Package: QsRutils (via r-universe)

November 18, 2026

Type Package

Title R Functions Useful for Community Ecology

Version 0.3.0

Description A collection of utility functions for community ecology analyses, with emphasis on workflows using the 'phyloseq' and 'vegan' packages. Includes functions for normalizing OTU tables, computing alpha diversity via rarefaction (using a fast C++ implementation), differential abundance comparisons with compact letter displays, primer checking for amplicon sequencing, plotting QIIME 2/DADA2 generated transition stats and miscellaneous helpers for ordination plots and taxonomic name formatting.

Depends R (>= 4.1.0)

Imports agricolae, ape, Biostrings, dada2, data.table, dplyr, ggplot2, insect, multcompView, magrittr, phyloseq, Rcpp, readr, scales, ShortRead, SRS, stringr, tibble, vegan (>= 2.4-6)

LinkingTo Rcpp

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Roxygen list(markdown = TRUE)

Suggests dunn.test, knitr, rmarkdown, reshape2, testthat (>= 3.0.0)

VignetteBuilder knitr

URL https://github.com/jfq3/QsRutils

BugReports https://github.com/jfq3/QsRutils/issues

Config/testthat/edition 3

Additional_repositories https://bioconductor.org/packages/release/bioc

Config/pak/sysreqs cmake libglpk-dev make libbz2-dev libicu-dev libjpeg-dev liblzma-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libx11-dev xz-utils zlib1g-dev

Repository https://jfq3.r-universe.dev

Date/Publication 2026-10-19 01:26:53 UTC

RemoteUrl https://github.com/jfq3/qsrutils

RemoteRef HEAD

RemoteSha 58fdb7f615fb913d3695ae6e962a0420fc8a56e3

Contents

%wo% . . . 3
arc_sine . . . 4
asterix . . . 4
avg_alpha . . . 5
avg_dist . . . 6
check_primer_hits . . . 8
check_var . . . 9
cld_dunn . . . 10
cld_hsd . . . 11
clear_warnings . . . 11
comb . . . 12
comp_assemble . . . 13
comp_comparisons . . . 14
comp_make_f_tests . . . 15
comp_means_sd . . . 16
comp_prepare_otu_table . . . 17
comp_prepare_phyloseq . . . 18
deg2rad . . . 19
expt . . . 19
extract_adapter_content . . . 20
find_truncation_position . . . 20
format_taxon . . . 21
generate_password . . . 22
get_groups . . . 23
get_plot_limits . . . 24
goods . . . 24
hash_dna_seqs . . . 25
its.root . . . 26
log_arc_sine . . . 26
make_comparisons . . . 27
make_letter_assignments . . . 28
merge_2_frames . . . 29
ord_labels . . . 30
pca_labels . . . 31
perm . . . 32
plot_df . . . 33
plot_transition_stats . . . 33
prop_filter . . . 34

QsRutils 34
rad2deg 35
rda_labels 35
root_phyloseq_tree 36
se 37
sqrt_arc_sine 38
srs_p 38
subset_by_refseq_lengths 39
subset_dist 40
vegan_stand 41
veganotu 42
vegansam 43

Index **44**

`%wo%` *Remove elements from a vector*

Description

Removes elements present in `y` from `x`.

Usage

```
x %wo% y
```

Arguments

`x`	A vector.
`y`	A vector of elements to remove from `x`.

Value

A vector containing elements of `x` that are not in `y`.

Examples

```
c(1, 2, 3, 4, 5) %wo% c(2, 4)
letters[1:5] %wo% c("b", "d")
```

arc_sine *arc_sine*

Description

Transform a percentage to its arc sine.

Usage

```
arc_sine(x)
```

Arguments

x	A percentage.

Value

The arcsine transformation of x.

Examples

```
arc_sine(30.1)
```

asterix *Indicate Significance with Stars*

Description

Indicate Significance with Stars

Usage

```
asterix(prob)
```

Arguments

prob	p value

Details

Returns '**; for p < 0.001, **' *for p < 0.01, "* for p < 0.05.

