% between two technical replicates, and 8.2%+/-2.3% among three technical replicates, which is most likely due to problems associated with random sampling processes. Such variations in technical replicates could have substantial effects on estimating beta-diversity but less on alpha-diversity. A high variation was also observed in the control across different samples (for example, 66.7-fold for the forward primer), suggesting that the amplicon sequencing-based detection approach could not be quantitative. In addition, various strategies were examined to improve the comparability of amplicon sequencing data, such as increasing biological replicates, and removing singleton sequences and less-representative OTUs across biological replicates. Finally, as expected, various statistical analyses with preprocessed experimental data revealed clear differences in the composition and structure of microbial communities between warming and non-warming, or between clipping and non-clipping. Taken together, these results suggest that amplicon sequencing-based detection is useful in analyzing microbial community structure even though it is not reproducible and quantitative. However, great caution should be taken in experimental design and data interpretation when the amplicon sequencing-based detection approach is used for quantitative analysis of the beta-diversity of microbial communities.

(end quote)

### Author(s)

Jizhong Zhou, et al.

### References

Zhou, J., Wu, L., Deng, Y., Zhi, X., Jiang, Y.-H., Tu, Q., Xie, J., et al. Reproducibility and quantitation of amplicon sequencing-based detection. The ISME Journal. (2011) 5(8):1303-1313. doi:10.1038/ismej.2011.11

The article can be accessed online at http://www.nature.com/ismej/journal/v5/n8/full/ismej201111a.html

### Examples

```
# Load the data
data(soilrep)
########################################################
# Alpha diversity (richness) example. Accept null hypothesis:
# No convincing difference in species richness between warmed/unwarmed soils.
########################################################
# Graphically compare richness between the different treatments.
man.col <- c(WC="red", WU="brown", UC="blue", UU="darkgreen")
(p <- plot_richness_estimates(soilrep, x="Treatment", color="Treatment") )
# Add boxplots using ggplot2
# library(ggplot2)
# p + geom_boxplot() + scale_color_manual(values=man.col)
# # # # # # # # # # # # # # # # # # # # # # # # # # # #
# The treatments do not appear to have affected the
# estimated total richness between warmed/unwarmed soil samples
# Test this formally:
DF <- data.frame(sample_data(soilrep), estimate_richness(soilrep) )
t.test(x=subset(DF, warmed=="yes")[, "S.chao1"], y=subset(DF, warmed=="no")[, "S.chao1"])
########################################################
# A beta diversity comparison.
########################################################
# Perform non-metric multidimensional scaling, using Bray-Curtis distance
soil.NMDS <- ordinate(soilrep, "NMDS", "bray")
(p <- plot_ordination(soilrep, soil.NMDS, "samples", color="Treatment") )
# Additional formatting using ggplot2
# library(ggplot2)
# ( p <- p + geom_point(size=5, alpha=0.5) + facet_grid(warmed ~ clipped) )
```

---

distance *General distance / dissimilarity index calculator*

---

### Description

Takes a phyloseq-class object and method option, and returns a distance object suitable for certain ordination methods and other distance-based analyses. There are currently 44 explicitly supported method options, as well as user-provided arbitrary methods via an interface to designdist. For the complete list of currently supported options/arguments to the method parameter, type distance("list") at the command-line. Only sample-wise distances are currently supported (the type argument), but eventually species-wise (OTU-wise) distances will be supported as well.

## Usage

```
distance(physeq, method="unifrac", type="samples", ...)
```

## Arguments

| | |
|---|---|
| physeq | (Required). A phyloseq-class or an otu_table-class object. The latter is only appropriate for methods that do not require any additional data (one-table). For example, the "unifrac" option (UniFrac) requires phyloseq-class that contains both an otu_table and a phylogenetic tree (phylo). |
| method | (Optional). A character string. Default is "unifrac". Provide one of the 44 currently supported options. To see a list of supported options, enter the following into the command line: distance("list") For further details and additional arguments, see the documentation for the supprting functions, linked below under "See Also". In particular, there are three methods included by the phyloseq-package, and accessed by the following method options: "unifrac", for UniFrac based distances, UniFrac; "dpcoa", sample-wise distance from Double Principle Coordinate Analysis, DPCoA; "jsd", for Jensen-Shannon Divergence, JSD; and it is recommended that you see their documentation for details, references, background and examples for use. Alternatively, you can provide a character string that defines a custom distance method, if it has the form described in designdist. |
| type | (Optional). A character string. The type of pairwise comparisons being calculated: sample-wise or taxa-wise. The default is c("samples"). |
| ... | Additional arguments passed on to the appropriate distance function, determined by the method argument. |

## Details

Depending on the method argument, distance() wraps one of UniFrac, DPCoA, JSD, vegdist, betadiver, designdist, or dist.

## Value

An object of class "dist" suitable for certain ordination methods and other distance-based analyses.

## See Also

plot_ordination, UniFrac, DPCoA, JSD, vegdist, betadiver, designdist, dist.

## Examples

```
data(esophagus)
distance(esophagus) # Unweighted UniFrac
distance(esophagus, weighted=TRUE) # weighted UniFrac
distance(esophagus, "jaccard") # vegdist jaccard
distance(esophagus, "gower") # vegdist option "gower"
distance(esophagus, "g") # designdist method option "g"
distance(esophagus, "minkowski") # invokes a method from the base dist() function.
distance(esophagus, "(A+B-2*J)/(A+B)") # designdist custom distance
distance("help")
distance("list")
help("distance")
```

## DPCoA — *Calculate Double Principle Coordinate Analysis (DPCoA) using phylogenetic distance*

### Description

Function uses abundance (otu_table-class) and phylogenetic (phylo) components of a phyloseq-class experiment-level object to perform a Double Principle Coordinate Analysis (DPCoA), relying heavily on the underlying (and more general) function, dpcoa. The distance object ultimately provided as the cophenetic/patristic (cophenetic.phylo) distance between the species.

### Usage

```
DPCoA(physeq, correction=cailliez, scannf=FALSE, ...)
```

### Arguments

| | |
|---|---|
| physeq | (Required). A phyloseq-class object containing, at a minimum, abundance (otu_table-class) and phylogenetic (phylo) components. As a test, the accessors otu_table and phy_tree should return an object without error. |
| correction | (Optional). A function. The function must be able to take a non-Euclidean distance object, and return a new distance object that is Euclidean. If testing a distance object, try is.euclid.<br>In most real-life, real-data applications, the phylogenetic tree will not provide a Euclidean distance matrix, and so a correction will be needed. Two recommended correction methods are cailliez and lingoes. The default is cailliez, but not for any particularly special reason. If the patristic distance matrix turns out to be Euclidian, no correction will be performed, regardless of the value of the correction argument. |
| scannf | (Optional). Logical. Default is FALSE. This is passed directly to dpcoa, and causes a barplot of eigenvalues to be created if TRUE. This is not included in ... because the default for dpcoa is TRUE, although in many expected situations we would want to suppress creating the barplot. |
| ... | Additional arguments passed to dpcoa. |

### Details

In most real-life, real-data applications, the phylogenetic tree will not provide a Euclidean distance matrix, and so a correction will be performed, if needed. See correction argument.

### Value

A dpcoa-class object (see dpcoa).

### Author(s)

Julia Fukuyama <julia.fukuyama@gmail.com>. Adapted for phyloseq by Paul J. McMurdie.

### References

Pavoine, S., Dufour, A.B. and Chessel, D. (2004) From dissimilarities among species to dissimilarities among communities: a double principal coordinate analysis. Journal of Theoretical Biology, 228, 523-537.

### See Also

dpcoa

### Examples

```
# # # # # # # Esophagus
# data(esophagus)
# eso.dpcoa <- DPCoA(esophagus)
# plot_ordination(esophagus, eso.dpcoa, "samples")
# plot_ordination(esophagus, eso.dpcoa, "species")
# plot_ordination(esophagus, eso.dpcoa, "biplot")
# #
# #
# # # # # # # GlobalPatterns
# data(GlobalPatterns)
# # subset GP to top-150 taxa (to save computation time in example)
# keepTaxa <- names(sort(taxa_sums(GlobalPatterns), TRUE)[1:150])
# GP       <- prune_taxa(keepTaxa, GlobalPatterns)
# # Perform DPCoA
# GP.dpcoa <- DPCoA(GP)
# plot_ordination(GP, GP.dpcoa, color="SampleType")
```

---

## estimate_richness *Summarize richness estimates*

---

### Description

Performs a number of standard richness estimates, and returns the results as a data.frame. Can operate on the cumulative population of all samples in the dataset, or by repeating the richness estimates for each sample individually. NOTE: You must use untrimmed datasets for meaningful results, as these estimates (and even the "observed" richness) are highly dependent on the number of singletons. You can always trim the data later on if needed, just not before using this function.

### Usage

```
estimate_richness(physeq, split=TRUE)
```

### Arguments

| | |
|---|---|
| physeq | (Required). phyloseq-class, or alternatively, an otu_table-class. The data about which you want to estimate the richness. |
| split | (Optional). Logical. Should a separate set of richness estimates be performed for each sample? Or alternatively, pool all samples and estimate richness of the entire set. |

## Value

A data.frame of the richness estimates, and their standard error.

## See Also

Check out the custom plotting function, plot_richness_estimates, for easily showing the results of different estimates, with method-specific error-bars. Also check out the internal functions borrowed from the vegan package: estimateR, diversity

## Examples

```
data(GlobalPatterns)
( S.GP <- estimate_richness(GlobalPatterns) )
# # Make the plots
# plot_richness_estimates(GlobalPatterns, "SampleType")
# plot_richness_estimates(GlobalPatterns, "SampleType", "SampleType")
# For more plotting examples, see plot_richness_estimates()
```

---

## export_env_file *Export environment (ENV) file for UniFrac Server.*

---

## Description

Creates the environment table that is needed for the original UniFrac algorithm. Useful for cross-checking, or if want to use UniFrac server. Optionally the ENV-formatted table can be returned to the R workspace, and the tree component can be exported as Nexus format (Recommended).

## Usage

```
export_env_file(physeq, file = "", writeTree = TRUE,
  return = FALSE)
```

## Arguments

| | |
|---|---|
| physeq | (Required). Experiment-level (phyloseq-class) object. Ideally this also contains the phylogenetic tree, which is also exported by default. |
| file | (Optional). The file path for export. If not-provided, the expectation is that you will want to set return to TRUE, and manipulate the ENV table on your own. Default is "", skipping the ENV file from being written to a file. |
| writeTree | (Optional). Write the phylogenetic tree as well as the the ENV table. Default is TRUE. |
| return | (Optional). Should the ENV table be returned to the R workspace? Default is FALSE. |

## Examples

```
# # Load example data
# data(esophagus)
# export_env_file(esophagus, "~/Desktop/esophagus.txt")
```

## export_mothur_dist *Export a distance object as* .names *and* .dist *files for mothur*

### Description

The purpose of this function is to allow a user to easily export a distance object as a pair of files that can be immediately imported by mothur for OTU clustering and related analysis. A distance object can be created in R in a number of ways, including via cataloguing the cophentic distances of a tree object.

### Usage

```
export_mothur_dist(x, out=NULL,
makeTrivialNamesFile=NULL)
```

### Arguments

| | |
|---|---|
| x | (Required). A "dist" object, or a symmetric matrix. |
| out | (Optional). The desired output filename for the .dist file, OR left NULL, the default, in which case the mothur-formated distance table is returned to R standard out. |
| makeTrivialNamesFile | (Optional). Default NULL. The desired name of the .names file. If left NULL, the file name will be a modified version of the out argument. |

### Value

A character vector of the different cutoff values contained in the file. For a given set of arguments to the cluster() command from within *mothur*, a number of OTU-clustering results are returned in the same list file. The exact cutoff values used by *mothur* can vary depending on the input data. This simple function returns the cutoffs that were actually included in the *mothur* output. This an important extra step prior to importing the OTUs with the import_mothur_otulist() function.

### Examples

```
#
### data(GlobalPatterns)
### myDistObject <- as.dist(cophenetic(tre(GlobalPatterns)))
### export_mothur_dist(myDistObject, "myfilepathname.dist")
```

## filterfun_sample *A sample-wise filter function builder, analogous to* filterfun.

### Description

See the filterfun, from the Bioconductor repository, for a taxa-/gene-wise filter (and further examples).

### Usage

```
filterfun_sample(...)

filterfunSample(...)
```

### Arguments

| | |
|---|---|
| ... | A comma-separated list of functions. |

### Value

An enclosure (function) that itself will return a logical vector, according to the functions provided in the argument list, evaluated in order. The output of filterfun_sample is appropriate for the 'flist' argument to the genefilter_sample method.

### See Also

filterfun, genefilter_sample

### Examples

```
## Use simulated abundance matrix
# set.seed(711)
# testOTU <- otu_table(matrix(sample(1:50, 25, replace=TRUE), 5, 5), taxa_are_rows=FALSE)
# f1  <- filterfun_sample(topk(2))
# wh1 <- genefilter_sample(testOTU, f1, A=2)
# wh2 <- c(T, T, T, F, F)
# prune_taxa(wh1, testOTU)
# prune_taxa(wh2, testOTU)
```

---

## filter_taxa *Filter taxa based on across-sample OTU abundance criteria*

---

### Description

This function is directly analogous to the genefilter function for microarray filtering, but is used for filtering OTUs from phyloseq objects. It applies an arbitrary set of functions — as a function list, for instance, created by filterfun — as across-sample criteria, one OTU at a time. It takes as input a phyloseq object, and returns a logical vector indicating whether or not each OTU passed the criteria. Alternatively, if the "prune" option is set to FALSE, it returns the already-trimmed version of the phyloseq object.

### Usage

```
filter_taxa(physeq, flist, prune=FALSE)
```

## Arguments

| | |
|---|---|
| physeq | (Required). A phyloseq-class object that you want to trim/filter. |
| flist | (Required). A function or list of functions that take a vector of abundance values and return a logical. Some canned useful function types are included in the genefilter-package. |
| prune | (Optional). A logical. Default FALSE. If TRUE, then the function returns the pruned phyloseq-class object, rather than the logical vector of taxa that passed the filter. |

## Value

A logical vector equal to the number of taxa in physeq. This can be provided directly to prune_taxa as first argument. Alternatively, if prune==TRUE, the pruned phyloseq-class object is returned instead.

## See Also

filterfun, genefilter_sample, filterfun_sample

## Examples

```
data("enterotype")
require("genefilter")
flist    <- filterfun(kOverA(5, 2e-05))
ent.logi <- filter_taxa(enterotype, flist)
ent.trim <- filter_taxa(enterotype, flist, TRUE)
identical(ent.trim, prune_taxa(ent.logi, enterotype))
identical(sum(ent.logi), ntaxa(ent.trim))
filter_taxa(enterotype, flist, TRUE)
```

---

## genefilter_sample *Filter OTUs with arbitrary function, sample-wise.*

---

## Description

A general OTU trimming function for selecting OTUs that satisfy some criteria within the distribution of each sample, and then also an additional criteria for number of samples that must pass. This is a genefilter-like function that only considers sample-wise criteria. The number of acceptable samples is used as the final criteria (set by the argument A) to determine whether or not the taxa should be retained (TRUE) or not (FALSE). Just like with genefilter, a logical having length equal to nrow()/ntaxa is returned, indicating which should be kept. This output can be provided directly to OTU trimming function, prune_taxa. By contrast, genefilter, of the genefilter package in Bioconductor, works only on the rows of a matrix. Note that, because otu_table-class inherits directly from the matrix-class, an unmodified otu_table can be provided to genefilter, but be mindful of the orientation of the otu_table (use taxa_are_rows), and transpose (t) if needed.

## Usage

```
genefilter_sample(X, flist, A=1)

genefilterSample(X, flist, A = 1)
```

## Arguments

| | |
|---|---|
| X | The object that needs trimming. Can be matrix, otu_table, or higher- order phyloseq classes that contain an otu_table. |
| flist | An enclosure object, typically created with filterfun_sample |
| A | An integer. The number of samples in which a taxa / OTUs passed the filter for it to be labeled TRUE in the output logical vector. |

## Value

A logical vector with names equal to taxa_names (or rownames, if matrix).

## See Also

genefilter, filterfun_sample, t, prune_taxa

## Examples

```
#
## testOTU <- otu_table(matrix(sample(1:50, 25, replace=TRUE), 5, 5), taxa_are_rows=FALSE)
## f1  <- filterfun_sample(topk(2))
## wh1 <- genefilter_sample(testOTU, f1, A=2)
## wh2 <- c(T, T, T, F, F)
## prune_taxa(wh1, testOTU)
## prune_taxa(wh2, testOTU)
##
## tax_table1 <- tax_table(matrix("abc", 5, 5))
## prune_taxa(wh1, tax_table1)
## prune_taxa(wh2, tax_table1)
```

---

getslots.phyloseq *Return the non-empty slot names of a phyloseq object.*

---

## Description

Like getSlots, but returns the class name if argument is component data object.

## Usage

```
getslots.phyloseq(physeq)
```

## Arguments

| | |
|---|---|
| physeq | A phyloseq-class object. If physeq is a component data class, then just returns the class of physeq. |

## Value

identical to getSlots. A named character vector of the slot classes of a particular S4 class, where each element is named by the slot name it represents. If physeq is a component data object, then a vector of length (1) is returned, named according to its slot name in the phyloseq-class.

### See Also

merge_phyloseq

### Examples

```
#
data(GlobalPatterns)
getslots.phyloseq(GlobalPatterns)
data(esophagus)
getslots.phyloseq(esophagus)
```

---

## get_sample — *Returns all abundance values for species* i.

---

### Description

This is a simple accessor function for investigating a single species-of-interest.

### Usage

```
get_sample(physeq, i)

getSamples(physeq, i)
```

### Arguments

| | |
|---|---|
| physeq | (Required). otu_table-class, or phyloseq-class. |
| i | (Required). A single taxa/species/OTU ID for which you want to know the abundance in each sample. |

### Value

An integer vector of the abundance values for each sample in physeq for species i

### See Also

get_taxa taxa_names sample_names

### Examples

```
data(esophagus)
taxa_names(esophagus)
get_sample(esophagus, "59_5_19")
```

## get_taxa — *Returns all abundance values of sample* i.

### Description

This is a simple accessor function for investigating a single sample-of-interest.

### Usage

```
get_taxa(physeq, i)

getSpecies(physeq, i)
```

### Arguments

| | |
|---|---|
| physeq | (Required). otu_table-class, or phyloseq-class. |
| i | (Required). A single sample for which you want to know the abundance of each species. Can be integer for index value, or sample name. |

### Value

An integer vector of the abundance values for each species in physeq for sample i

### See Also

get_sample taxa_names sample_names

### Examples

```
data(esophagus)
sample_names(esophagus)
get_taxa(esophagus, "B")
```

## get_taxa_unique — *Get a unique vector of the observed taxa at a particular taxonomic rank*

### Description

This is a simple accessor function to make it more convenient to determine the different taxa present for a particular taxonomic rank in a given phyloseq-class object.

### Usage

```
get_taxa_unique(physeq,
taxonomic.rank=rank_names(physeq)[1], errorIfNULL=TRUE)

getTaxa(physeq, taxonomic.rank = rank_names(physeq)[1],
errorIfNULL = TRUE)
```

## Arguments

| | |
|---|---|
| physeq | (Required). taxonomyTable-class, or phyloseq-class. |
| taxonomic.rank | (Optional). Character. The taxonomic rank to use. Must select from the set indicated by get_taxa_unique. Default is to take the first column of the taxonomyTable component. |
| errorIfNULL | (Optional). Logical. Should the accessor stop with an error if the slot is empty (NULL)? Default TRUE. |

## Value

Character vector. Unique vector of the observed taxa at a particular taxonomic rank

## See Also

get_taxa taxa_names sample_names

## Examples

```
data(enterotype)
get_taxa_unique(enterotype)
data(GlobalPatterns)
get_taxa_unique(GlobalPatterns, "Family")
```

---

get_variable *Get the values for a particular variable in sample_data*

---

## Description

This is a simple accessor function for streamlining access to values/vectors/factors/etc contained in the sample_data.

## Usage

```
get_variable(physeq, varName)

getVariable(physeq, varName)
```

## Arguments

| | |
|---|---|
| physeq | (Required). sample_data-class, or phyloseq-class. |
| varName | (Required). Character string of the variable name in sample_data. Use sample_variables(physeq) for available variables in your object. |

## Value

Data. The clas of the data depends on what the contents of sample_data.

## See Also

get_taxa taxa_names sample_names

sample_variables

### Examples

```
# Load the GlobalPatterns dataset into the workspace environment
data(GlobalPatterns)
# Look at the different values for SampleType
get_variable(GlobalPatterns, "SampleType")
```

---

## import    *Universal import method (wrapper) for phyloseq-package*

---

### Description

A user must still understand the additional arguments required for each type of import data. Those arguments are described in detail at the tool-specific import_* links below. Each clustering tool / package / pipeline has its own idiosyncratic set of file names / types, and it remains the responsibility of the user to understand which file-path should be provided to each argument for the particular importing submethod. This method merely provides a central documentation and method-name, and the arguments are passed along as-is.

### Usage

```
import(pipelineName, ...)
```

### Arguments

| | |
|---|---|
| pipelineName | (Required). Character string. The name of the analysis tool / pipeline / package that created the OTU-cluster data or other data that you now want to import. Current options are c("mothur", "pyrotagger", "QIIME", "RDP"), and only the first letter is necessary. |
| ... | (Required). Additional arguments providing file paths, and possible other paramaters to the desired tool-specific import function. |

### Value

In most cases a phyloseq-class will be returned, though the included component data will vary by pipeline/tool, and also by the types of data files provided. The expected behavior is to return the most-comprehensive object possible, given the provided arguments and pipeline/tool.

### References

mothur: http://www.mothur.org/wiki/Main_Page

PyroTagger: http://pyrotagger.jgi-psf.org/

QIIME: http://qiime.org/

BIOM: http://www.biom-format.org/

RDP pipeline: http://pyro.cme.msu.edu/index.jsp

## See Also

For mothur, see: import_mothur

Separate tools for mothur are also: show_mothur_list_cutoffs import_mothur_dist export_mothur_dist

For PyroTagger, see: import_pyrotagger_tab

For QIIME, see: import_qiime

For BIOM format, see: import_biom

For RDP pipeline, see: import_RDP_cluster

import_RDP_otu

## Examples

```
## import("QIIME", otufilename=myOtuTaxFilePath, mapfilename=myMapFilePath)
```

---

import_biom *Import phyloseq data from BIOM file*

---

## Description

New versions of QIIME produce a more-comprehensive and formally-defined JSON file format. From the QIIME website:

## Usage

```
import_biom(BIOMfilename, taxaPrefix=NULL,
parallel=FALSE, version=0.9)
```

## Arguments

| | |
|---|---|
| BIOMfilename | (Required). A character string indicating the file location of the BIOM formatted file. This is a JSON formatted file, specific to biological datasets, as described in http://www.qiime.org/svn_documentation/documentation/biom_format.html |
| taxaPrefix | (Optional). Character string. What category of prefix precedes the taxonomic label at each taxonomic rank. Currently only "greengenes" is a supported option, and implies that the first letter indicates the taxonomic rank, followed by two underscores and then the actual taxonomic assignment at that rank. The default value is NULL, meaning that no prefix or rank identifier will be interpreted. |
| parallel | (Optional). Logical. Wrapper option for .parallel parameter in plyr-package functions. If TRUE, apply parsing functions in parallel, using parallel backend provided by foreach and its supporting backend packages. One caveat, plyr-parallelization currently works most-cleanly with multicore-like backends (Mac OS X, Unix?), and may throw warnings for SNOW-like backends. See the example below for code invoking multicore-style backend within the doParallel package.<br>Finally, for many datasets a parallel import should not be necessary because a serial import will be just as fast and the import is often only performed one time; after which the data should be saved as an RData file using the save function. |

| | |
|---|---|
| version | (Optional). Numeric. The expected version number of the file. As the BIOM format evolves, version-specific importers will be available by adjusting the version value. Default is 0.9. Not implemented. Has no effect (yet). |

## Details

"The biom file format (canonically pronounced 'biome') is designed to be a general-use format for representing counts of observations in one or more biological samples."

http://www.qiime.org/svn_documentation/documentation/biom_format.html

## Value

A phyloseq-class object.

## References

http://www.qiime.org/svn_documentation/documentation/biom_format.html

## See Also

import, import_qiime

## Examples

```
# # # import with default parameters, specify a file
# import_BIOM(myBIOMfile)
# # # Example code for importing large file with parallel backend
# library("doParallel")
# registerDoParallel(cores=6)
# import_biom("my/file/path/file.biom", taxaPrefix="greengenes", parallel=TRUE)
```

---

import_env_file *Read a UniFrac-formatted ENV file.*

---

## Description

Convenience wrapper function to read the environment-file, as formatted for input to the UniFrac server (http://bmf2.colorado.edu/unifrac/). The official format of these files is that each row specifies (in order) the sequence name, source sample, and (optionally) the number of times the sequence was observed.

## Usage

```
import_env_file(envfilename, tree=NULL, sep="\t", ...)
```

## Arguments

| | |
|---|---|
| envfilename | (Required). A charater string of the ENV filename (relative or absolute) |
| tree | (Optional). phylo-class object to be paired with the output otu_table. |
| sep | A character string indicating the delimiter used in the file. The default is "\t". |
| ... | Additional parameters passed on to read.table. |

## Value

An otu_table-class, or phyloseq-class if a phylo-class argument is provided to tree.

## References

http://bmf2.colorado.edu/unifrac/

## See Also

import, tip_glom

## Examples

```
# import_env_file(myEnvFile, myTree)
```

---

import_mothur *General function for importing mothur files into phyloseq.*

---

## Description

General function for importing mothur files into phyloseq.

## Usage

```
import_mothur(mothur_list_file, mothur_group_file=NULL,
mothur_tree_file=NULL, cutoff=NULL)
```

## Arguments

| | |
|---|---|
| mothur_list_file | Required. The list file name / location produced by *mothur*. |
| mothur_group_file | Optional. The name/location of the group file produced by *mothur*'s make.group() function. It contains information about the sample source of individual sequences, necessary for creating a species/taxa abundance table (otu_table). See http://www.mothur.org/wiki/Make.group |
| mothur_tree_file | Optional. The tree file name produced by *mothur*. Probably a file that ends with the suffix ".tree". |
| cutoff | A character string indicating the cutoff value, (or "unique"), that matches one of the cutoff-values used to produce the OTU clustering results contained within the list-file created by *mothur* (and specified by the mothur_list_file argument). The default is to take the largest value among the cutoff values contained in the list file. If only one cutoff is included in the file, it is taken and this argument does not need to be specified. Note that the cluster() function within the *mothur* package will often produce a list file with multiple cutoff values, even if a specific cutoff is specified. It is suggested that you check which cutoff values are available in a given list file using the show_mothur_list_cutoffs function. |

## Value

The object class depends on the provided arguments. If the first three arguments are provided, then an otuTree object should be returned, containing both an OTU-only tree and its associated otu_table-class abundance table. If only a list and group file are provided, then an otu_table object is returned. Similarly, if only a list and tree file are provided, then only a tree is returned ("phylo" class).

## References

http://www.mothur.org/wiki/Main_Page

Schloss, P.D., et al., Introducing mothur: Open-source, platform-independent, community-supported software for describing and comparing microbial communities. Appl Environ Microbiol, 2009. 75(23):7537-41.

## Examples

```
# # The following example assumes you have downloaded the esophagus example
# # dataset from the mothur wiki:
# # "http://www.mothur.org/wiki/Esophageal_community_analysis"
# # "http://www.mothur.org/w/images/5/55/Esophagus.zip"
# # The path on your machine may (probably will) vary
# mothur_list_file  <- "~/Downloads/mothur/Esophagus/esophagus.an.list"
# mothur_group_file <- "~/Downloads/mothur/Esophagus/esophagus.good.groups"
# mothur_tree_file  <- "~/Downloads/mothur/Esophagus/esophagus.tree"
# # # Actual examples follow:
# show_mothur_list_cutoffs(mothur_list_file)
# test1 <- import_mothur(mothur_list_file, mothur_group_file, mothur_tree_file)
# test2 <- import_mothur(mothur_list_file, mothur_group_file, mothur_tree_file, cutoff="0.02")
# # Returns just a tree
# import_mothur(mothur_list_file, mothur_tree_file=mothur_tree_file)
# # Returns just an otu_table
# import_mothur(mothur_list_file, mothur_group_file=mothur_group_file)
# # Returns an error
# import_mothur(mothur_list_file)
# # Should return an "OMG, you must provide the list file" error
# import_mothur()
```

---

import_mothur_dist *Import mothur-formatted distance file*

---

## Description

The mothur application will produce a file containing the pairwise distances between all sequences in a dataset. This distance matrix can be the basis for OTU cluster designations. R also has many built-in or off-the-shelf tools for dealing with distance matrices.

## Usage

```
import_mothur_dist(mothur_dist_file)
```

### Arguments

| | |
|---|---|
| mothur_dist_file | Required. The distance file name / location produced by *mothur*. |

### Value

A distance matrix object describing all sequences in a dataset.

### See Also

import_mothur

### Examples

```
## Take a look at the dataset shown here as an example:
## "http://www.mothur.org/wiki/Esophageal_community_analysis"
## find the file ending with extension ".dist", download to your system
## The location of your file may vary
# mothur_dist_file <- "~/Downloads/mothur/Esophagus/esophagus.dist"
# myNewDistObject  <- import_mothur_dist(mothur_dist_file)
```

---

## import_pyrotagger_tab *Imports a tab-delimited version of the pyrotagger output file.*

---

### Description

PyroTagger is a web-server that takes raw, barcoded 16S rRNA amplicon sequences and returns an excel spreadsheet (".xls") with both abundance and taxonomy data. It also includes some confidence information related to the taxonomic assignment.

### Usage

```
import_pyrotagger_tab(pyrotagger_tab_file,
strict_taxonomy=FALSE, keep_potential_chimeras=FALSE)
```

### Arguments

| | |
|---|---|
| pyrotagger_tab_file | (Required). A character string. The name of the tab-delimited pyrotagger output table. |
| strict_taxonomy | (Optional). Logical. Default FALSE. Should the taxonomyTable component be limited to just taxonomic data? Default includes all fields from the pyrotagger file. |
| keep_potential_chimeras | (Optional). Logical. Default FALSE. The pyrotagger output also includes OTUs that are tagged by pyrotagger as likely chimeras. These putative chimeric OTUs can be retained if set to TRUE. The putative chimeras are excluded by default. |

## Details

PyroTagger is created and maintained by the Joint Genome Institute at "http://pyrotagger.jgi-psf.org/"

The typical output form PyroTagger is a spreadsheet format ".xls", which poses additional import challenges. However, virtually all spreadsheet applications support the ".xls" format, and can further export this file in a tab-delimited format. It is recommended that you convert the xls-file without any modification (as tempting as it might be once you have loaded it) into a tab-delimited text file. Deselect any options to encapsulate fields in quotes, as extra quotes around each cell's contents might cause problems during file processing. These quotes will also inflate the file-size, so leave them out as much as possible, while also resisting any temptation to modify the xls-file "by hand".

A highly-functional and free spreadsheet application can be obtained as part of the cross-platform OpenOffice suite. It works for the above required conversion. Go to "http://www.openoffice.org/".

It is regrettable that this importer does not take the xls-file directly as input. However, because of the moving-target nature of spreadsheet file formats, there is limited support for direct import of these formats into R. Rather than add to the dependency requirements of emphphyloseq and the relative support of these xls-support packages, it seems more efficient to choose an arbitrary delimited text format, and focus on the data structure in the PyroTagger output. This will be easier to support in the long-run.

## Value

An otuTax object containing both the otu_table and TaxonomyTable data components, parsed from the pyrotagger output.

## References

http://pyrotagger.jgi-psf.org/

## Examples

```
## New_otuTaxObject <- import_pyrotagger_tab(pyrotagger_tab_file)
```

---

import_qiime *Import function to read files created by the QIIME pipeline.*

---

## Description

QIIME produces several files that can be analyzed in the phyloseq-package, including especially an OTU file that typically contains both OTU-abundance and taxonomic identity information. The map-file is also an important input to QIIME that stores sample covariates, converted naturally to the sample_data-class component data type in the phyloseq-package. QIIME may also produce a phylogenetic tree with a tip for each OTU, which can also be imported by this function.

## Usage

```
import_qiime(otufilename=NULL, mapfilename=NULL,
treefilename=NULL, biotaxonomy=NULL, showProgress=TRUE,
chunk.size=1000L, ...)
```

## Arguments

| | |
|---|---|
| otufilename | (Optional). A character string indicating the file location of the OTU file. The combined OTU abundance and taxonomic identification file, tab-delimited, as produced by QIIME under default output settings. Default value is NULL. |
| mapfilename | (Optional). The QIIME map file required for processing pyrosequencing tags in QIIME as well as some of the post-clustering analysis. This is a required input file for running QIIME. Its strict formatting specification should be followed for correct parsing by this function. Default value is NULL. |
| treefilename | (Optional). A file representing a phylogenetic tree or a phylo object. Files can be NEXUS or Newick format. See read_tree for more details. If provided, the tree should have the same OTUs/tip-labels as the OTUs in the other files. Any taxa or samples missing in one of the files is removed from all. For the QIIME pipeline this tree is typically a tree of the representative 16S rRNA sequences from each OTU cluster, with the number of leaves/tips equal to the number of taxa/species/OTUs. Default value is NULL. Note that this argument can be a tree object (phylo-class) for cases where the tree has been — or needs to be — imported separately. |
| biotaxonomy | (Optional). A character vector indicating the name of each taxonomic level in the taxonomy-portion of the otu-file, which may not specify these levels explicitly. Default value is NULL. |
| showProgress | (Optional). A logical. Indicates whether import progress/status should be printed to the terminal. Default value is TRUE, meaning the progress will be shown. |
| chunk.size | (Optional). Positive integer. Default is 1000L. This is the number of lines to be read and processed at a time, passed directly to the import_qiime_otu_tax function. A lower value helps control memory-fault, but slows down the import. |
| ... | Additional arguments passed to read_tree |

## Details

See "http://www.qiime.org/" for details on using QIIME. While there are many complex dependencies, QIIME can be downloaded as a pre-installed linux virtual machine that runs “off the shelf”.

The different files useful for import to *phyloseq* are not collocated in a typical run of the QIIME pipeline. See the main *phyloseq* vignette for an example of where ot find the relevant files in the output directory.

## Value

A phyloseq-class object.

## References

http://qiime.org/

“QIIME allows analysis of high-throughput community sequencing data.” J Gregory Caporaso, Justin Kuczynski, Jesse Stombaugh, Kyle Bittinger, Frederic D Bushman, Elizabeth K Costello, Noah Fierer, Antonio Gonzalez Pena, Julia K Goodrich, Jeffrey I Gordon, Gavin A Huttley, Scott T Kelley, Dan Knights, Jeremy E Koenig, Ruth E Ley, Catherine A Lozupone, Daniel McDonald, Brian D Muegge, Meg Pirrung, Jens Reeder, Joel R Sevinsky, Peter J Turnbaugh, William A Walters, Jeremy Widmann, Tanya Yatsunenko, Jesse Zaneveld and Rob Knight; Nature Methods, 2010; doi:10.1038/nmeth.f.303

### See Also

phyloseq

merge_phyloseq

read_tree

### Examples

```
# import_qiime(myOtuTaxFilePath, myMapFilePath)
```

---

## import_qiime_otu_tax *Import a QIIME-formatted otu-tax file into a list of two matrices.*

---

### Description

QIIME produces several files that can be analyzed in the phyloseq-package, including especially an OTU file that typically contains both OTU-abundance and taxonomic identity information.