Value

Character vector of asterisks indicating significance level.

Examples

```
asterix(0.039)
```

---

avg_alpha *Average Alpha Diversity (faster implementation)*

Description

Calculates alpha-diversity metrics from n samplings of an OTU table to a constant number of counts per sample.

Usage

```
avg_alpha(
  otu,
  sampling_depth,
  iterations = 100,
  sum_method = c("median", "mean"),
  ncores = 1
)
```

Arguments

otu	An OTU table as a data frame or matrix with samples as rows and taxa as columns.
sampling_depth	The number of counts per sample in the sampled OTU table
iterations	The number of times the OTU table should be sampled.
sum_method	Method ("median" or "mean") for summarizing replication results.
ncores	Number of cores to use for parallel execution. Default 1 (no parallelism).

Details

This implementation focuses on speed:

- vectorizes the alpha-metric calculations (avoids repeated calls to vegan::diversity and vegan::specnumber)
- uses base R operations (rowSums, logical ops, arithmetic) which are much faster in tight loops
- optionally parallelizes replicates across cores (platform-aware).

The OTU data frame supplied must be in typical vegan format: samples as row names and taxa as column names. The minimum row sum must be greater than or equal to the sampling depth.

By default the sum_method is mean. For a similar function in QIIME2, the default sum_method is median.

Value

Returns a data frame with Shannon, Observed, Pielou, Simpson and Inverse Simpson as the column names for each sample as the row names in an OTU table.

Examples

```
{
data(BCI, package = "vegan")
otu <- BCI[rowSums(BCI) > 400, ]
avg_alpha(otu, sampling_depth = 400, iterations = 100)
}
```

---

avg_dist *Averaged Sub-sampled Dissimilarity Matrices*

Description

The function computes the dissimilarity matrix of a data set multiple times using vegdist while randomly sub-sampling the dataset each time. All of the subsampled iterations are then averaged (mean) to provide a distance matrix that represents the average of multiple subsampling iterations. This emulates the behavior of the distance matrix calculator within the Mothur microbial ecology toolkit.

Usage

```
avg_dist(x, sample, distfun = vegan::vegdist, meanfun = mean,
transf = NULL, iterations = 100, dmethod = "bray",
diag = TRUE, upper = TRUE, ncores = 1L, ...)
```

Arguments

x	Community data matrix
sample	The subsampling depth to be used in each iteration. Samples that do not meet this threshold will be removed from the analysis, and their identity returned to the user in stdout.
distfun	The dissmilarity matrix function to he used. Default is the vegan vegdist
meanfun	The calculation to use for the average (mean or median)
transf	Transformation function to apply to the distance matrix. Default is NULL.
iterations	Number of subsampling iterations. Default is 100.
dmethod	Distance method. Default is "bray".
diag	Whether to include the diagonal of the distance matrix. Default is TRUE.
upper	Whether to include the upper triangle of the distance matrix. Default is TRUE.
ncores	Number of cores to use for parallel computation via makeCluster. Default is 1L (no parallelism). Values greater than 1 spawn a socket cluster, which works on all platforms including Windows.
...	Any additional arguments to add to the distance function of mean/median function specified.

Details

Parallelism. Each iteration is independent, so the work can be distributed across cores with `ncores > 1`. Whether this is worthwhile depends on the cost of each iteration relative to the fixed overhead of spawning a socket cluster (typically 1–2 seconds).