### Usage

```
import_qiime_otu_tax(file, biotaxonomy=NULL,
parallel=FALSE, chunk.size=1000, verbose=TRUE)
```

### Arguments

| | |
|---|---|
| file | (Required). The path to the qiime-formatted file you want to import into R. Can be compressed (e.g. .gz, etc.), though the details may be OS-specific. That is, Windows-beware. |
| biotaxonomy | (Optional). A character vector of the taxonomic ranks. Default is NULL, meaning no rank-names will be added as colnames to the $tax_table matrix. |
| parallel | (Optional). Logical. Should the parsing be performed in parallel?. Default is FALSE. Only a few steps are actually parallelized, and for most datasets it will actually be faster and more efficient to keep this set to FALSE. Also, to get any benefit at all, you will need to register a parallel “backend” through one of the backend packages supported by the foreach-package. |
| chunk.size | (Optional). Positive integer. Default is 1000L. This is the number of lines to be read and processed at a time. In many cases this will have a much larger effect on speed and efficiency than the setting for parallel. If you set this value too large, you risk a memory fault. If you set it too low, you risk a very slow parsing process. |
| verbose | (Optional). Logical. Default is TRUE. Should status updates of the parsing process be printed to screen? |

### Details

See "http://www.qiime.org/" for details on using QIIME. While there are many complex dependencies, QIIME can be downloaded as a pre-installed linux virtual machine that runs “off the shelf”.

This function uses chunking to perform both the reading and parsing in blocks of optional size, thus constrain the peak memory usage. feature should make this importer accessible to machines with modest memory, but with the caveat that the full numeric matrix must be a manageable size at the end, too. In principle, the final tables will be large, but much more efficiently represented than the character-stored numbers. If total memory for storing the numeric matrix becomes problematic, a switch to a sparse matrix representation of the abundance – which is typically well-suited to this data – might provide a solution.

### Value

A list of two matrices. `$otutab` contains the OTU Table as a numeric matrix, while `$tax_table` contains a character matrix of the taxonomy assignments.

### See Also

import, merge_phyloseq, phyloseq, import_qiime import_qiime_otu_tax import_env_file

### Examples

```
# # data_list <- import_qiime_otu_tax(file)
```

## import_qiime_sample_data

*Import just* `sample_data` *file from QIIME pipeline.*

### Description

QIIME produces several files that can be analyzed in the phyloseq-package, This includes the map-file, which is an important *input* to QIIME that can also indicate sample covariates. It is converted naturally to the sample_data component data type in phyloseq-package, based on the R data.frame.

### Usage

```
import_qiime_sample_data(mapfilename)

import_qiime_sampleData(mapfilename)
```

### Arguments

| | |
|---|---|
| mapfilename | (Required). A character string or connection. That is, any suitable file argument to the read.table function. The name of the QIIME map file required for processing pyrosequencing tags in QIIME as well as some of the post-clustering analysis. This is a required input file for running QIIME. Its strict formatting specification is expected by this function, do not attempt to modify it manually once it has worked properly in QIIME. |

### Details

See import_qiime for more information about QIIME. It is also the suggested function for importing QIIME-produced data files.

### Value

A `sample_data` object.

### See Also

import, merge_phyloseq, phyloseq, import_qiime import_qiime_otu_tax import_env_file

### Examples

```
mapfile <- system.file("extdata", "master_map.txt", package = "phyloseq")
import_qiime_sample_data(mapfile)
```

## import_RDP_cluster *Import RDP cluster file and return otu_table (abundance table).*

### Description

The RDP cluster pipeline (specifically, the output of the complete linkage clustering step) has no formal documentation for the ".clust" file or its apparent sequence naming convention.

### Usage

```
import_RDP_cluster(RDP_cluster_file)
```

### Arguments

| | |
|---|---|
| RDP_cluster_file | A character string. The name of the ".clust" file produced by the the complete linkage clustering step of the RDP pipeline. |

### Details

http://pyro.cme.msu.edu/index.jsp

The cluster file itself contains the names of all sequences contained in input alignment. If the upstream barcode and aligment processing steps are also done with the RDP pipeline, then the sequence names follow a predictable naming convention wherein each sequence is named by its sample and sequence ID, separated by a "_" as delimiter:

"sampleName_sequenceIDnumber"

This import function assumes that the sequence names in the cluster file follow this convention, and that the sample name does not contain any "_". It is unlikely to work if this is not the case. It is likely to work if you used the upstream steps in the RDP pipeline to process your raw (barcoded, untrimmed) fasta/fastq data.

This function first loops through the ".clust" file and collects all of the sample names that appear. It secondly loops through each OTU ("cluster"; each row of the cluster file) and sums the number of sequences (reads) from each sample. The resulting abundance table of OTU-by-sample is trivially coerced to an otu_table object, and returned.

### Value

An otu_table object parsed from the ".clust" file.

### References

http://pyro.cme.msu.edu/index.jsp

## import_RDP_otu *Import new RDP OTU-table format*

### Description

Recently updated tools on RDP Pyro site make it easier to import Pyrosequencing output into R. The modified tool "Cluster To R Formatter" can take a cluster file (generated from RDP Clustering tools) to create a community data matrix file for distance cutoff range you are interested in. The resulting output file is a tab-delimited file containing the number of sequences for each sample for each OTU. The OTU header naming convention is "OTU_" followed by the OTU number in the cluster file. It pads "0"s to make the OTU header easy to sort. The OTU numbers are not necessarily in order.

### Usage

```
import_RDP_otu(otufile)
```

### Arguments

| | |
|---|---|
| otufile | (Optional). A character string indicating the file location of the OTU file, produced/exported according to the instructions above. |

### Value

A otu_table-class object.

### See Also

An alternative "cluster" file importer for RDP results: import_RDP_cluster

The main RDP-pyrosequencing website http://pyro.cme.msu.edu/index.jsp

### Examples

```
otufile <- system.file("extdata", "rformat_dist_0.03.txt.gz", package="phyloseq")
### the gzipped file is automatically recognized, and read using R-connections
ex_otu  <- import_RDP_otu(otufile)
class(ex_otu)
ntaxa(ex_otu)
nsamples(ex_otu)
sample_sums(ex_otu)
head(t(ex_otu))
```

## make_network *Make microbiome network (igraph0)*

### Description

A specialized function for creating a network representation of microbiomes, sample-wise or taxa-wise, based on a user-defined ecological distance and (potentially arbitrary) threshold. The graph is ultimately represented using the igraph0-package.

### Usage

```
make_network(physeq, type="samples", distance="jaccard",
max.dist = 0.4, keep.isolates=FALSE, ...)
```

### Arguments

| | |
|---|---|
| physeq | (Required). Default NULL. A phyloseq-class object, or otu_table-class object, on which g is based. phyloseq-class recommended. |
| type | (Optional). Default "samples". Whether the network should be samples or taxa/OTUs. Supported arguments are "samples", "taxa", where "taxa" indicates using the OTUs/taxaindices, whether they actually represent species or some other taxonomic rank. NOTE: not all distance methods are supported if "taxa" selected for type. For example, the UniFrac distance and DPCoA cannot be calculated for taxa-wise distances, because they use a taxa-wise tree as part of their calculation between samples, and there is no transpose-equivalent for this tree. |
| distance | (Optional). Default "jaccard". Any supported argument to the method parameter of the distance function is supported here. Some distance methods, like "unifrac", may take a non-trivial amount of time to calculate, in which case you probably want to calculate the distance matrix separately, save, and then provide it as the argument to distance instead. See below for alternatives). Alternatively, if you have already calculated the sample-wise distance object, the resulting dist-class object can be provided as distance instead (see examples). A third alternative is to provide a function that takes a sample-by-taxa matrix (typical vegan orientation) and returns a sample-wise distance matrix. |
| max.dist | (Optional). Default 0.4. The maximum ecological distance (as defined by distance) allowed between two samples to still consider them "connected" by an edge in the graphical model. |
| keep.isolates | (Optional). Default FALSE. Logical. Whether to keep isolates (un-connected samples, not microbial isolates) in the graphical model that is returned. Default results in isolates being removed from the object. |
| ... | (Optional). Additional parameters passed on to distance. |

### Value

A igraph0-class object.

### See Also

plot_network

### Examples

```
# # Example plots with Enterotype Dataset
data(enterotype)
ig <- make_network(enterotype, max.dist=0.3)
plot_network(ig, enterotype, color="SeqTech", shape="Enterotype", line_weight=0.3, label=NULL)
#
# ig <- make_network(enterotype, max.dist=0.2)
# plot_network(ig, enterotype, color="SeqTech", shape="Enterotype", line_weight=0.3, label=NULL)
#
# # Three methods of choosing/providing distance/distance-method
# Provide method name available to distance
ig <- make_network(enterotype, max.dist=0.3, distance="jaccard")
# Provide distance object, already computed
jaccdist <- distance(enterotype, "jaccard")
ih <- make_network(enterotype, max.dist=0.3, distance=jaccdist)
# Provide "custom" function.
ii <- make_network(enterotype, max.dist=0.3, distance=function(x){vegdist(x, "jaccard")})
# The have equal results:
all.equal(ig, ih)
all.equal(ig, ii)
#
# Try out making a trivial "network" of the 3-sample esophagus data,
# with weighted-UniFrac as distance
data(esophagus)
ij <- make_network(esophagus, "samples", "unifrac", weighted=TRUE)
```

---

## merge_phyloseq *Merge arguments into one phyloseq object.*

---

### Description

Takes a comma-separated list of phyloseq objects as arguments, and returns the most-comprehensive single phyloseq object possible.

### Usage

```
merge_phyloseq(...)
```

### Arguments

| | |
|---|---|
| ... | a comma-separated list of phyloseq objects. |

### Details

Higher-order objects can be created if arguments are appropriate component data types of different classes, and this should mirror the behavior of the phyloseq method, which is the suggested method if the goal is simply to create a higher-order phyloseq object from different data types (1 of each class) describing the same experiment.

By contrast, this method is intended for situations in which one wants to combine multiple higher-order objects, or multiple core component data objects (e.g. more than one `otu_table`) that should be combined into one object.

Merges are performed by first separating higher-order objects into a list of their component objects; then, merging any component objects of the same class into one object according to the behavior desribed in merge_phyloseq_pair; and finally, building back up a merged-object according to the constructor behavior of the phyloseq method. If the arguments contain only a single component type – several otu_table objects, for example – then a single merged object of that component type is returned.

## Value

Merges are performed by first separating higher-order objects into a list of their component objects; then, merging any component objects of the same class into one object according to the behavior desribed in merge_phyloseq_pair; and finally, re-building a merged-object according to the constructor behavior of the phyloseq method. If the arguments contain only a single component type – several otu_table objects, for example – then a single merged object of the relevant component type is returned.

Merges between 2 or more tree objects are ultimately done using consensus from the ape package. This has the potential to limit somewhat the final data object, because trees don't merge with other trees in the same granular manner as data tables, and ultimately the species/taxa in higher-order phyloseq objects will be clipped to what is contained in the tree. If this an issue, the tree component should be ommitted from the argument list.

## Examples

```
#
## # Make a random complex object
## OTU1 <- otu_table(matrix(sample(0:5,250,TRUE),25,10), taxa_are_rows=TRUE)
## tax1 <- tax_table(matrix("abc", 30, 8))
## map1 <- data.frame( matrix(sample(0:3,250,TRUE),25,10),
##   matrix(sample(c("a","b","c"),150,TRUE), 25, 6) )
## map1 <- sample_data(map1)
## exam1 <- phyloseq(OTU1, map1, tax1)
## x <- exam1
## x <- phyloseq(exam1)
## y <- tax_table(exam1)
## merge_phyloseq(x, y)
## merge_phyloseq(y, y, y, y)
```

---

## merge_phyloseq_pair    *Merge pair of phyloseq component data objects of the same class.*

---

## Description

Internal S4 methods to combine pairs of objects of classes specified in the phyloseq package. These objects must be component data of the same type (class). This is mainly an internal method, provided to illustrate how merging is performed by the more general merge_phyloseq function.

## Usage

```
merge_phyloseq_pair(x, y)
```

## Arguments

| | |
|---|---|
| x | A character vector of the species in object x that you want to keep – OR alternatively – a logical vector where the kept species are TRUE, and length is equal to the number of species in object x. If species is a named logical, the species retained is based on those names. Make sure they are compatible with the taxa_names of the object you are modifying (x). |
| y | Any phyloseq object. |

## Details

The merge_phyloseq function is recommended in general.

Special note: non-identical trees are merged using consensus.

## Value

A single component data object that matches x and y arguments. The returned object will contain the union of the species and/or samples of each. If there is redundant information between a pair of arguments of the same class, the values in x are used by default. Abundance values are summed for otu_table objects for those elements that describe the same species and sample in x and y.

## See Also

merge_phyloseq merge_taxa

## Examples

```
#
## # merge two simulated otu_table objects.
## x  <- otu_table(matrix(sample(0:5,200,TRUE),20,10), taxa_are_rows=TRUE)
## y  <- otu_table(matrix(sample(0:5,300,TRUE),30,10), taxa_are_rows=FALSE)
## xy <- merge_phyloseq_pair(x, y)
## yx <- merge_phyloseq_pair(y, x)
## # merge two simulated tax_table objects
## x <- tax_table(matrix("abc", 20, 6))
## y <- tax_table(matrix("def", 30, 8))
## xy <- merge_phyloseq_pair(x, y)
## # merge two simulated sample_data objects
## x <- data.frame( matrix(sample(0:3,250,TRUE),25,10),
##   matrix(sample(c("a","b","c"),150,TRUE),25,6) )
## x <- sample_data(x)
## y <- data.frame( matrix(sample(4:6,200,TRUE),20,10),
##   matrix(sample(c("d","e","f"),120,TRUE),20,8) )
## y <- sample_data(y)
## merge_phyloseq_pair(x, y)
## data.frame(merge_phyloseq_pair(x, y))
## data.frame(merge_phyloseq_pair(y, x))
```

merge_samples *Merge samples based on a sample variable or factor.*

## Description

The purpose of this method is to merge/agglomerate the sample indices of a phyloseq object according to a categorical variable contained in a sample_data or a provided factor.

## Usage

```
merge_samples(x, group, fun=mean)
```

## Arguments

| | |
|---|---|
| x | (Required). An instance of a phyloseq class that has sample indices. This includes sample_data-class, otu_table-class, and phyloseq-class. |
| group | (Required). Either the a single character string matching a variable name in the corresponding sample_data of x, or a factor with the same length as the number of samples in x. |
| fun | (Optional). The function that will be used to merge the values that correspond to the same group for each variable. It must take a numeric vector as first argument and return a single value. Default is mean. Note that this is (currently) ignored for the otu_table, where the equivalent function is sum, but evaluated via rowsum for efficiency. |

## Details

NOTE: (phylo) trees and taxonomyTable-class are not modified by this function, but returned in the output object as-is.

## Value

A phyloseq object that has had its sample indices merged according to the factor indicated by the group argument. The output class matches x.

## See Also

merge_taxa, codemerge_phyloseq

## Examples

```
#
# data(GlobalPatterns)
# t1 <- merge_samples(sample_data(GlobalPatterns), "SampleType")
# t4 <- merge_samples(GlobalPatterns, "SampleType")
# identical(t1, sample_data(t4))
```

merge_taxa *Merge a subset of the species in* x *into one species/taxa/OTU.*

## Description

Takes as input an object that describes species/taxa (e.g. phyloseq-class, otu_table-class, phylo-class, taxonomyTable-class), as well as a vector of species that should be merged. It is intended to be able to operate at a low-level such that related methods, such as tip_glom and tax_glom can both reliably call merge_taxa for their respective purposes.

## Usage

```
merge_taxa(x, eqspecies, archetype=1)

merge_species(x, eqspecies, archetype = 1)
```

## Arguments

| | |
|---|---|
| x | (Required). An object that describes species (taxa). This includes phyloseq-class, otu_table-class, taxonomyTable-class, phylo. |
| eqspecies | (Required). The species names, or indices, that should be merged together. If length(eqspecies) < 2, then the object x will be returned safely unchanged. |
| archetype | The index of eqspecies indicating the species that should be kept (default is 1) to represent the summed/merged group of species/taxa/OTUs. If archetype is not an index or index-name in eqspecies, the first will be used, and the value in archetype will be used as the index-name for the new species. |

## Value

The object, x, in its original class, but with the specified species merged into one entry in all relevant components.

## See Also

tip_glom, tax_glom, merge_phyloseq, merge_samples

## Examples

```
#
# # data(phylocom)
# # tree <- phylocom$phylo
# # otu  <- otu_table(phylocom$sample, taxa_are_rows=FALSE)
# # otutree0 <- phyloseq(otu, tree)
# # plot(otutree0)
# # otutree1 <- merge_taxa(otutree0, tree$tip.label[1:8], 2)
# # plot(otutree1)
```

## mt — *Multiple testing of taxa abundance acccording to sample categories/classes*

### Description

Multiple testing of taxa abundance acccording to sample categories/classes

### Usage

```
mt(physeq, classlabel, minPmaxT="minP", ...)
```

### Arguments

| | |
|---|---|
| physeq | (Required). otu_table-class or phyloseq-class. In this multiple testing framework, different taxa correspond to variables (hypotheses), and samples to observations. |
| classlabel | (Required). A single character index of the sample-variable in the sample_data of physeq that will be used for multiple testing. Alternatively, classlabel can be a custom integer (or numeric coercable to an integer), character, or factor with length equal to nsamples(physeq).<br>NOTE: the default test applied to each taxa is a two-sample two-sided t.test, WHICH WILL FAIL with an error if you provide a data variable (or custom vector) that contains MORE THAN TWO classes. One alternative to consider is an F-test, by specifying test="f" as an additional argument. See the first example below, and/or further documentation of mt.maxT or mt.minP for other options and formal details. |
| minPmaxT | (Optional). Character string. "mt.minP" or "mt.maxT". Default is to use mt.minP. |
| ... | (Optional). Additional arguments, forwarded to mt.maxT or mt.minP |

### Value

A dataframe with components specified in the documentation for mt.maxT or mt.minP, respectively.