Benchmarks on a Windows machine with 100 samples × 5000 OTUs (Bray-Curtis, `sample = 20000`) gave the following observed speedups relative to `ncores = 1`:

ncores	50 iters	200 iters
1	1.00× (48 s)	1.00× (204 s)
2	1.44× (33 s)	1.48× (138 s)
4	1.91× (25 s)	2.04× (100 s)
8	2.12× (23 s)	2.37× (86 s)

Observed speedups are well below the theoretical maximum due to inter-process communication overhead in socket clusters (the only cluster type available on Windows). The relative benefit of parallelism improves with more iterations because the fixed cluster overhead is amortized across a larger amount of work. For small problems (e.g., <50 samples and `iterations` <= 100 with the default `vegan::vegdist`), the per-iteration cost is low enough that `ncores = 1L` is likely faster.

Value

An averaged distance matrix.

Note

The function builds on the function `rrarefy` and an additional distance matrix function (e.g. `vegdist`) to add more meaningful representations of distances among randomly subsampled datasets by presenting the average of multiple random iterations. This functionality has been utilized in the Mothur standalone microbial ecology toolkit, see https://mothur.org/wiki/Dist.shared.

Author(s)

Geoffrey Hannigan, with some minor tweaks by Gavin L. Simpson and parallelization by John Quensen.

See Also

`vegan::vegdist()`

`vegan::rarefy()`

Examples

```
# Import an example count dataset
data(BCI, package = "vegan")
mean.avg.dist <- avg_dist(BCI, sample = 50, iterations = 5, ncores = 2)
mean.avg.dist
```

`check_primer_hits` *Check Primer Hits*

Description

Determine hits of all orientations of primers to paired sequence files.

Usage

```
check_primer_hits(
  path,
  fwd_pattern = "_R1.fastq",
  rev_pattern = "_R2.fastq",
  fwd_primer = "GGAAGTAAAAGTCGTAACAAGG",
  rev_primer = "GCTGCGTTCTTCATCGATGC"
)
```

Arguments

`path`	Path to paired sequence files
`fwd_pattern`	Portion of file name that distinguishes forward read files. The default is "_R1.fastq"
`rev_pattern`	Portion of the file name that distinguishes the reverse file. The default is "_R2.fastq"
`fwd_primer`	Nucleotide sequence of the forward primer
`rev_primer`	Nucleotide sequence of the reverse primer

Details

This function is for checking the effectiveness of primer trimming of ITS sequences. Because the ITS region varies in length, it is possible for forward sequences to extend past the reverse primenr region and vice-versa, leadsing to what Robert Edgar calls staggered pairs fi the sequences are merged.

The fwd_pattern and rev_pattern must contain the file extension. The defaults are "_R1.fastq" and "_R2.fastq".

Default primers are ITS5 (forward) and ITS2 (reverse) from White et.al 1990.

ITS5: "GGAAGTAAAAGTCGTAACAAGG"

ITS2: "GCTGCGTTCTTCATCGATGC"

Value

A table of hits to the sequences by all primer orientations

Examples

```
{
  # Copy example fastq files to a writable temp directory
  src <- system.file("extdata", package = "QsRutils")
  path <- file.path(tempdir(), "QsRutils_example")
  dir.create(path, showWarnings = FALSE)
  file.copy(list.files(src, pattern = "\\.fastq\\.gz$", full.names = TRUE), path)
  check_primer_hits(
    path = path,
    fwd_pattern = "raw_1.fastq.gz",
    rev_pattern = "raw_2.fastq.gz",
    fwd_primer = "GGAAGTAAAAGTCGTAACAAGG",
    rev_primer = "GCTGCGTTCTTCATCGATGC"
  )
}
```

---

check_var *Check Variance*

Description

Tests for Heterogeneity of Variances in make_comparisons Result

Usage

```
check_var(otu.pc.transformed, group.vector)

## S3 method for class 'check_var'
print(x, ...)
```

Arguments

otu.pc.transformed	An OTU matrix of transformed data. Taxa are rows.
group.vector	A vector of treatments.
x	A check_var object.
...	Arguments passed to print.data.frame.

Value

A data frame of class "check_var" with one row per taxon and columns taxon, statistic, df, and p.value from the Fligner-Killeen test. The object can be printed with print().