### See Also

mt.maxT, mt.minP

### Examples

```
#
## # Simple example, testing genera that sig correlate with Enterotypes
## data(enterotype)
## # Filter samples that don't have Enterotype
## x <- subset_samples(enterotype, !is.na(Enterotype))
## # (the taxa are at the genera level in this dataset)
## mt(x, "Enterotype", test="f")
## # Not surprisingly, Prevotella and Bacteroides top the list.
## # Different test, multiple-adjusted t-test, whether samples are ent-2 or not.
## mt(x, get_variable(x, "Enterotype")==2)
```

## nsamples *Get the number of samples.*

### Description

Get the number of samples.

### Usage

```
nsamples(physeq)
```

### Arguments

| | |
|---|---|
| physeq | A phyloseq-class, sample_data, or otu_table-class. |

### Value

An integer indicating the total number of samples.

### See Also

taxa_names, sample_names, ntaxa

### Examples

```
#
# # From "picante" package
# data("phylocom")
# tree <- phylocom$phylo
# OTU1 <- otu_table(phylocom$sample, taxa_are_rows=FALSE)
# nsamples(OTU1)
# physeq1 <- phyloseq(OTU1, tree)
# nsamples(physeq1)
```

## ntaxa *Get the number of taxa/species.*

### Description

Get the number of taxa/species.

### Usage

```
ntaxa(physeq)

nspecies(physeq)
```

### Arguments

| | |
|---|---|
| physeq | phyloseq-class, otu_table-class, taxonomyTable-class, or phylo |

## Value

An integer indicating the number of taxa / species.

## See Also

taxa_names

## Examples

```
#
# # From "picante" package
# data("phylocom")
# tree <- phylocom$phylo
# ntaxa(tree)
```

---

ordinate *Perform an ordination on phyloseq data*

---

## Description

This function wraps several commonly-used ordination methods. The type of ordination depends upon the argument to method. Try ordinate("help") or ordinate("list") for the currently supported method options.

## Usage

```
ordinate(physeq, method="DCA", distance="unifrac", ...)
```

## Arguments

| | |
|---|---|
| physeq | (Required). Phylogenetic sequencing data (phyloseq-class). The data on which you want to perform the the ordination. In general, these methods will be based in some fashion on the abundance table ultimately stored as a contingency matrix (otu_table-class). If you're able to import data into phyloseq-class format, than you don't need to worry, as an otu_table is a required component of this class. In addition, some ordination methods require additional data, like a constraining variable or phylogenetic tree. If that is the case, the relevant data should be included in physeq prior to running. Integrating the data in this way also results in these different data components being checked for validity and completeness by the method. |
| method | (Optional). A character string. Default is "DCA". Currently supported method options are: c("DCA", "CCA", "RDA", "DPCoA", "NMDS", "MD **DCA** Performs detrended correspondence analysis using decorana **CCA** Performs correspondence analysis, or optionally, constrained correspondence analysis (a.k.a. canonical correspondence analysis), via cca **RDA** Performs redundancy analysis, or optionally principal components analysis, via rda **DPCoA** Performs Double Principle Coordinate Analysis using a (corrected, if necessary) phylogenetic/patristic distance between species. The calculation is performed by DPCoA(), which ultimately uses dpcoa after making the appropriate accessions/corrections of the data. |

**NMDS** Performs Non-metric MultiDimenstional Scaling of a sample-wise ecological distance matrix onto a user-specified number of axes, k. By default, k=2, but this can be modified as a supplementary argument. This method is ultimately carried out by metaMDS after the appropriate accessions and distance calculations. Because metaMDS includes its own distance calculation wrappers to vegdist, and these provide additional functionality in the form of species scores, ordinate will pass-on the distance argument to metaMDS if it is among the supported vegdist methods. However, all distance methods supported by distance are supported here, including "unifrac" (the default) and "DPCoA".

**MDS/PCoA** Performs principal coordinate analysis (also called principle coordinate decomposition, multidimensional scaling (MDS), or classical scaling) of a distance matrix (Gower 1966), including two correction methods for negative eigenvalues. See pcoa for further details.

| | |
|---|---|
| distance | (Optional). A character string matching a distance method; or, alternatively, a pre-computed dist-class object. This argument is only utilized if a distance matrix is required by the ordination method specified by the method argument (above). Any supported distance methods are supported arguments to distance here. Try distance("list") for a explicitly supported distance method abbreviations. User-specified custom distance equations should also work, e.g. "(A+B-2*J)/(A+B)". See distance for more details, examples. |
| ... | (Optional). Additional arguments to supporting functions. For example, the additional argument weighted=TRUE would be passed on to UniFrac if "unifrac" were chosen as the distance option and "MDS" as the ordination method option. Alternatively, if "DCA" were chosen as the ordination method option, additional arguments would be passed on to the relevant ordination function, decorana, for example. |

## Value

An ordination object. The specific class of the returned object depends upon the ordination method, as well as the function/package that is called internally to perform it. As a general rule, any of the ordination classes returned by this function will be recognized by downstream tools in the phyloseq package, for example the ordination plotting function, plot_ordination.

## See Also

Related component ordination functions described within phyloseq:

DPCoA

Described/provided by other packages:

cca/rda, decorana, metaMDS, pcoa

NMDS and MDS/PCoA both operate on distance matrices, typically based on some pairwise comparison of the microbiomes in an experiment/project. There are a number of common methods to use to calculate these pairwise distances, and the most convenient function (from a phyloseq point of view) for calculating these distance matrices is the

distance

function. It can be thought of as a distance / dissimilarity-index companion function for ordinate, and indeed the distance options provided to ordinate simply passed on to distance.

A good quick summary of ordination is provided in the introductory vignette for vegan:

vegan introductory vignette

The following R task views are also useful for understanding the available tools in R:

Analysis of Ecological and Environmental Data

Multivariate Statistics

## Examples

```
# # Take a subset of the GP dataset for quicker computation of examples
# data(GlobalPatterns)
# # Keep top 200 species
# topsp <- names(sort(taxa_sums(GlobalPatterns), TRUE)[1:200])
# GP     <- prune_taxa(topsp, GlobalPatterns)
# # Subset further to top 5 phyla
# top5ph <- sort(tapply(taxa_sums(GP), tax_table(GP)[, "Phylum"], sum), decreasing=TRUE)[1:5]
# GP      <- subset_taxa(GP, Phylum %in% names(top5ph))
# #
# # Examples performing ordination with NMDS. Default distance is unweighted UniFrac
# GP.NMDS.UF.ord    <- ordinate(GP, "NMDS")
# GP.NMDS.wUF.ord  <- ordinate(GP, "NMDS", "unifrac", weighted=TRUE)
# GP.NMDS.Bray.ord <- ordinate(GP, "NMDS", "bray")
# #
# # # An example plot with default, or manually-defined shapes
# (p <- plot_ordination(GP, GP.NMDS.Bray.ord, "biplot", color="SampleType", shape="Phylum"))
# # define manual shape scale:
# man.shapes <- 21:25
# names(man.shapes) <- c(get_taxa_unique(GP, "Phylum"))
# man.shapes <- c(samples=19, man.shapes)
# p + scale_shape_manual(value=man.shapes)
# #
# # An example of constrained ordination
# GP.cca <- ordinate(GP~SampleType, "CCA")
# #
# # Run-through "quick" plot examples of the other ordination options currently supported
# # Only showing "samples" in these examples, but "species" options supported for some methods
# plot_ordination(GP, ordinate(GP, "DCA"), "samples", color="SampleType")
# plot_ordination(GP, ordinate(GP, "CCA"), "samples", color="SampleType")
# plot_ordination(GP, ordinate(GP~SampleType, "CCA"), "samples", color="SampleType")
# plot_ordination(GP, ordinate(GP, "RDA"), "samples", color="SampleType")
# plot_ordination(GP, ordinate(GP~SampleType, "RDA"), "samples", color="SampleType")
# plot_ordination(GP, ordinate(GP, "DPCoA"), "samples", color="SampleType")
# plot_ordination(GP, ordinate(GP, "MDS"), "samples", color="SampleType")
# plot_ordination(GP, ordinate(GP, "PCoA"), "samples", color="SampleType")
# plot_ordination(GP, ordinate(GP, "NMDS"), "samples", color="SampleType")
# plot_ordination(GP, ordinate(GP, "NMDS", "w"), "samples", color="SampleType")
```

---

## otu_table *Build or access the otu_table.*

---

## Description

This is the suggested method for both constructing and accessing Operational Taxonomic Unit (OTU) abundance (otu_table-class) objects. When the first argument is a matrix, otu_table() will attempt to create and return an otu_table-class object, which further depends on whether or not taxa_are_rows is provided as an additional argument. Alternatively, if the first argument is an experiment-level (phyloseq-class) object, then the corresponding otu_table is returned.

### Usage

```
otu_table(object, taxa_are_rows, errorIfNULL=TRUE)

otuTable(object, taxa_are_rows, errorIfNULL = TRUE)
```

### Arguments

| | |
|---|---|
| object | (Required). An integer matrix, otu_table-class, or phyloseq-class. |
| taxa_are_rows | (Conditionally optional). Logical; of length 1. Ignored unless object is a matrix, in which case it is is required. |
| errorIfNULL | (Optional). Logical. Should the accessor stop with an error if the slot is empty (NULL)? Default TRUE. Ignored if object argument is a matrix (constructor invoked instead). |

### Value

An otu_table-class object.

### See Also

phy_tree, sample_data, tax_table phyloseq, merge_phyloseq

### Examples

```
#
# data(GlobalPatterns)
# otu_table(GlobalPatterns)
```

## otu_table-class *The S4 class for storing taxa-abundance information.*

### Description

Because orientation of these tables can vary by method, the orientation is defined explicitly in the taxa_are_rows slot (a logical). The otu_table class inherits the matrix class to store abundance values. Various standard subset and assignment nomenclature has been extended to apply to the otu_table class, including square-bracket, t, etc.

### Details

**taxa_are_rows** A single logical specifying the orientation of the abundance table.

**.Data** This slot is inherited from the matrix class.

## otu_table<- *Assign a new OTU Table to* x

### Description

Assign a new OTU Table to x

### Usage

```
otu_table(x) <- value
```

### Arguments

| | |
|---|---|
| x | (Required). phyloseq-class |
| value | (Required). otu_table-class or phyloseq-class. |

### Examples

```
# data(GlobalPatterns)
# # An example of pruning to just the first 100 taxa in GlobalPatterns.
# ex2a <- prune_species(taxa_names(GlobalPatterns)[1:100], GlobalPatterns)
# # The following 3 lines produces an ex2b that is equal to ex2a
# ex2b <- GlobalPatterns
# OTU <- otu_table(GlobalPatterns)[1:100, ]
# otu_table(ex2b) <- OTU
# identical(ex2a, ex2b)
# print(ex2b)
# # Relace otu_table by implying the component in context.
# ex2c <- GlobalPatterns
# otu_table(ex2c) <- ex2b
# identical(ex2a, ex2c)
```

## phylo-class *An S4 copy of the main phylogenetic tree class from the ape package.*

### Description

See the ape package for details about this type of representation of a phylogenetic tree. It is used throught ape.

### See Also

phylo, setOldClass

| phyloseq | *Build phyloseq-class objects from their components.* |
|---|---|

## Description

phyloseq() is a constructor method, This is the main method suggested for constructing an experiment-level (phyloseq-class) object from its component data (component data classes: otu_table-class, sample_data-class, taxonomyTable-class, phylo-class).

## Usage

```
phyloseq(...)
```

## Arguments

| | |
|---|---|
| ... | One or more component objects among the set of classes defined by the phyloseq package, as well as phylo-class (defined by the ape-package). Each argument should be a different class. For combining multiple components of the same class, or multiple phyloseq-class objects, use the merge_phyloseq function. Unlike in earlier versions, the arguments to phyloseq do not need to be named, and the order of the arguments does not matter. |

## Value

The class of the returned object depends on the argument class(es). For an experiment-level object, two or more component data objects must be provided. Otherwise, if a single component-class is provided, it is simply returned as-is. The order of arguments does not matter.

## See Also

merge_phyloseq

## Examples

```
#
# # data(GlobalPatterns)
# # GP <- GlobalPatterns
# # phyloseq(sample_data(GP), otu_table(GP))
# # phyloseq(otu_table(GP), phy_tree(GP))
# # phyloseq(tax_table(GP), otu_table(GP))
# # phyloseq(phy_tree(GP), otu_table(GP), sample_data(GP))
# # phyloseq(otu_table(GP), tax_table(GP), sample_data(GP))
# # phyloseq(otu_table(GP), phy_tree(GP), tax_table(GP), sample_data(GP))
```

phyloseq-class *The main experiment-level class for phyloseq data*

## Description

Contains all component classes: otu_table-class, sample_data-class, taxonomyTable-class ("tax_table" slot), and phylo-class ("phy_tree" slot). There are several advantages to storing your phylogenetic sequencing experiment as an instance of the phyloseq class, not the least of which is that it is easy to return to the data later and feel confident that the different data types "belong" to one another. Furthermore, the phyloseq constructor ensures that the different data components have compatible indices (e.g. species and samples), and performs the necessary trimming automatically when you create your "experiment-level" object. Downstream analyses are aware of which data classes they require – and where to find them – often making your phyloseq-class object the only data argument to analysis and plotting functions (although there are many options and parameter arguments waiting for you).

## Details

In the case of missing component data, the slots are set to NULL. As soon as a phyloseq-class object is to be updated with new component data (previously missing/NULL or not), the indices of all components are re-checked for compatibility and trimmed if necessary. This is to ensure by design that components describe the same taxa/samples, and also that these trimming/validity checks do not need to be repeated in downstream analyses.

slots:

**otu_table** a single object of class otu_table.

**sam_data** a single object of class sample_data.

**tax_table** a single object of class taxonomyTable.

**phy_tree** a single object of the phylo-class, from the ape package

## See Also

The constructor, phyloseq, the merger merge_phyloseq, and also the component constructor/accessors otu_table, sample_data, tax_table, and phy_tree.

phy_tree *Retrieve phylogenetic tree (phylo-class) from object.*

## Description

This is the suggested method for accessing the phylogenetic tree, (phylo-class) from a phyloseq-class. Like other accessors (see See Also, below), the default behavior of this method is to stop with an error if physeq is a phyloseq-class but does not contain a phylogenetic tree.

## Usage

```
phy_tree(physeq, errorIfNULL=TRUE)

tre(physeq, errorIfNULL = TRUE)
```

## Arguments

| | |
|---|---|
| physeq | (Required). An instance of phyloseq-class that contains a phylogenetic tree. If physeq is a phylogenetic tree (a component data class), then it is returned as-is. |
| errorIfNULL | (Optional). Logical. Should the accessor stop with an error if the slot is empty (NULL)? Default TRUE. |

## Details

Note that the tip labels should be named to match the taxa_names of the other objects to which it is going to be paired. The phyloseq constructor automatically checks for exact agreement in the set of species described by the phlyogenetic tree and the other components (taxonomyTable, otu_table), and trims as-needed. Thus, the tip.labels in a phylo object must be named to match the results of taxa_names of the other objects to which it will ultimately be paired.

## Value

The phylo-class object contained within physeq; or NULL if physeq does not have a tree. This method stops with an error in the latter NULL case be default, which can be over-ridden by changing the value of errorIfNULL to FALSE.

## See Also

otu_table, sample_data, tax_table phyloseq, merge_phyloseq

## Examples

```
data(GlobalPatterns)
phy_tree(GlobalPatterns)
```

---

## phy_tree<- *Assign a (new) phylogenetic tree to* x

---

## Description

Assign a (new) phylogenetic tree to x

## Usage

```
phy_tree(x) <- value
```

## Arguments

| | |
|---|---|
| x | (Required). phyloseq-class |
| value | (Required). phylo-class, or phyloseq-class |

### Examples

```
#
# data(GlobalPatterns)
# # An example of pruning to just the first 100 taxa in GlobalPatterns.
# ex2a <- prune_species(taxa_names(GlobalPatterns)[1:100], GlobalPatterns)
# # The following 3 lines produces an ex2b that is equal to ex2a
# ex2b <- GlobalPatterns
# tree <- prune_species(taxa_names(GlobalPatterns)[1:100], phy_tree(GlobalPatterns))
# phy_tree(ex2b) <- tree
# identical(ex2a, ex2b)
# print(ex2b)
# # Example adding a phylo tree from phyloseq class
# ex2c <- phyloseq(otu_table(ex2b), sample_data(ex2b), tax_table(ex2b))
# phy_tree(ex2c) <- ex2b
# identical(ex2b, ex2c)
```

## plot_heatmap *Create an ecologically-organized heatmap using ggplot2 graphics*

### Description

In a 2010 article in BMC Genomics, Rajaram and Oono show describe an approach to creating a heatmap using ordination methods to organize the rows and columns instead of (hierarchical) cluster analysis. In many cases the ordination-based ordering does a much better job than h-clustering. An immediately useful example of their approach is provided in the NeatMap package for R. The NeatMap package can be used directly on the abundance table (otu_table-class) of phylogenetic-sequencing data, but the NMDS or PCA ordination options that it supports are not based on ecological distances. To fill this void, phyloseq provides the plot_heatmap() function as an ecology-oriented variant of the NeatMap approach to organizing a heatmap and build it using ggplot2 graphics tools. The distance and method arguments are the same as for the plot_ordination function, and support large number of distances and ordination methods, respectively, with a strong leaning toward ecology. This function also provides the options to re-label the OTU and sample axis-ticks with a taxonomic name and/or sample variable, respectively, in the hope that this might hasten your interpretation of the patterns (See the sample.label and species.label documentation, below). Note that this function makes no attempt to overlay hierarchical clustering trees on the axes, as hierarchical clustering is not used to organize the plot. Also note that each re-ordered axis repeats at the edge, and so apparent clusters at the far right/left or top/bottom of the heat-map may actually be the same. For now, the placement of this edge can be considered arbitrary, so beware of this artifact of this graphical representation. If you benefit from this phyloseq-specific implementation of the NeatMap approach, please cite both our packages/articles.

### Usage

```
plot_heatmap(physeq, method="NMDS", distance="bray",
sample.label=NULL, species.label=NULL, low="#000033",
high="#66CCFF", na.value="black", trans=log_trans(4),
max.label=250, title=NULL, ...)
```

### Arguments

| | |
|---|---|
| physeq | (Required). The data, in the form of an instance of the phyloseq-class. This should be what you get as a result from one of the import functions, or any |

| | |
|---|---|
| | of the processing downstream. No data components beyond the otu_table are strictly necessary, though they may be useful if you want to re-label the axis ticks according to some observable or taxonomic rank, for instance, or if you want to use a UniFrac-based distance (in which case your physeq data would need to have a tree included). |
| method | (Optional). The ordination method to use for organizing the heatmap. A great deal of the usefulness of a heatmap graphic depends upon the way in which the rows and columns are ordered. |
| distance | (Optional). A character string. The ecological distance method to use in the ordination. See distance. |
| sample.label | (Optional). A character string. The sample variable by which you want to re-label the sample (horizontal) axis. |
| species.label | (Optional). A character string. The taxonomic rank by which you want to re-label the species/OTU (vertical) axis. |
| low | (Optional). A character string. An R color. See colors for options support in R (there are lots). The color that represents the lowest non-zero value in the heatmap. Default is a dark blue color, "#000033". |
| high | (Optional). A character string. An R color. See colors for options support in R (there are lots). The color that will represent the highest value in the heatmap. The default is "#66CCFF". Zero-values are treated as NA, and set to "black", to represent a background color. |
| na.value | (Optional). A character string. An R color. See colors for options support in R (there are lots). The color to represent what is essentially the background of the plot, the non-observations that occur as NA or 0 values in the abundance table. The default is "black", which works well on computer-screen graphics devices, but may be a poor choice for printers, in which case you might want this value to be "white", and reverse the values of high and low, above. |
| trans | (Optional). "trans"-class transformer-definition object. A numerical transformer to use in the continuous color scale. See trans_new for details. The default is log_trans(4). |
| max.label | (Optional). Integer. Default is 250. The maximum number of labeles to fit on a given axis (either x or y). If number of taxa or samples exceeds this value, the corresponding axis will be stripped of any labels. This supercedes any arguments provided to sample.label or species.label. Make sure to increase this value if, for example, you want a special label for an axis that has 300 indices. |
| title | (Optional). Default NULL. Character string. The main title for the graphic. |
| ... | (Optional). Additional parameters passed to ordinate. |

## Details

This approach borrows heavily from the heatmap1 function in the NeatMap package. Highly recommended, and we are grateful for their package and ideas, which we have adapted for our specific purposes here, but did not use an explicit dependency. At the time of the first version of this implementation, the NeatMap package depends on the rgl-package, which is not needed in phyloseq, at present. Although likely a transient issue, the rgl-package has some known installation issues that have further influenced to avoid making NeatMap a formal dependency (Although we love both NeatMap and rgl!).

## Value

A heatmap plot, in the form of a ggplot plot object, which can be further saved and modified.

## References

Because this function relies so heavily in principle, and in code, on some of the functionality in NeatMap, please site their article if you use this function in your work.

Rajaram, S., & Oono, Y. (2010). NeatMap–non-clustering heat map alternatives in R. BMC Bioinformatics, 11, 45.

## Examples

```
data("GlobalPatterns")
gpac <- subset_species(GlobalPatterns, Phylum=="Crenarchaeota")
# FYI, the base-R function uses a non-ecological ordering scheme,
# but does add potentially useful hclust dendrogram to the sides...
heatmap(otu_table(gpac))
plot_heatmap(gpac)
# example relabelling based on a sample variable and taxonomic rank.
plot_heatmap(gpac, "NMDS", "bray", "SampleType", "Family")
# Now repeat the plot, but change the color scheme in various ways.
# This may be worthwhile, depending on the graphics device or paper
# on which you want to discplay the heatmap.
plot_heatmap(gpac, "NMDS", "bray", "SampleType", "Family", low="#000033", high="#CCFF66")
plot_heatmap(gpac, "NMDS", "bray", "SampleType", "Family", low="#000033", high="#FF3300")
plot_heatmap(gpac, "NMDS", "bray", "SampleType", "Family", low="#000033", high="#66CCFF")
plot_heatmap(gpac, "NMDS", "bray", "SampleType", "Family", low="#66CCFF", high="#000033", na.value="white
plot_heatmap(gpac, "NMDS", "bray", "SampleType", "Family", low="#FFFFCC", high="#000033", na.value="whit
# Now try the default color scheme, but using different ecological distances/ordinations
plot_heatmap(gpac, "NMDS", "jaccard")
plot_heatmap(gpac, "DCA")
plot_heatmap(gpac, "RDA")
plot_heatmap(gpac, "MDS", "bray")
plot_heatmap(gpac, "MDS", "unifrac")
plot_heatmap(gpac, "MDS", "unifrac", weighted=TRUE)
```

---

## plot_network *Plot a network using ggplot2 (represent microbiome)*

---

## Description

A custom plotting function for displaying networks using advanced ggplot2 formatting. The network itself should be represented using the igraph0 package. For the phyloseq-package it is suggested that the network object (argument g) be created using the make_network function, and based upon sample-wise or taxa-wise microbiome ecological distances calculated from a phylogenetic sequencing experiment (phyloseq-class). In this case, edges in the network are created if the distance between nodes is below a potentially arbitrary threshold, and special care should be given to considering the choice of this threshold.

## Usage

```
plot_network(g, physeq=NULL, type="samples", color=NULL,
shape=NULL, point_size=4, alpha=1, label="value", hjust =
1.35, line_weight=0.5, line_color=color, line_alpha=0.4,
layout.method=layout.fruchterman.reingold, title=NULL)
```

## Arguments

| | |
|---|---|
| g | (Required). An igraph0-class object created either by the convenience wrapper make_network, or directly by the tools in the igraph0-package. |
| physeq | (Optional). Default NULL. A phyloseq-class object on which g is based. |
| type | (Optional). Default "samples". Whether the network represented in the primary argument, g, is samples or taxa/OTUs. Supported arguments are "samples", "taxa", where "taxa" indicates using the taxa indices, whether they actually represent species or some other taxonomic rank. |
| color | (Optional). Default NULL. The name of the sample variable in physeq to use for color mapping of points (graph vertices). |
| shape | (Optional). Default NULL. The name of the sample variable in physeq to use for shape mapping. of points (graph vertices). |
| point_size | (Optional). Default 4. The size of the vertex points. |
| alpha | (Optional). Default 1. A value between 0 and 1 for the alpha transparency of the vertex points. |
| label | (Optional). Default "value". The name of the sample variable in physeq to use for labelling the vertex points. |
| hjust | (Optional). Default 1.35. The amount of horizontal justification to use for each label. |
| line_weight | (Optional). Default 0.3. The line thickness to use to label graph edges. |
| line_color | (Optional). Default color. The name of the sample variable in physeq to use for color mapping of lines (graph edges). |
| line_alpha | (Optional). Default 0.4. The transparency level for graph-edge lines. |
| layout.method | (Optional). Default layout.fruchterman.reingold. A function (closure) that determines the placement of the vertices for drawing a graph. Should be able to take an igraph0-class as sole argument, and return a two-column coordinate matrix with nrow equal to the number of vertices. For possible options already included in igraph0-package, see the others also described in the help file: |
| title | (Optional). Default NULL. Character string. The main title for the graphic. layout.fruchterman.reingold |

## Value

A ggplot2 plot representing the network, with optional mapping of variable(s) to point color or shape.

## References

This code was adapted from a repo original hosted on GitHub by Scott Chamberlain: https://github.com/SChamberlain/gggraph

The code most directly used/modified was first posted here: http://www.r-bloggers.com/basic-ggplot2-network-grap

### See Also

make_network

### Examples

```
data(enterotype)
ig <- make_network(enterotype, max.dist=0.3)
plot_network(ig, enterotype, color="SeqTech", shape="Enterotype", line_weight=0.3, label=NULL)
# Change distance parameter
ig <- make_network(enterotype, max.dist=0.2)
plot_network(ig, enterotype, color="SeqTech", shape="Enterotype", line_weight=0.3, label=NULL)
```

---

## plot_ordination — *General ordination plotter based on ggplot2.*

---

### Description

Convenience wrapper for plotting ordination results as a ggplot2-graphic, including additional annotation in the form of shading, shape, and/or labels of sample variables.

### Usage

```
plot_ordination(physeq, ordination, type="samples",
axes=c(1, 2), color=NULL, shape=NULL, label=NULL,
title=NULL, justDF=FALSE)
```

### Arguments

| | |
|---|---|
| physeq | (Required). phyloseq-class. The data about which you want to plot and annotate the ordination. |
| ordination | (Required). An ordination object. Many different classes of ordination are defined by R packages. The supported classes should be listed explicitly, but in the meantime, all ordination classes currently supported by the scores function are supported here. There is no default, as the expectation is that the ordination will be performed and saved prior to calling this plot function. |
| type | (Optional). The plot type. Default is "samples". The currently supported options are c("samples", "sites", "species", "taxa", "biplot", "split"). The option “taxa” is equivalent to “species” in this case, and similarly, “samples” is equivalent to “sites”. The options "sites" and "species" result in a single-plot of just the sites/samples or species/taxa of the ordination, respectively. The "biplot" and "split" options result in a combined plot with both taxa and samples, either combined into one plot (“biplot”) or separated in two facet panels (“split”), respectively. |
| axes | (Optional). A 2-element vector indicating the axes of the ordination that should be used for plotting. Can be character-class or integer-class, naming the index name or index of the desired axis for the horizontal and vertical axes, respectively, in that order. The default value, c(1, 2), specifies the first two axes of the provided ordination. |

| | |
|---|---|
| color | (Optional). Default NULL. Character string. The name of the variable to map to colors in the plot. This can be a sample variable (among the set returned by sample_variables(physeq) ) or taxonomic rank (among the set returned by rank_names(physeq)).<br>Alternatively, if type indicates a single-plot ("samples" or "species"), then it is also possible to supply a custom vector with length equal to the relevant number of samples or species (nsamples(physeq) or ntaxa(physeq)).<br>Finally, The color scheme is chosen automatically by link{ggplot}, but it can be modified afterward with an additional layer using scale_color_manual. |
| shape | (Optional). Default NULL. Character string. The name of the variable to map to different shapes on the plot. Similar to color option, but for the shape if points.<br>The shape scale is chosen automatically by link{ggplot}, but it can be modified afterward with an additional layer using scale_shape_manual. |
| label | (Optional). Default NULL. Character string. The name of the variable to map to text labels on the plot. Similar to color option, but for plotting text. |
| title | (Optional). Default NULL. Character string. The main title for the graphic. |
| justDF | (Optional). Default FALSE. Logical. Instead of returning a ggplot2-object, do you just want the relevant data.frame that was used to build the plot? This is a user-accessible option for obtaining the data.frame, in in principal to make a custom plot that isn't possible with the available options in this function. For contributing new functions (developers), the phyloseq-package provides/uses an internal function to build the key features of the data.frame prior to plot-build. |

## Value

A ggplot plot object, graphically summarizing the ordination result for the specified axes.

## See Also

The examples on the phyloseq wiki page for plot_ordination show many more examples:

https://github.com/joey711/phyloseq/wiki/plot_ordination

Also see the general wrapping function:

plot_phyloseq

## Examples

```
data(GlobalPatterns)
# Need to clean the zeros from GlobalPatterns:
GP <- prune_taxa(taxa_sums(GlobalPatterns)>0, GlobalPatterns)
# Define a human-associated versus non-human binary variable:
sample_data(GP)$human <- get_variable(GP, "SampleType") %in% c("Feces", "Mock", "Skin", "Tongue")
# Get the names of the most-abundant
top.TaxaGroup <- sort(
tapply(taxa_sums(GP), tax_table(GP)[, "Phylum"], sum, na.rm = TRUE),
decreasing = TRUE)
top.TaxaGroup <- top.TaxaGroup[top.TaxaGroup > 1*10^6]
# Now prune further, to just the most-abundant phyla
GP <- subset_taxa(GP, Phylum %in% names(top.TaxaGroup))
topsp <- names(sort(taxa_sums(GP), TRUE)[1:200])
GP1   <- prune_taxa(topsp, GP)
GP.dpcoa <- ordinate(GP1, "DPCoA")
```

```
plot_ordination(GP1, GP.dpcoa, type="taxa", color="Phylum")
# Customize with ggplot2 layers added directly to output
library("ggplot2")
plot_ordination(GP1, GP.dpcoa, type="samples", color="SampleType") + geom_line() + geom_point(size=5)
p <- plot_ordination(GP1, GP.dpcoa, type="samples", color="SampleType", shape="human")
print(p)
# library("ggplot2")
# p + geom_line() + geom_point(size=5)
# plot_ordination(GP1, GP.dpcoa, type="taxa", color="Phylum") + geom_line() + geom_point(size=5)
plot_ordination(GP1, GP.dpcoa, type="biplot", shape="Phylum", label="SampleType")
plot_ordination(GP1, GP.dpcoa, type="biplot", shape="Phylum")
plot_ordination(GP1, GP.dpcoa, type="biplot", color="Phylum")
plot_ordination(GP1, GP.dpcoa, type="biplot", label="Phylum")
plot_ordination(GP1, GP.dpcoa, type="split", color="Phylum", label="SampleType")
plot_ordination(GP1, GP.dpcoa, type="split", color="SampleType", shape="Phylum", label="SampleType")
```

## plot_phyloseq *Generic plot defaults for phyloseq.*

### Description

The specific plot type is chosen according to available non-empty slots. This is mainly for syntactic convenience and quick-plotting. See links below for some examples of available graphics tools available in the phyloseq-package.

### Usage

```
plot_phyloseq(physeq, ...)
```

### Arguments

| | |
|---|---|
| physeq | (Required). phyloseq-class. The actual plot type depends on the available (non-empty) component data types contained within. |
| ... | (Optional). Additional parameters to be passed on to the respective specific plotting function. See below for different plotting functions that might be called by this generic plotting wrapper. |

### Value

A plot is created. The nature and class of the plot depends on the physeq argument, specifically, which component data classes are present.

### See Also

phyloseq graphics examples (wiki).

plot_ordination plot_heatmap plot_tree plot_network plot_taxa_bar plot_richness

### Examples

```
data(esophagus)
plot_phyloseq(esophagus)
```

plot_richness *Plot richness estimates, flexibly with ggplot2*

## Description

Performs a number of standard richness estimates using the estimate_richness function, and returns a ggplot plotting object. This plot shows the individual richness estimates for each sample, as well as the observed richness. You must use untrimmed datasets for meaningful results, as these estimates (and even the "observed" richness) are highly dependent on the number of singletons. You can always trim the data later on if needed, just not before using this function.

## Usage

```
plot_richness(physeq, x, color=NULL, shape=NULL,
title=NULL)

plot_richness_estimates(physeq, x = "sample_names", color
= NULL, shape = NULL, title = NULL)
```

## Arguments

| | |
|---|---|
| physeq | (Required). phyloseq-class, or alternatively, an otu_table-class. The data about which you want to estimate the richness. |
| x | (Optional). A variable to map to the horizontal axis. The vertical axis will be mapped to richness estimates and have units of total taxa. This parameter (x) can be either a character string indicating a variable in sample_data (among the set returned by sample_variables(physeq) ); or a custom supplied vector with length equal to the number of samples in the dataset (nsamples(physeq)). The default value is "sample_names", which will map each sample's name to a separate horizontal position in the plot. |
| color | (Optional). Default NULL. The sample variable to map to different colors. Like x, this can be a single character string of the variable name in sample_data (among the set returned by sample_variables(physeq) ); or a custom supplied vector with length equal to the number of samples in the dataset (nsamples(physeq)). The color scheme is chosen automatically by link{ggplot}, but it can be modified afterward with an additional layer using scale_color_manual. |
| shape | (Optional). Default NULL. The sample variable to map to different shapes. Like x and color, this can be a single character string of the variable name in sample_data (among the set returned by sample_variables(physeq) ); or a custom supplied vector with length equal to the number of samples in the dataset (nsamples(physeq)). The shape scale is chosen automatically by link{ggplot}, but it can be modified afterward with an additional layer using scale_shape_manual. |
| title | (Optional). Default NULL. Character string. The main title for the graphic. |

## Details

NOTE: Because this plotting function incorporates the output from estimate_richness, the variable names of that output should not be used as x or color (even if it works, the resulting plot might be kindof strange, and not the intended behavior of this function). The following are the names you will want to avoid using in x or color:

```
c("S.obs", "S.chao1", "se.chao1", "S.ACE",  "se.ACE", "shannon", "simpson")
```

## Value

A ggplot plot object summarizing the richness estimates, and their standard error.

## See Also

estimate_richness, estimateR, diversity

## Examples

```
data(GlobalPatterns)
plot_richness(GlobalPatterns, "SampleType")
plot_richness(GlobalPatterns, "SampleType", "SampleType")
# # Define a human-associated versus non-human categorical variable:
GP <- GlobalPatterns
human <- get_variable(GP, "SampleType") %in%
c("Feces", "Mock", "Skin", "Tongue")
names(human) <- sample_names(GP)
# # Replace current SD with new one that includes human variable:
sample_data(GP)$human <- human
# # Can use new "human" variable within GP as a discrete variable in the plot
plot_richness(GP, "human", "SampleType")
plot_richness(GP, "SampleType", "human")
# # Can also provide custom factor directly:
plot_richness(GP, "SampleType", human)
plot_richness(GP, human, "SampleType")
# # Not run: Should cause an error:
# plot_richness(GP, "value", "value")
# #
```

---

plot_taxa_bar *Create a structured barplot graphic of the taxonomic groups.*

---

## Description

This function wraps ggplot2 plotting, and returns a ggplot2 graphic object that can be saved or further modified with additional layers, options, etc. The main purpose of this function is to quickly and easily create informative summary graphics of the differences in taxa abundance between samples in an experiment.