See Also

make_comparisons

Examples

```
# Transform species matrix to proportion;
# Check variances for the first three species
# in the dune data set grouped by Management.
data(dune, package = "vegan")
data(dune.env, package = "vegan")
dune <- vegan::decostand(dune, method = "total")
dune <- dune[, 1:3]
dune <- t(dune)
check_var(dune, dune.env$Management)
```

`cld_dunn` *CLDs from DUNN*

Description

Generate compact letter displays from Dunn.test results

Usage

```
cld_dunn(dunn_rslt, significance = 0.05)
```

Arguments

`dunn_rslt`	The result of the function dunn.test::dunn.test
`significance`	The alpha level for statistical significance

Value

A list of two items. The first, p_adj_matrix, is an object of class `dist` giving p values adjusted for multiple comparisons. The second,clds, is a character vector of compact letter displays (CLDs) for each treatment.

Examples

```
# Example cribbed and modified from the kruskal.test documentation
x <- c(2.9, 3.0, 2.5, 2.6, 3.2) # normal subjects
y <- c(3.8, 2.7, 4.0, 2.4)      # with obstructive airway disease
z <- c(2.8, 3.4, 3.7, 2.2, 2.0) # with asbestosis
x <- c(x, y, z)
g <- factor(rep(1:3, c(5, 4, 5)),
            labels = c("Normal",
                       "COPD",
                       "Asbestosis"))
dunn_rslt <- dunn.test::dunn.test(x, g)
cld_dunn(dunn_rslt, significance = 0.05)
```

cld_hsd *Make CLD tibble from Tukey HSD Results*

Description

Makes a tibble for adding compact letter assignments to a boxplot using HSD.test results.

Usage

```
cld_hsd(hsd_rslt, y_pos = "boxtop")
```

Arguments

hsd_rslt	The result of the HSD.test function of package agricolae
y_pos	The y-position in relation to the boxplots. Choices are at the top of the box ("boxtop", the default) or at the maximum group value ("max").

Details

hsd_rslt must be created with agricolae::HSD.test

Value

A tibble with columns for treatment groups (x), the y-positions of the treatment CLD (y), and the CLD letters indicating significantly different treatments.

Examples

```
data("iris")
model <- lm(Petal.Length ~ Species, data = iris)
hsd_rslt <- agricolae::HSD.test(model, trt="Species")
cld_hsd(hsd_rslt)
```

clear_warnings *Clear Warnings*

Description

Clears all warning messages from the base environment.

Usage

```
clear_warnings()
```

Details

Sometimes when working in the console R retains a list of warnings such that they keep being reported after the function call which originated them. This function removes them so that they are not a nuisance.

Value

No return value, called for side effects. Clears the stored warning list so that stale warnings are no longer reported in the console.

Examples

```
{
as.numeric(c("1", "2", "apples"))
summary(warnings())
clear_warnings()
summary(warnings())
}
```

---

comb *comb*

Description

Calculates the number of combinations of n things drawn r at a time.

Usage

```
comb(n, r, repetition = FALSE)
```

Arguments

n	The total number of items.
r	The number of items to be drawn.
repetition	A logical, whether or not repetitions are allowed. FALSE by default.

Value

An integer giving the number of ways a set of r items can be drawn from a set of n items.

Examples

```
comb(5, 3)
comb(5, 3, repetition = TRUE)
```

comp_assemble *Assemble Comparison Parts*

Description

Assembles Comparison Data Frame

Usage

```
comp_assemble(part1, part2, part3)
```

Arguments

part1	Result from comp_means_sd
part2	Result from comp_make_f_tests
part3	Result from comp_comparisons

Details

The data frame returned has taxa as row names. The first three column names are mean (relative abundance of the taxa), sd and F_value for comparisons among groups. The remaining column names are the groups. The group columns show the mean relative abundance for the group plus/minus the standard deviation and a compact letter display for the group. See also the vignette "Compare Relative Abundances Among Treatments."