## Usage

```
plot_taxa_bar(physeq, taxavec="Domain",
showOnlyTheseTaxa=NULL, threshold=NULL, x="sample",
fill=x, facet_formula = . ~ TaxaGroup, OTUpoints=FALSE,
labelOTUs=FALSE, title=NULL)

taxaplot(physeq, taxavec = "Domain", showOnlyTheseTaxa =
NULL, threshold = NULL, x = "sample", fill = x,
facet_formula = . ~ TaxaGroup, OTUpoints = FALSE,
labelOTUs = FALSE, title = NULL)
```

## Arguments

| | |
|---|---|
| physeq | (Required). An otu_table-class or phyloseq-class. If physeq does not contain a taxonomyTable component, then the second argument, taxavec, is required and should have length equal to the number of species/taxa in physeq. |
| taxavec | (Optional, but recommended). A character vector of the desired taxonomic rank to use in grouping each species/OTU in physeq. If physeq contains a taxonomyTable, then taxavec can alternatively specify the desired taxonomic level as a character string of length 1. E.g. taxavec = "Phylum". Default value is "Domain". |
| showOnlyTheseTaxa | A vector of the taxonomic labels that you want included. If NULL, the default, then all taxonomic labels are used, except for the empty character string, “”, which is trimmed away. |
| threshold | (Optional). A [0,1] numeric. Fraction of abundance of the taxonomic groups to keep for each sample. The higher the value, the larger the diversity of taxonomica groups included. That is, a greater number of the rare groups are included. If NULL (or 1), the default, all taxonomic groups are included. |
| x | (Optional). A character string indicating which sample variable should be used to define the horizontal axis categories. Default is "sample". Note that a few column-names are added by default and are available as options. They are “sample”, “Abundance”, and “TaxaGroup”. |
| fill | (Optional). A character string. Indicates which sample variable should be used to define the fill color of the bars. This does not have to match x, but does so by default. Note that a few column-names are added by default and are available as options. They are “sample”, “Abundance”, and “TaxaGroup”. |
| facet_formula | A formula object as used by facet_grid in ggplot or qplot commands The default is: . ~ TaxaGroup. Note that a few column-names are added by default and are available as options. They are “sample”, “Abundance”, and “TaxaGroup”. E.g. An alternative facet_grid could be sample ~ TaxaGroup. |
| OTUpoints | (Optional). Logical. Default FALSE. Whether to add small grey semi-transparent points for each OTU. Helps convey the relative distribution within each bar if it combines many different OTUs. For datasets with large numbers of samples and for complicated plotting arrangements, this might be too cluttered to be meaningful. |
| labelOTUs | (Optional). Logical. Default FALSE. Whether to add a label over the top few OTUs within each bar. As with OTUpoints, this is probably not a good idea for plots with large complexity. For low numbers of total OTUs this can be informative, and help display multiple layers of information on the same graphic. |
| title | (Optional). Default NULL. Character string. The main title for the graphic. |

## Details

The vertical axis is always relative abundance, but the data can be further organized at the horizontal axis and faceting grid by any combination of variates present in the sample_data component of physeq.

## Value

A ggplot2 graphic object.

**See Also**

otu2df, qplot, ggplot

**Examples**

```
data(enterotype)
TopNOTUs <- names(sort(taxa_sums(enterotype), TRUE)[1:10])
ent10   <- prune_species(TopNOTUs, enterotype)
plot_taxa_bar(ent10, "Genus", x="SeqTech", fill="TaxaGroup")
library("ggplot2")
plot_taxa_bar(ent10, "Genus", x="SeqTech", fill="TaxaGroup") + facet_wrap(~Enterotype)
```

## plot_tree    *Plot a phylogenetic tree with optional annotations*

**Description**

This function is intended to facilitate easy graphical investigation of the phylogenetic tree, as well as sample data. Note that for phylogenetic sequencing of samples with large richness, some of the options in this function will be prohibitively slow to render, or too dense to be interpretable. A rough "rule of thumb" is to use subsets of data with not many more than 200 taxa per plot, sometimes less depending on the complexity of the additional annotations being mapped to the tree. It is usually possible to create an unreadable, uninterpretable tree with modern datasets. However, the goal should be toward parameter settings and data subsets that convey (honestly, accurately) some biologically relevant feature of the data. One of the goals of the phyloseq-package is to make the determination of these features/settings as easy as possible.

**Usage**

```
plot_tree(physeq, method="sampledodge", color=NULL,
shape=NULL, size=NULL, min.abundance=Inf,
label.tips=NULL, text.size=NULL, sizebase=5,
base.spacing=0.02, title=NULL)
```

**Arguments**

| | |
|---|---|
| physeq | (Required). The data about which you want to plot and annotate a phylogenetic tree, in the form of a single instance of the phyloseq-class, containing at minimum a phylogenetic tree component (try phy_tree). One of the major advantages of this function over basic tree-plotting utilities in the ape-package is the ability to easily annotate the tree with sample variables and taxonomic information. For these uses, the physeq argument should also have a sample_data and/or tax_table component(s). |
| method | (Optional). Character string. Default "sampledodge". The name of the annotation method to use. This will be expanded in future versions. Currently only "sampledodge" and "treeonly" are supported. The "sampledodge" option results in points drawn next to leaves if individuals from that taxa were observed, and a separate point is drawn for each sample. |
| color | (Optional). Character string. Default NULL. The name of the variable in physeq to map to point color. |

| | |
|---|---|
| shape | (Optional). Character string. Default NULL. The name of the variable in physeq to map to point shape. |
| size | (Optional). Character string. Default NULL. The name of the variable in physeq to map to point size. A special argument "abundance" is reserved here and scales point size using abundance in each sample on a log scale. |
| min.abundance | (Optional). Numeric. The minimum number of individuals required to label a point with the precise number. Default is Inf, meaning that no points will have their abundance labeled. If a vector, only the first element is used. |
| label.tips | (Optional). Character string. Default is NULL, indicating that no tip labels will be printed. If "taxa_names", then the name of the taxa will be added to the tree; either next to the leaves, or next to the set of points that label the leaves. Alternatively, if this is one of the rank names (from rank_names(physeq)), then the identity (if any) for that particular taxonomic rank is printed instead. |
| text.size | (Optional). Numeric. Should be positive. The size parameter used to control the text size of taxa labels. Default is NULL. If left NULL, this function will automatically calculate a (hopefully) optimal text size given the vertical constraints posed by the tree itself. This argument is included in case the automatically-calculated size is wrong, and you want to change it. Note that this parameter is only meaningful if label.tips is not NULL. |
| sizebase | (Optional). Numeric. Should be positive. The base of the logarithm used to scale point sizes to graphically represent abundance of species in a given sample. Default is 5. |
| base.spacing | (Optional). Numeric. Default is 0.02. Should be positive. This defines the base-spacing between points at each tip/leaf in the the tree. The larger this value, the larger the spacing between points. This is useful if you have problems with overlapping large points and/or text indicating abundance, for example. Similarly, if you don't have this problem and want tighter point-spacing, you can shrink this value. |
| title | (Optional). Default NULL. Character string. The main title for the graphic. |

## Details

This function received a development contribution from the work of Gregory Jordan for the ggphylo package, which provides tools for rendering a phylogenetic tree in ggplot2 graphics. That package is not currently available from CRAN or Bioconductor, but is available in development (roughly "beta") form from GitHub. Furthermore, although ggphylo awesomely supports radial and unrooted trees, plot_tree currently only supports "standard" square-horizontal trees. Additional support for these types of features (like radial trees) is planned. Send us development feedback if this is a feature you really want to have soon.

## Value

A ggplot2 plot.

## Author(s)

Paul McMurdie, relying on supporting code from Gregory Jordan <gjuggler@gmail.com>

### See Also

plot.phylo

This function is a special use-case that relies on code borrowed directly, with permission, from the not-yet-officially-released package, ggphylo, currently only available from GitHub at: https://github.com/gjuggler/ggphylo

### Examples

```
## Using plot_tree() with the esophagus dataset.
data(esophagus)
plot_tree(esophagus)
plot_tree(esophagus, color="samples")
plot_tree(esophagus, size="abundance")
plot_tree(esophagus, size="abundance", color="samples")
plot_tree(esophagus, size="abundance", color="samples", base.spacing=0.03)
## Using plot_tree with the Global Patterns dataset
data("GlobalPatterns")
# Subset Global Patterns dataset to just the observed Archaea
gpa <- subset_species(GlobalPatterns, Kingdom=="Archaea")
# The number of different Archaeal species from this dataset is small enough ...
ntaxa(gpa)
# ... that it is reasonable to consider displaying the phylogenetic tree directly.
# (probably not true of the total dataset)
ntaxa(GlobalPatterns)
# Some patterns are immediately discernable with minimal parameter choices:
# plot_tree(gpa, color="SampleType")
# plot_tree(gpa, color="Phylum")
# plot_tree(gpa, color="SampleType", shape="Phylum")
# plot_tree(gpa, color="Phylum", label.tips="Genus")
## However, the text-label size scales with number of species, and with common
## graphics-divice sizes/resolutions, these ~200 taxa still make for a
## somewhat crowded graphic.
##
## Let's instead subset further ot just the Crenarchaeota
gpac <- subset_species(gpa, Phylum=="Crenarchaeota")
plot_tree(gpac, color="SampleType", shape="Genus")
# plot_tree(gpac, color="SampleType", label.tips="Genus")
## Let's add some abundance information.
## Notice that the default spacing gets a little crowded when we map
## species-abundance to point-size:
# plot_tree(gpac, color="SampleType", shape="Genus", size="abundance")
## So let's spread it out a little bit with the base.spacing parameter.
# plot_tree(gpac, color="SampleType", shape="Genus", size="abundance", base.spacing=0.05)
```

---

## prune_samples *Prune unwanted samples from a phyloseq object.*

---

### Description

An S4 Generic method for removing (pruning) unwanted samples.

### Usage

```
prune_samples(samples, x)
```

### Arguments

| | |
|---|---|
| samples | (Required). A character vector of the samples in object x that you want to keep – OR alternatively – a logical vector where the kept samples are TRUE, and length is equal to the number of samples in object x. If samples is a named logical, the samples retained is based on those names. Make sure they are compatible with the taxa_names of the object you are modifying (x). |
| x | A phyloseq object. |

### Value

The class of the object returned by prune_samples matches the class of the phyloseq object, x.

### See Also

subset_samples

### Examples

```
#
data(GlobalPatterns)
# Subset to just the Chlamydiae phylum.
GP.chl <- subset_taxa(GlobalPatterns, Phylum=="Chlamydiae")
# Remove the samples that have less than 20 total reads from Chlamydiae
GP.chl <- prune_samples(sampleSums(GP.chl)>=20, GP.chl)
# (p <- plot_tree(GP.chl, color="SampleType", shape="Family", label.tips="Genus", size="abundance"))
```

---

## prune_taxa *Prune unwanted OTUs / taxa from a phylogenetic object.*

---

### Description

An S4 Generic method for removing (pruning) unwanted OTUs/taxa from phylogenetic objects, including phylo-class trees, as well as native phyloseq package objects. This is particularly useful for pruning a phyloseq object that has more than one component that describes OTUs. Credit: the phylo-class version is adapted from prune.sample.

### Usage

```
prune_species(taxa, x)
```

### Arguments

| | |
|---|---|
| taxa | (Required). A character vector of the taxa in object x that you want to keep – OR alternatively – a logical vector where the kept taxa are TRUE, and length is equal to the number of taxa in object x. If taxa is a named logical, the taxa retained are based on those names. Make sure they are compatible with the taxa_names of the object you are modifying (x). |
| x | (Required). A phylogenetic object, including phylo trees, as well as all phyloseq classes that represent taxa. If the function taxa_names returns a non-NULL value, then your object can be pruned by this function. |

### Value

The class of the object returned by prune_taxa matches the class of the argument, x.

### See Also

prune_taxa

prune.sample

### Examples

```
#
## testOTU <- otu_table(matrix(sample(1:50, 25, replace=TRUE), 5, 5), taxa_are_rows=FALSE)
## f1  <- filterfun_sample(topk(2))
## wh1 <- genefilter_sample(testOTU, f1, A=2)
## wh2 <- c(T, T, T, F, F)
## prune_taxa(wh1, testOTU)
## prune_taxa(wh2, testOTU)
##
## tax_table1 <- tax_table(matrix("abc", 5, 5))
## prune_taxa(wh1, tax_table1)
## prune_taxa(wh2, tax_table1)
```

---

## rank_names *Retrieve the names of the taxonomic ranks*

---

### Description

This is a simple accessor function to make it more convenient to determine the taxonomic ranks that are available in a given phyloseq-class object.

### Usage

```
rank_names(physeq, errorIfNULL=TRUE)

rank.names(physeq, errorIfNULL = TRUE)
```

### Arguments

| | |
|---|---|
| physeq | (Required). taxonomyTable-class, or phyloseq-class. |
| errorIfNULL | (Optional). Logical. Should the accessor stop with an error if the slot is empty (NULL)? Default TRUE. |

### Value

Character vector. The names of the available taxonomic ranks.

### See Also

get_taxa taxa_names sample_names

## Examples

```
data(enterotype)
rank_names(enterotype)
```

---

rarefy_even_depth *Perform a random subsampling of an OTU table to a level of even depth.*

---

## Description

This function uses the sample function to randomly subset from the abundance values in each sample of the otu_table component in the physeq argument. Sampling is performed with replacement from a vector of taxa indices, with length equal to the argument to sample.size, and probability according to the abundances for that sample in physeq.

## Usage

```
rarefy_even_depth(physeq,
sample.size=min(sample_sums(physeq)))
```

## Arguments

| | |
|---|---|
| physeq | (Required). A phyloseq-class object that you want to trim/filter. |
| sample.size | (Optional). A single integer value equal to the number of reads being simulated, also known as the depth, and also equal to each value returned by sample_sums on the output. |

## Details

This is sometimes (somewhat mistakenly) called "rarefaction", though it actually a single random subsampling procedure in this case. The original rarefaction procedure includes many random subsampling iterations at increasing depth as a means to infer richness/alpha-diversity

Make sure to use set.seed for exactly-reproducible results of the random subsampling.

## Value

An object of class phyloseq. Only the otu_table component is modified.

## See Also

sample

set.seed

## Examples

```
set.seed(711)
# Test with esophagus dataset
data("esophagus")
eso <- rarefy_even_depth(esophagus)
plot(as(otu_table(eso), "vector"), as(otu_table(esophagus), "vector"))
UniFrac(eso); UniFrac(esophagus)
# Test with GlobalPatterns dataset
data("GlobalPatterns")
GP.chl <- subset_taxa(GlobalPatterns, Phylum=="Chlamydiae")
# remove the samples that have less than 20 total reads from Chlamydiae
GP.chl <- prune_samples(names(which(sample_sums(GP.chl)>=20)), GP.chl)
# # (p <- plot_tree(GP.chl, color="SampleType", shape="Family", label.tips="Genus", size="abundance"))
GP.chl.r <- rarefy_even_depth(GP.chl)
plot(as(otu_table(GP.chl.r), "vector"), as(otu_table(GP.chl), "vector"))
# Try ordination of GP.chl and GP.chl.r (default distance is unweighted UniFrac)
plot_ordination(GP.chl, ordinate(GP.chl, "MDS"), color="SampleType") #+ geom_point(size=5)
plot_ordination(GP.chl.r, ordinate(GP.chl.r, "MDS"), color="SampleType") #+ geom_point(size=5)
```

---

read_tree *Somewhat flexible tree-import function*

---

## Description

This function is a convenience wrapper around the read.tree (Newick-format) and read.nexus (Nexus-format) importers provided by the ape-package. This function attempts to return a valid tree if possible using either format importer. If it fails, it silently returns NULL by default, rather than throwing a show-stopping error.

## Usage

```
read_tree(treefile, errorIfNULL = FALSE, ...)
```

## Arguments

| | |
|---|---|
| treefile | (Required). A character string implying a file connection (like a path or URL), or an actual connection. Must be a Newick- or Nexus-formatted tree. |
| errorIfNULL | (Optional). Logical. Should an error be thrown if no tree can be extracted from the connection? Default is FALSE, indicating that NULL will be SILENTLY returned, rather than an error. Be cautious about this behavior. Useful for phyloseq internals, but might be hard to track in treefileyour own code if you're not aware of this "no error by default" setting. If this is a problem, change this value to TRUE, and you can still use the function. |
| ... | (Optional). Additional parameter(s) passed to the relevant tree-importing function. |

## Details

read_tree(treefile, errorIfNULL=FALSE, ...)

## Value

If successful, returns a phylo-class object as defined in the ape-package. Returns NULL if neither tree-reading function worked.

## See Also

phylo, read.tree, read.nexus

## Examples

```
read_tree(system.file("extdata", "esophagus.tree.gz", package="phyloseq"))
read_tree(system.file("extdata", "GP_tree_rand_short.newick.gz", package="phyloseq"))
```

---

## rm_outlierf *Set to FALSE any outlier species greater than f fractional abundance.*

---

### Description

This is for removing overly-abundant outlier taxa, not for trimming low-abundance taxa.

### Usage

```
rm_outlierf(f, na.rm=TRUE)
```

### Arguments

| | |
|---|---|
| f | Single numeric value between 0 and 1. The maximum fractional abundance value that a taxa will be allowed to have in a sample without being marked for trimming. |
| na.rm | Logical. Should we remove NA values. Default TRUE. |

### Value

A function (enclosure), suitable for filterfun_sample.

### See Also

topk, topf, topp, rm_outlierf

### Examples

```
t1 <- 1:10; names(t1)<-paste("t", 1:10, sep="")
rm_outlierf(0.15)(t1)
## Use simulated abundance matrix
# set.seed(711)
# testOTU <- otu_table(matrix(sample(1:50, 25, replace=TRUE), 5, 5), taxa_are_rows=FALSE)
# taxa_sums(testOTU)
# f1  <- filterfun_sample(rm_outlierf(0.1))
# (wh1 <- genefilter_sample(testOTU, f1, A=1))
# wh2 <- c(T, T, T, F, F)
# prune_taxa(wh1, testOTU)
# prune_taxa(wh2, testOTU)
```

## sample_data *Build or access sample_data.*

### Description

This is the suggested method for both constructing and accessing a table of sample-level variables (sample_data-class), which in the phyloseq-package is represented as a special extension of the data.frame-class. When the argument is a data.frame, sample_data() will create a sample_data-class object. In this case, the rows should be named to match the sample.names of the other objects to which it will ultimately be paired. Alternatively, if the first argument is an experiment-level (phyloseq-class) object, then the corresponding sample_data is returned. Like other accessors (see See Also, below), the default behavior of this method is to stop with an error if object is a phyloseq-class but does not contain a sample_data.