Value

A summary data frame of differential abundances by taxon and treatment.

Examples

```
{
data("its.root")
temp1 <- comp_prepare_phyloseq(its.root)
temp2 <- comp_prepare_otu_table(temp1$expt.taxon.pc,
                                grps = "Label",
                                transformation = "sqrt_arc_sine")
temp3 <- comp_means_sd(temp2$otu.pc)
temp4 <- comp_make_f_tests(temp2$otu.pc.trans,
                           grps = temp2$groups,
                           var.equal = TRUE)
temp5 <- comp_comparisons(otu.pc = temp2$otu.pc,
                          otu.pc.trans = temp2$otu.pc.trans,
                          grps = temp2$groups,
                          p.adjust.method = "BH",
                          pool.sd = TRUE)
comp_assemble(temp3, temp4, temp5) |>
  dplyr::arrange(desc(mean))
```

```
}
```

`comp_comparisons` *Make Comparisons*

Description

Calculates the treatment comparison portion of a table comparing relative abundances of each taxon among treatments.

Usage

```
comp_comparisons(
  otu.pc,
  otu.pc.trans,
  grps,
  p.adjust.method = "BH",
  pool.sd = FALSE
)
```

Arguments

`otu.pc`	An OTU table of percentages.
`otu.pc.trans`	An OTU table of transfromed data.
`grps`	A vector of treatemnt groups for which to make comparisons.
`p.adjust.method`	Adjustment method for multiple comparisons.
`pool.sd`	A logical, whether or not to pool standard deviations.

Details

The row names of the data frame returned are taxa. The columns are of type character and their names are the group names. For each group, the entry gives the mean relative abundance +/- the standard deviation and a compact letter display (CLD) for the group. See also the vignette "Compare Relative Abundances Among Treatments."

Value

A data frame of differences in relative abundances among treatments.

Examples

```
{
data("its.root")
temp1 <- comp_prepare_phyloseq(its.root)
temp2 <- comp_prepare_otu_table(temp1$expt.taxon.pc,
                                grps = "Label",
                                transformation = "sqrt_arc_sine")
comp_comparisons(otu.pc = temp2$otu.pc,
                 otu.pc.trans = temp2$otu.pc.trans,
                 grps = temp2$groups,
                 p.adjust.method = "BH",
                 pool.sd = TRUE)
}
```

`comp_make_f_tests` *Make F Tests*

Description

Calculates omnibus F tests to be included in a table comparing relative abundances of each taxon among treatments.

Usage

```
comp_make_f_tests(otu.pc.trans, grps, var.equal = FALSE)
```

Arguments

`otu.pc.trans`	An OTU table of transformed data from comp_prepare_otu_table.
`grps`	A vector of treatment groups for which to make comparisons.
`var.equal`	Logical, whether or not to assume variances equal.

Details

The row names of the data frame returned are taxa. The column names are `F`, `Prob`, `sig` and `F_value`. The column `sig` includes asterisks indicating the degree of significance and the `F_value` column is the F value to 2 decimal places plus the asterisk(s) from `sig`. See also the vignette "Compare Relative Abundances Among Treatments."

Value

A data frame of the F-test results.

Examples

```
{
data("its.root")
temp1 <- comp_prepare_phyloseq(its.root)
temp2 <- comp_prepare_otu_table(temp1$expt.taxon.pc,
                                grps = "Label",
                                transformation = "sqrt_arc_sine")
temp4 <- comp_make_f_tests(temp2$otu.pc.trans,
                           grps = temp2$groups,
                           var.equal = TRUE)
temp4
}
```

`comp_means_sd` *Calculate Means and Standard Deviations*

Description

Calculates means and standard deviation for each taxon to be included in a table comparing relative abundances of each taxon among treatments.

Usage

```
comp_means_sd(otu.pc)
```

Arguments

`otu.pc`	An OTU table with data as percentages.

Details

The OTU table should be created with comp_prepare_otu_table. The data frame returned has taxa as row names. For each taxa, `mean` is the mean relative abundance and `sd` is the standard deviation. See also the vignette "Compare Relative Abundances Among Treatments."

Value

A data frame with mean relative abundance and standard deviations by taxon.

Examples

```
{
data("its.root")
temp1 <- comp_prepare_phyloseq(its.root)
temp2 <- comp_prepare_otu_table(temp1$expt.taxon.pc,
                                grps = "Label",
                                transformation = "sqrt_arc_sine")
temp3 <- comp_means_sd(temp2$otu.pc)
```

```
temp3
}
```

`comp_prepare_otu_table` *Prepare OTU Table*

Description

Make OTU tables for making comparisons of relative abundances among treatments.

Usage

```
comp_prepare_otu_table(
  expt.taxon.pc,
  grps = "Treatment",
  transformation = "sqrt_arc_sine"
)
```

Arguments

`expt.taxon.pc`	Phyloseq object from comp_prepare_phyloseq with percentages in the otu_table.
`grps`	Factor in sample data for which to make comparisons.
`transformation`	Transformation function to use.

Details

The transformation applied may be "none" or a user-supplied function name in quotation marks or any of the built-it transformations("arc_sine", "log_arc_sine", or "sqrt_arc_sine"). The "sqrt_arc_sine" has generally proven most effective. See also the vignette "Compare Relative Abundances Among Treatments."

Value

A list consisting of an OTU table with percentages, an OTU table with transformed data, and a vector of treatment groups.

Examples

```
{
data("its.root")
temp1 <- comp_prepare_phyloseq(its.root)
temp2 <- comp_prepare_otu_table(temp1$expt.taxon.pc,
                                grps = "Label",
                                transformation = "sqrt_arc_sine")
temp2
}
```

`comp_prepare_phyloseq` *Prepare Phyloseq*

Description

Prepares a phyloseq object for making comparisons of relative abundances among treatments.

Usage

```
comp_prepare_phyloseq(expt, taxrank = "Phylum", pc.filter = 0.01)
```

Arguments

`expt`	Experiment level phyloseq object.
`taxrank`	Taxonomic rank for which to make comparisons.
`pc.filter`	Minimum percentage of total counts to include rank in result.

Details

For both returned phyloseq objects, the OTU table has been filtered to include only OTUs initially present at >= pc.filter times the original total count and only taxrank is included in the taxonomy table. For the second object in the list, the OTU table has been transformed to percentages of the total counts per sample. See also the vignette "Compare Relative Abundances Among Treatments."

Value

A list of two modified experiment level phyloseq objects.

Examples

```
{
data(its.root)
temp1 <- comp_prepare_phyloseq(its.root)
temp1
}
```

deg2rad *deg2rad*

Description

Transform an angle in degrees to radians.

Usage

```
deg2rad(x)
```

Arguments

x	Angle in degrees

Value

Angle in radians.

Examples

```
deg2rad(90)
```

expt *A 16S Experiment Level phyloseq Object*

Description

Based on 16S rRNA gene sequences from Iowa loess soil

Usage

```
expt
```

Format

A phyloseq object with otu_table, sample_data, tax_table, phylogenetic tree and refseqs.

Depth 0-20 cm, 60-80cm

Site HNC

Slope Top, Middle, Bottom

`extract_adapter_content` *Extract FastQC Adapter Content Table*

Description

Parses a FastQC `.zip` file to locate the "Adapter Content" module and extracts the data as a clean data frame, preserving read positions as ordered categories.

Usage

```
extract_adapter_content(zip_path)
```

Arguments

`zip_path`	A character string specifying the path to the FastQC `.zip` file.