### Usage

```
sample_data(object, errorIfNULL=TRUE)

sam_data(object, errorIfNULL = TRUE)

sampleData(object, errorIfNULL = TRUE)

samData(object, errorIfNULL = TRUE)
```

### Arguments

| | |
|---|---|
| object | (Required). A data.frame-class, or a phyloseq-class object. |
| errorIfNULL | (Optional). Logical. Should the accessor stop with an error if the slot is empty (NULL)? Default TRUE. |

### Details

Note that the sam_data() and sampleMap() functions are provided for convenience and backward compatibility, respectively, but should provide the exact same behavior as sample_data().

### Value

A sample_data-class object representing the sample variates of an experiment.

### See Also

phy_tree, tax_table, otu_table phyloseq, merge_phyloseq

### Examples

```
#
# data(GlobalPatterns)
# sample_data(GlobalPatterns)
## shorter (convenience) wrapper of sample_data()
# sam_data(GlobalPatterns)
```

## sample_data-class *The S4 for storing sample variables.*

### Description

Row indices represent samples, while column indices represent experimental categories, variables (and so forth) that describe the samples.

### Details

**.Data** data-frame data, inherited from the data.frame class.

**row.names** Also inherited from the data.frame class; it should contain the sample names.

**names** Inherited from the data.frame class.

## sample_data<- *Assign (new) sample_data to* x

### Description

This replaces the current sample_data component of x with value, if value is a sample_data-class. However, if value is a data.frame, then value is first coerced to a sample_data-class, and then assigned. Alternatively, if value is phyloseq-class, then the sample_data component will first be accessed from value and then assigned. This makes possible some concise assignment/replacement statements when adjusting, modifying, or building subsets of experiment-level data. See some examples below.

### Usage

```
sample_data(x) <- value

sam_data(x) <- value
```

### Arguments

| | |
|---|---|
| x | (Required). phyloseq-class. The object to modify. |
| value | (Required). Either a sample_data-class, a data.frame that can be coerced into sample_data-class, or a phyloseq-class that contains a suitable sample_data component to assign to x. If unsure, try sample_data(value), which should return a sample_data-class object without error. |

### Details

Internally, this re-builds the phyloseq-class object using the standard phyloseq constructor. Thus, index mismatches between sample-describing components will not be allowed, and subsetting will occurr automatically such that only the intersection of sample IDs are included in any components. This has the added benefit of re-checking (internally) for any other issues.

### Value

No return. This is an assignment statement.

### Examples

```
#
# data(GlobalPatterns)
# # An example of pruning to just the first 10 samples in GlobalPatterns
# ex2a <- prune_samples(sample_names(GlobalPatterns)[1:10], GlobalPatterns)
# # The following 3 lines produces an ex2b that is equal to ex2a
# ex2b <- GlobalPatterns
# SD <- sample_data(GlobalPatterns)[1:10, ]
# sample_data(ex2b) <- SD
# identical(ex2a, ex2b)
# print(ex2b)
# # Example restoring the original sample_data component. ex2c lacks sample_data
# ex2c <- phyloseq(otu_table(GlobalPatterns), tax_table(GlobalPatterns), phy_tree(GlobalPatterns))
# sample_data(ex2c) <- GlobalPatterns
# identical(ex2c, GlobalPatterns)
# # Can try on ex2b, but other components have only 10 samples. No change.
# sample_data(ex2b) <- GlobalPatterns
# identical(ex2a, ex2b) # still true.
```

---

## sample_names    *Get sample names.*

---

### Description

Get sample names.

### Usage

```
sample_names(physeq)

sampleNames(physeq)

sample.names(physeq)
```

### Arguments

| | |
|---|---|
| physeq | (Required). A phyloseq-class, sample_data, or otu_table-class. |

### Value

A character vector. The names of the samples in physeq.

### See Also

taxa_names, nsamples

## Examples

```
#
# # From "picante" package
# data(GlobalPatterns)
# sample_names(GlobalPatterns)
```

---

sample_sums *Returns the total number of individuals observed from each sample.*

---

## Description

A convenience function equivalent to rowSums or colSums, but where the orientation of the otu_table is automatically handled.

## Usage

```
sample_sums(x)

sampleSums(x)
```

## Arguments

| | |
|---|---|
| x | otu_table-class, or phyloseq-class. |

## Value

A named numeric-class length equal to the number of samples in the x, name indicating the sample ID, and value equal to the sum of all individuals observed for each sample in x.

## See Also

taxa_sums, rowSums, colSums

## Examples

```
data(enterotype)
sample_sums(enterotype)
data(esophagus)
sample_sums(esophagus)
```

## sample_variables — *Get the sample variables present in sample_data*

### Description

This is a simple accessor function to make it more convenient to determine the sample variable names of a particular phyloseq-class object.

### Usage

```
sample_variables(physeq, errorIfNULL=TRUE)

sample.variables(physeq, errorIfNULL = TRUE)
```

### Arguments

| | |
|---|---|
| physeq | (Required). sample_data-class, or phyloseq-class. |
| errorIfNULL | (Optional). Logical. Should the accessor stop with an error if the slot is empty (NULL)? Default TRUE. |

### Value

Character vector. The names of the variables in the sample_data data.frame. Essentially the column names. Useful for selecting model and graphics parameters that interact with sample_data.

### See Also

get_taxa taxa_names sample_names

### Examples

```
data(enterotype)
sample_variables(enterotype)
```

## show — *method extensions to show for phyloseq objects.*

### Description

See the general documentation of show method for expected behavior.

### See Also

show

### Examples

```
# data(GlobalPatterns)
# show(GlobalPatterns)
# GlobalPatterns
```

## show_mothur_list_cutoffs — *Show cutoff values available in a mothur list file*

### Description

This is a helper function to report back to the user the different cutoff values available in a given *list* file created by the OTU clustering and analysis package called *mothur*

### Usage

```
show_mothur_list_cutoffs(mothur_list_file)
```

### Arguments

| | |
|---|---|
| mothur_list_file | The list file name and/or location as produced by *mothur*. |

### Value

A character vector of the different cutoff values contained in the file. For a given set of arguments to the cluster() command from within *mothur*, a number of OTU-clustering results are returned in the same list file. The exact cutoff values used by *mothur* can vary depending on the input data. This simple function returns the cutoffs that were actually included in the *mothur* output. This an important extra step prior to importing the OTUs with the import_mothur_otulist() function.

### See Also

import_mothur

## subset_ord_plot — *Subset points from an ordination-derived ggplot*

### Description

Easily retrieve a plot-derived data.frame with a subset of points according to a threshold and method. The meaning of the threshold depends upon the method. See argument description below.

### Usage

```
subset_ord_plot(p, threshold=0.05, method="farthest")
```

## Arguments

| | |
|---|---|
| p | (Required). A ggplot object created by plot_ordination. It contains the complete data that you want to subset. |
| threshold | (Optional). A numeric scalar. Default is 0.05. This value determines a coordinate threshold or population threshold, depending on the value of the method argument, ultimately determining which points are included in returned data.frame. |
| method | (Optional). A character string. One of c("farthest", "radial", "square"). Default is "farthest". This determines how threshold will be interpreted. |

**farthest** Unlike the other two options, this option implies removing a certain fraction or number of points from the plot, depending on the value of threshold. If threshold is greater than or equal to 1, then all but threshold number of points farthest from the origin are removed. Otherwise, if threshold is less than 1, all but threshold fraction of points farthests from origin are retained.

**radial** Keep only those points that are beyond threshold radial distance from the origin. Has the effect of removing a circle of points from the plot, centered at the origin.

**square** Keep only those points with at least one coordinate greater than threshold. Has the effect of removing a “square” of points from the plot, centered at the origin.

## Value

A data.frame suitable for creating a ggplot plot object, graphically summarizing the ordination result according to previously-specified parameters.

## See Also

plot_ordination

## Examples

```
data(GlobalPatterns)
# Need to clean the zeros from GlobalPatterns:
GP <- GlobalPatterns
GP <- prune_species(taxa_sums(GP)>0, GP)
# # Add "human" variable to GP
human <- get_variable(GP, "SampleType") %in%
c("Feces", "Mock", "Skin", "Tongue")
names(human) <- sample_names(GP)
sample_data(GP)$human <- human
# Get the names of the most-abundant phyla
top.TaxaGroup <- sort(
tapply(taxa_sums(GP), tax_table(GP)[, "Phylum"], sum, na.rm = TRUE),
decreasing = TRUE)
top.TaxaGroup <- top.TaxaGroup[top.TaxaGroup > 1*10^6]
# Prune to just the most-abundant phyla
GP <- subset_species(GP, Phylum %in% names(top.TaxaGroup))
# Perform a correspondence analysis
gpca <- ordinate(GP, "CCA")
# # Make species topo with a subset of points layered
# First, make a basic plot of just the species
p1 <- plot_ordination(GP, gpca, "species", color="Phylum")
```

```
# Re-draw this as topo without points, and facet
library("ggplot2")
p1 <- ggplot(p1$data, p1$mapping) + geom_density2d() + facet_wrap(~Phylum)
# Add a layer of a subset of species-points that are furthest from origin.
p53 <- p1 + geom_point(data=subset_ord_plot(p1, 1.0, "square"), size=1)
print(p53)
```

## subset_samples *Subset samples by sample_data expression*

### Description

This is a convenience wrapper around the subset function. It is intended to allow subsetting complex experimental objects with one function call. The subsetting will be based on an expression related to the columns and values within the sample_data.

### Usage

```
subset_samples(physeq, ...)
```

### Arguments

| | |
|---|---|
| physeq | A sample_data-class, or a phyloseq-class object with a sample_data. If the sample_data slot is missing in physeq, then physeq will be returned as-is, and a warning will be printed to screen. |
| ... | The subsetting expression that should be applied to the sample_data. This is passed on to subset, see its documentation for more details. |

### Value

A subsetted object with the same class as physeq.

### See Also

subset_species

### Examples

```
# data(GlobalPatterns)
# subset_samples(GlobalPatterns, SampleType=="Ocean")
```

## subset_taxa — *Subset species by taxonomic expression*

### Description

This is a convenience wrapper around the subset function. It is intended to speed subsetting complex experimental objects with one function call. In the case of subset_taxa, the subsetting will be based on an expression related to the columns and values within the tax_table (taxonomyTable component) slot of physeq.

### Usage

```
subset_taxa(physeq, ...)

subset_species(physeq, ...)
```

### Arguments

| | |
|---|---|
| physeq | A taxonomyTable-class, or phyloseq-class that contains a taxonomyTable. If the tax_table slot is missing in physeq, then physeq will be returned as-is and a warning will be printed to screen. |
| ... | The subsetting expression that should be applied to the taxonomyTable. This is passed on to subset, and more details and examples about how it functions can be found in its documentation. |

### Value

A subsetted object with the same class as physeq.

### See Also

subset_samples

### Examples

```
## ex3 <- subset_taxa(GlobalPatterns, Phylum=="Bacteroidetes")
```

## t — *Transpose otu_table-class or phyloseq-class*

### Description

Extends the base transpose method, t.

### Usage

```
t(x)
```

### Arguments

| | |
|---|---|
| x | An otu_table or phyloseq-class. |

### Value

The class of the object returned by t matches the class of the argument, x. The otu_table is transposed, and taxa_are_rows value is toggled.

### Examples

```
data(GlobalPatterns)
otu_table(GlobalPatterns)
t( otu_table(GlobalPatterns) )
```

---

## taxa_are_rows    *Access taxa_are_rows slot from otu_table objects.*

---

### Description

Access taxa_are_rows slot from otu_table objects.

### Usage

```
taxa_are_rows(physeq)

speciesarerows(physeq)

speciesAreRows(physeq)
```

### Arguments

| | |
|---|---|
| physeq | (Required). phyloseq-class, or otu_table-class. |

### Value

A logical indicating the orientation of the otu_table.

### See Also

otu_table

## taxa_are_rows<- *Manually change taxa_are_rows through assignment.*

### Description

The taxa_are_rows slot is a logical indicating the orientation of the abundance table contained in object x.

### Usage

```
taxa_are_rows(x) <- value
```

### Arguments

| | |
|---|---|
| x | otu_table-class or phyloseq-class |
| value | A logical of length equal to 1. If length(value) > 1, the additional elements will be ignored. Only the first element is assigned to the taxa_are_rows slot. |

### Examples

```
#
# data(GlobalPatterns)
# taxa_are_rows(GlobalPatterns)
# taxa_are_rows(otu_table(GlobalPatterns))
```

## taxa_names *Get species / taxa names.*

### Description

Get species / taxa names.

### Usage

```
taxa_names(physeq)

species.names(physeq)
```

### Arguments

| | |
|---|---|
| physeq | phyloseq-class, otu_table-class, taxonomyTable-class, or phylo |

### Value

A character vector of the names of the species in physeq.

### See Also

ntaxa

## Examples

```
#
# # From "picante" package
# data("phylocom")
# tree <- phylocom$phylo
# OTU1 <- otu_table(phylocom$sample, taxa_are_rows=FALSE)
# taxa_names(tree)
# taxa_names(OTU1)
# physeq1 <- phyloseq(OTU1, tree)
# taxa_names(physeq1)
```

---

## taxa_sums *Returns the total number of individuals observed from each species/taxa/OTU.*

---

### Description

A convenience function equivalent to rowSums or colSums, but where the orientation of the otu_table is automatically handled.

### Usage

```
taxa_sums(x)

speciesSums(x)
```

### Arguments

| | |
|---|---|
| x | otu_table-class, or phyloseq-class. |

### Value

A numeric-class with length equal to the number of species in the table, name indicated the taxa ID, and value equal to the sum of all individuals observed for each taxa in x.

### See Also

sample_sums, rowSums, colSums

### Examples

```
data(enterotype)
taxa_sums(enterotype)
data(esophagus)
taxa_sums(esophagus)
```

## taxonomyTable-class *An S4 class that holds taxonomic classification data as a character matrix.*

### Description

Row indices represent taxa, columns represent taxonomic classifiers.

### Details

**.Data** This slot is inherited from the matrix class.

## tax_glom *Agglomerate taxa of the same type.*

### Description

This method merges species that have the same taxonomy at a certain taxaonomic rank. Its approach is analogous to tip_glom, but uses categorical data instead of a tree. In principal, other categorical data known for all taxa could also be used in place of taxonomy, but for the moment, this must be stored in the taxonomyTable of the data. Also, columns/ranks to the right of the rank chosen to use for agglomeration will be replaced with NA, because they should be meaningless following agglomeration.

### Usage

```
tax_glom(physeq, taxrank=rank_names(physeq)[1],
NArm=TRUE, bad_empty=c(NA, "", " ", "\t"))

taxglom(physeq, taxrank = rank_names(physeq)[1], NArm =
TRUE, bad_empty = c(NA, "", " ", "\t"))
```

### Arguments

| | |
|---|---|
| physeq | (Required). phyloseq-class or otu_table. |
| taxrank | A character string specifying the taxonomic level that you want to agglomerate over. Should be among the results of rank_names(physeq). The default value is rank_names(physeq)[1], which may agglomerate too broadly for a given experiment. You are strongly encouraged to try different values for this argument. |
| NArm | (Optional). Logical, length equal to one. Default is TRUE. CAUTION. The decision to prune (or not) taxa for which you lack categorical data could have a large effect on downstream analysis. You may want to re-compute your analysis under both conditions, or at least think carefully about what the effect might be and the reasons explaining the absence of information for certain taxa. In the case of taxonomy, it is often a result of imprecision in taxonomic designation based on short phylogenetic sequences and a patchy system of nomenclature. If this seems to be an issue for your analysis, think about also trying the nomenclature-agnostic tip_glom method if you have a phylogenetic tree available. |

| | |
|---|---|
| bad_empty | (Optional). Character vector. Default: c(NA, "", " ", "\t"). Defines the bad/empty values that should be ignored and/or considered unknown. They will be removed from the internal agglomeration vector derived from the argument to tax, and therefore agglomeration will not combine taxa according to the presence of these values in tax. Furthermore, the corresponding taxa can be optionally pruned from the output if NArm is set to TRUE. |

## Value

A taxonomically-agglomerated, optionally-pruned, object with class matching the class of physeq.

## See Also

tip_glom

prune_taxa

merge_taxa

## Examples

```
# data(GlobalPatterns)
# ## print the available taxonomic ranks
# colnames(tax_table(GlobalPatterns))
# ## agglomerate at the Family taxonomic rank
# (x1 <- tax_glom(GlobalPatterns, taxrank="Family") )
# ## How many taxa before/after agglomeration?
# ntaxa(GlobalPatterns); ntaxa(x1)
# ## Look at enterotype dataset...
# data(enterotype)
# ## print the available taxonomic ranks. Shows only 1 rank available, not useful for tax_glom
# colnames(tax_table(enterotype))
```

---

tax_table *Build or access the taxonomyTable.*

---

## Description

This is the suggested method for both constructing and accessing a table of taxonomic names, organized with ranks as columns (taxonomyTable-class). When the argument is a character matrix, tax_table() will create and return a taxonomyTable-class object. In this case, the rows should be named to match the species.names of the other objects to which it will ultimately be paired. Alternatively, if the first argument is an experiment-level (phyloseq-class) object, then the corresponding taxonomyTable is returned. Like other accessors (see See Also, below), the default behavior of this method is to stop with an error if object is a phyloseq-class but does not contain a taxonomyTable.

## Usage

```
tax_table(object, errorIfNULL=TRUE)

taxtab(object, errorIfNULL = TRUE)

taxTab(object, errorIfNULL = TRUE)
```

### Arguments

| | |
|---|---|
| object | An object among the set of classes defined by the phyloseq package that contain taxonomyTable. |
| errorIfNULL | (Optional). Logical. Should the accessor stop with an error if the slot is empty (NULL)? Default TRUE. |

### Value

A taxonomyTable object. It is either grabbed from the relevant slot if `object` is complex, or built anew if `object` is a character matrix representing the taxonomic classification of species in the experiment.

### See Also

phy_tree, sample_data, otu_table phyloseq, merge_phyloseq

### Examples

```
#
# tax1 <- tax_table(matrix("abc", 30, 8))
# data(GlobalPatterns)
# tax_table(GlobalPatterns)
```

---

## tax_table<- *Assign a (new) Taxonomy Table to* x

---

### Description

Assign a (new) Taxonomy Table to x

### Usage

```
tax_table(x) <- value
```

### Arguments

| | |
|---|---|
| x | (Required). phyloseq-class |
| value | (Required). taxonomyTable-class. Alternatively, value can be a phyloseq-class that has a tax_table component, or a matrix-class that can be coerced to a taxonomyTable-class with row indices that match at least some of the taxa_names of x. |

### Examples

```
#
# data(GlobalPatterns)
# # An example of pruning to just the first 100 taxa in GlobalPatterns.
# ex2a <- prune_species(taxa_names(GlobalPatterns)[1:100], GlobalPatterns)
# # The following 3 lines produces an ex2b that is equal to ex2a
# ex2b <- GlobalPatterns
# TT <- tax_table(GlobalPatterns)[1:100, ]
# tax_table(ex2b) <- TT
# identical(ex2a, ex2b)
# print(ex2b)
# # 2 examples adding a tax_table component from phyloseq or matrix classes
# ex2c <- phyloseq(otu_table(ex2b), sample_data(ex2b), phy_tree(ex2b))
# tax_table(ex2c) <- ex2b
# identical(ex2a, ex2c)
# ex2c <- phyloseq(otu_table(ex2b), sample_data(ex2b), phy_tree(ex2b))
# tax_table(ex2c) <- as(tax_table(ex2b), "matrix")
# identical(ex2a, ex2c)
```

---

## threshrank — *Thresholded rank transformation.*

---

### Description

The lowest thresh values in x all get the value 'thresh'.

### Usage

```
threshrank(x, thresh, keep0s=FALSE, ...)
```

### Arguments

| | |
|---|---|
| x | (Required). Numeric vector to transform. |
| thresh | A single numeric value giving the threshold. |
| keep0s | A logical determining whether 0's in x should remain a zero-value in the output. If FALSE, zeros are treated as any other value. |
| ... | Further arguments passes to the rank function. |

### Value

A ranked, (optionally) thresholded numeric vector with length equal to x. Default arguments to rank are used, unless provided as additional arguments.

### See Also

transform_sample_counts, rank, threshrankfun

### Examples

```
#
(a_vector <- sample(0:10, 100, TRUE))
threshrank(a_vector, 5, keep0s=TRUE)
data(GlobalPatterns)
GP <- GlobalPatterns
## These three approaches result in identical otu_table
(x1 <- transform_sample_counts( otu_table(GP), threshrankfun(500)) )
(x2 <- otu_table(apply(otu_table(GP), 2, threshrankfun(500)), taxa_are_rows(GP)) )
identical(x1, x2)
(x3 <- otu_table(apply(otu_table(GP), 2, threshrank, thresh=500), taxa_are_rows(GP)) )
identical(x1, x3)
```

## threshrankfun *A closure version of the* threshrank *function.*

### Description

Takes the same arguments as threshrank, except for x, because the output is a single-argument function rather than a rank-transformed numeric. This is useful for higher-order functions that require a single-argument function as input, like transform_sample_counts.

### Usage

```
threshrankfun(thresh, keep0s=FALSE, ...)
```

### Arguments

| | |
|---|---|
| thresh | A single numeric value giving the threshold. |
| keep0s | A logical determining whether 0's in x should remain a zero-value in the output. If FALSE, zeros are treated as any other value. |
| ... | Further arguments passes to the rank function. |

### Value

A single-argument function with the options to threshrank set.

### See Also

transform_sample_counts, threshrankfun, threshrank

### Examples

```
data(GlobalPatterns)
GP <- GlobalPatterns
## These three approaches result in identical otu_table
(x1 <- transform_sample_counts( otu_table(GP), threshrankfun(500)) )
(x2 <- otu_table(apply(otu_table(GP), 2, threshrankfun(500)), taxa_are_rows(GP)) )
identical(x1, x2)
(x3 <- otu_table(apply(otu_table(GP), 2, threshrank, thresh=500), taxa_are_rows(GP)) )
identical(x1, x3)
```

## tip_glom *Agglomerate closely-related taxa using single-linkage clustering.*

### Description

All tips of the tree separated by a cophenetic distance smaller than speciationMinLength will be agglomerated into one taxa using merge_taxa.

## Usage

```
tip_glom(tree, OTU, speciationMinLength=0.02)

tipglom(tree, OTU, speciationMinLength = 0.02)
```

## Arguments

| | |
|---|---|
| tree | phyloseq-class, containing an OTU Table and phylogenetic tree. If, alternatively, tree is a phylo-class, then OTU is required. |
| OTU | An otu_table object. Optional. Ignored if tree is a phyloseq-class object. If tree is a phylo object and OTU is provided, then return will be an phyloseq object. |
| speciationMinLength | The minimum branch length that separates taxa. All tips of the tree separated by a cophenetic distance smaller than speciationMinLength will be agglomerated. Default is 0.02 |

## Details

Can be used to create a non-trivial OTU Table, if a phylogenetic tree is available.

For now, a simple, "greedy", single-linkage clustering is used. In future releases it should be possible to specify different clustering approaches available in R, in particular, complete-linkage clustering appears to be used more commonly for OTU clustering applications.

## Value

An object of class phyloseq. Output class matches the class of tree, unless it is a phylo object, in which case tip_glom returns an phyloseq object.

## Examples

```
#
# # # data(phylocom)
# # # otu  <- otu_table(phylocom$sample, taxa_are_rows=FALSE)
# # # x1   <- phyloseq(otu, phylocom$phylo)
# # # print(x1); par(mfrow=c(2, 1)); plot(phy_tree(x1))
# # # x2 <- tip_glom(x1, speciationMinLength = 2.5)
# # # plot(phy_tree(x2))
# # # ## Try on example datset 1
# # # data(GlobalPatterns); ntaxa(GlobalPatterns)
# # # ex7 <- tip_glom(GlobalPatterns, speciationMinLength = 0.05)
# # # ntaxa(ex7)
# data(esophagus); ntaxa(esophagus); par(mfrow=c(2, 1)); plot(phy_tree(esophagus))
# phy_tree(esophagus)$edge.length
# x3 <- tip_glom(esophagus, speciationMinLength = 0.20)
# ntaxa(x3); plot(phy_tree(x3))
```

topf *Make filter fun. that returns the top f fraction of taxa in a sample.*

### Description

As opposed to topp, which gives the most abundant p fraction of observed taxa (richness, instead of cumulative abundance. Said another way, topf ensures a certain fraction of the total sequences are retained, while topp ensures that a certain fraction of taxa/species/OTUs are retained.

### Usage

```
topf(f, na.rm=TRUE)
```

### Arguments

| | |
|---|---|
| f | Single numeric value between 0 and 1. |
| na.rm | Logical. Should we remove NA values. Default TRUE. |

### Value

A function (enclosure), suitable for filterfun_sample, that will return TRUE for each element in the taxa comprising the most abundant f fraction of individuals.

### See Also

topk, topf, topp, rm_outlierf

### Examples

```
# t1 <- 1:10; names(t1)<-paste("t", 1:10, sep="")
# topf(0.6)(t1)
## Use simulated abundance matrix
# set.seed(711)
# testOTU <- otu_table(matrix(sample(1:50, 25, replace=TRUE), 5, 5), taxa_are_rows=FALSE)
# f1  <- filterfun_sample(topf(0.4))
# (wh1 <- genefilter_sample(testOTU, f1, A=1))
# wh2 <- c(T, T, T, F, F)
# prune_taxa(wh1, testOTU)
# prune_taxa(wh2, testOTU)
```

topk *Make filter fun. the most abundant* k *taxa*

### Description

Make filter fun. the most abundant k taxa

### Usage

```
topk(k, na.rm=TRUE)
```

### Arguments

| | |
|---|---|
| k | An integer, indicating how many of the most abundant taxa should be kept. |
| na.rm | A logical. Should NAs be removed. Default is TRUE. |

### Value

Returns a function (enclosure) that will return TRUE for each element in the most abundant k values.

### See Also

topk, topf, topp, rm_outlierf

### Examples

```
## Use simulated abundance matrix
# set.seed(711)
# testOTU <- otu_table(matrix(sample(1:50, 25, replace=TRUE), 5, 5), taxa_are_rows=FALSE)
# f1  <- filterfun_sample(topk(2))
# wh1 <- genefilter_sample(testOTU, f1, A=2)
# wh2 <- c(T, T, T, F, F)
# prune_taxa(wh1, testOTU)
# prune_taxa(wh2, testOTU)
```

---

## topp — *Make filter fun. that returns the most abundant* p *fraction of taxa*

---

### Description

Make filter fun. that returns the most abundant p fraction of taxa

### Usage

```
topp(p, na.rm=TRUE)
```

### Arguments

| | |
|---|---|
| p | A numeric of length 1, indicating what fraction of the most abundant taxa should be kept. |
| na.rm | A logical. Should NAs be removed. Default is TRUE. |

### Value

A function (enclosure), suitable for filterfun_sample, that will return TRUE for each element in the most abundant p fraction of taxa.

### See Also

topk, topf, topp, rm_outlierf

## Examples

```
## Use simulated abundance matrix
# set.seed(711)
# testOTU <- otu_table(matrix(sample(1:50, 25, replace=TRUE), 5, 5), taxa_are_rows=FALSE)
# sample_sums(testOTU)
# f1  <- filterfun_sample(topp(0.2))
# (wh1 <- genefilter_sample(testOTU, f1, A=1))
# wh2 <- c(T, T, T, F, F)
# prune_taxa(wh1, testOTU)
# prune_taxa(wh2, testOTU)
```

---

## transform_sample_counts

*Transform the abundance count data in an* otu_table*, sample-by-sample.*

---

### Description

This function transforms the sample counts of a taxa abundance matrix according to a user-provided function. The counts of each sample will be transformed individually. No sample-sample interaction/comparison is possible by this method.

### Usage

```
transform_sample_counts(physeq, fun)

transformSampleCounts(physeq, fun)
```

### Arguments

| | |
|---|---|
| physeq | (Required). phyloseq-class of otu_table-class. |
| fun | (Required). A single-argument function that will be applied to the abundance counts of each sample. Can be an anonymous function. |

### Value

A transformed otu_table – or phyloseq object with its transformed otu_table. In general, trimming is not expected by this method, so it is suggested that the user provide only functions that return a full-length vector. Filtering/trimming can follow, for which the genefilter_sample and prune_taxa functions are suggested.

### See Also

threshrankfun, rank, log

## Examples

```
#
data(GlobalPatterns)
GP <- GlobalPatterns
## transform_sample_counts can work on phyloseq-class, modifying otu_table only
(GPr <- transform_sample_counts(GP, rank) )
## These two approaches result in identical otu_table
(x1 <- transform_sample_counts( otu_table(GP), threshrankfun(500)) )
(x2 <- otu_table(apply(otu_table(GP), 2, threshrankfun(500)), taxa_are_rows(GP)) )
identical(x1, x2)
```

---

UniFrac *Calculate weighted or unweighted (Fast) UniFrac distance for all sample pairs.*

---

## Description

This function calculates the (Fast) UniFrac distance for all sample-pairs in a phyloseq-class object.

## Usage

```
UniFrac(physeq, weighted=FALSE, normalized=TRUE,
parallel=FALSE, fast=TRUE)
```

## Arguments

| | |
|---|---|
| physeq | (Required). phyloseq-class, containing at minimum a phylogenetic tree (phylo-class) and contingency table (otu_table-class). See examples below for coercions that might be necessary. |
| weighted | (Optional). Logical. Should use weighted-UniFrac calculation? Weighted-UniFrac takes into account the relative abundance of species/taxa shared between samples, whereas unweighted-UniFrac only considers presence/absence. Default is FALSE, meaning the unweighted-UniFrac distance is calculated for all pairs of samples. |
| normalized | (Optional). Logical. Should the output be normalized such that values range from 0 to 1 independent of branch length values? Default is TRUE. Note that (unweighted) UniFrac is always normalized by total branch-length, and so this value is ignored when weighted == FALSE. |
| parallel | (Optional). Logical. Should execute calculation in parallel, using multiple CPU cores simultaneously? This can dramatically hasten the computation time for this function. However, it also requires that the user has registered a parallel "backend" prior to calling this function. Default is FALSE. If FALSE, UniFrac will register a serial backend so that foreach::%dopar% does not throw a warning. |
| fast | (Optional). Logical. Do you want to use the "Fast UniFrac" algorithm? Implemented natively in the phyloseq-package. This is the default and the recommended option. There should be no difference in the output between the two algorithms. Moreover, the original UniFrac algorithm only outperforms this implementation of fast-UniFrac if the datasets are so small (approximated by the value of ntaxa(physeq) * nsamples(physeq)) that the difference in time |

is inconsequential (less than 1 second). In practice it does not appear that this parameter should ever be set to FALSE, but the option is nevertheless included in the package for comparisons and instructional purposes.

## Details

UniFrac() accesses the abundance (otu_table-class) and a phylogenetic tree (phylo-class) data within an experiment-level (phyloseq-class) object. If the tree and contingency table are separate objects, suggested solution is to combine them into an experiment-level class using the phyloseq function. For example, the following code

phyloseq(myotu_table, myTree)

returns a phyloseq-class object that has been pruned and comprises the minimum arguments necessary for UniFrac().

Parallelization is possible for UniFrac calculated with the phyloseq-package, and is encouraged in the instances of large trees, many samples, or both. Parallelization has been implemented via the foreach-package. This means that parallel calls need to be preceded by 2 or more commands that register the parallel "backend". This is acheived via your choice of helper packages. One of the simplest seems to be the *doParallel* package.

For more information, see the following links on registering the "backend":

*foreach* package manual:

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

Notes on parallel computing in R. Skip to the section describing the *foreach Framework*. It gives off-the-shelf examples for registering a parallel backend using the *doMC*, *doSNOW*, or *doMPI* packages:

http://trg.apbionet.org/euasiagrid/docs/parallelR.notes.pdf

Furthermore, as of R version 2.14.0 and higher, a parallel package is included as part of the core installation, parallel-package, and this can be used as the parallel backend with the foreach-package using the adaptor package "doParallel". http://cran.r-project.org/web/packages/doParallel/index.html

See the vignette for some simple examples for using doParallel. http://cran.r-project.org/web/packages/doParallel/vignettes/gettingstartedParallel.pdf

UniFrac-specific examples for doParallel are provided in the example code below.

## Value

a sample-by-sample distance matrix, suitable for NMDS, etc.

## References

http://bmf.colorado.edu/unifrac/

The main implementation (Fast UniFrac) is adapted from the algorithm's description in:

Hamady, Lozupone, and Knight, "Fast UniFrac: facilitating high-throughput phylogenetic analyses of microbial communities including analysis of pyrosequencing and PhyloChip data." The ISME Journal (2010) 4, 17–27.

http://www.nature.com/ismej/journal/v4/n1/full/ismej200997a.html

See also additional descriptions of UniFrac in the following articles:

Lozupone, Hamady and Knight, "UniFrac - An Online Tool for Comparing Microbial Community Diversity in a Phylogenetic Context.", BMC Bioinformatics 2006, 7:371

Lozupone, Hamady, Kelley and Knight, “Quantitative and qualitative (beta) diversity measures lead to different insights into factors that structure microbial communities.” Appl Environ Microbiol. 2007

Lozupone C, Knight R. “UniFrac: a new phylogenetic method for comparing microbial communities.” Appl Environ Microbiol. 2005 71 (12):8228-35.

### See Also

distance, unifrac

### Examples

```
# ##################################################################
# # Perform UniFrac on esophagus data
# ##################################################################
# data("esophagus")
# (y <- UniFrac(esophagus, TRUE))
# UniFrac(esophagus, TRUE, FALSE)
# UniFrac(esophagus, FALSE)
# picante::unifrac(as(t(otu_table(esophagus)), "matrix"), tre(esophagus))
# ##################################################################
# # Try phylocom example data from picante package
# # It comes as a list, so you must construct the phyloseq object first.
# ##################################################################
# data("phylocom")
# (x1 <- phyloseq(otu_table(phylocom$sample, FALSE), phylocom$phylo))
# UniFrac(x1, TRUE)
# UniFrac(x1, TRUE, FALSE)
# UniFrac(x1, FALSE)
# picante::unifrac(phylocom$sample, phylocom$phylo)
# ##################################################################
# # Now try a parallel implementation using doParallel, which leverages the
# # new 'parallel' core package in R 2.14.0+
# # Note that simply loading the 'doParallel' package is not enough, you must
# # call a function that registers the backend. In general, this is pretty easy
# # with the 'doParallel package' (or one of the alternative 'do*' packages)
# #
# # Also note that the esophagus example has only 3 samples, and a relatively small
# # tree. This is fast to calculate even sequentially and does not warrant
# # parallelized computation, but provides a good quick example for using UniFrac()
# # in a parallel fashion. The number of cores you should specify during the
# # backend registration, using registerDoParallel(), depends on your system and
# # needs. 3 is chosen here for convenience. If your system has only 2 cores, this
# # will probably fault or run slower than necessary.
# ##################################################################
# library(doParallel)
# data(esophagus)
# # For SNOW-like functionality (works on Windows):
# cl <- makeCluster(3)
# registerDoParallel(cl)
# UniFrac(esophagus, TRUE)
# # Force to sequential backed:
# registerDoSEQ()
# # For multicore-like functionality (will probably not work on windows),
# # register the backend like this:
# registerDoParallel(cores=3)
# UniFrac(esophagus, TRUE)
################################################################
```

## [ *Extract parts of otu_table*

### Description

Extract parts of otu_table

extract parts of sample_data

extract parts of taxonomyTable

Generic extraction from higher-order object

# Index