Value

A `data.frame` where the first column (`Position`) is an ordered factor, subsequent columns represent the cumulative percentages of different adapter types, and two trailing columns: `sample_id` (derived from the file name by stripping FastQC suffixes and read-direction tags) and `direction` (`"forward"`, `"reverse"`, or `NA` if the direction cannot be determined from the file name). Returns `NULL` if the section is missing or empty.

`find_truncation_position` *Find Read Truncation Position Based on FastQC Quality and Read Direction*

Description

Parses a FastQC `.zip` file for a specific read direction (forward or reverse), looks for the "Per base sequence quality" section, and identifies the first position where the Lower Quartile dips to or below a specified Q-score threshold.

Usage

```
find_truncation_position(
  zip_path,
  read_direction = c("forward", "reverse"),
  target_q = 20
)
```

Arguments

zip_path	A character string specifying the path to the FastQC .zip file.
read_direction	A character string, either "forward" or "reverse". Matches common patterns like _R1_/_1 for forward and _R2_/_2 for reverse.
target_q	A numeric value specifying the Lower Quartile Q-score threshold. Default is 20.

Details

Results for all files in a path can be aggregated with purrr::list_rbind():

```
all_zips <- list.files(path = "path_to_zip_files", pattern = "\.zip$", full.names = TRUE)
purrr::map(all_zips, find_truncation_position, "forward") |> purrr::list_rbind()
purrr::map(all_zips, find_truncation_position, "reverse") |> purrr::list_rbind()
```

Value

A one-row data frame with columns sample_name (file name without _fastqc.zip), direction ("forward" or "reverse"), and position (quality result string). Returns NULL if the file does not match the requested read direction.

Examples

```
## Not run:
find_truncation_position("sample1_L001_R1_001_fastqc.zip", read_direction = "forward")

## End(Not run)
```

---

format_taxon *Format a taxon*

Description

Formats a taxon so that it will be properly italicized in a ggpplot

Usage

```
format_taxon(x)
```

Arguments

x	A string representing a phylum, class, etc.

Details

If a taxon begins with an upper case letter followed by lower case letters and does not contain an underscore, it is wrapped in asterisks. If it begins with an upper case letter followed by lower case letters and contains an underscore, the portion before the underscore is wrapped in asterisks, the underscore removed and any letters following the underscore left alone. If a taxon contains all upper case letters or digits it is not a proper taxon and is left alone.

Value

The string with asterisks properly inserted.

Examples

```
format_taxon("UAB")
format_taxon("Pseudomonas_B")
format_taxon("Pseudomonas")
```

---

generate_password *Genererate a Password*

Description

Generates a random character string of specified length.

Usage

```
generate_password(n, type = "alpha_numeric")
```

Arguments

n	Numberof characters in password.
type	c("alpha_numeric", "anything_else")

Details

If type equals "alpha_numeric" (the default), only alpha-numeric characters are used to generate the password. If type does not equal "alpha_numeric" then at least one non-alpha-numeric symbol will be included in the password. In either case, the alpha characters used are both upper and lower case.

Value

A character string.

Examples

```
generate_password(8)
```

get_groups *get_groups*

Description

Assign treatment groups based on pairwise t-tests.

Usage

```
get_groups(ptt.rslt, alpha = 0.05, rm.subset = FALSE)
```

Arguments

ptt.rslt	Result from the stats function pairwise.t.test.
alpha	Confidence level.
rm.subset	A logical; remove group subsets if true.

Details

This function aids in making letter assignments as to which treatments are significantly different. Also returns a square matrix of alpha values for all pairwise differences. This square matrix can serve as input to the multcompLetters function of the multcompView package which provides letter assignments. If rm.subset is FALSE, then groups such as {A,B} and {A, B, C} may be reported. This is redundant in the sense the {A, B} is a subset of {A, B, C}. In this case if rm.subset is FALSE, the group {A, B} is not reported.

Value

A list consisting of groups of treatment groups that are not significantly different and a matrix of p values.

See Also

make_letter_assignments

Examples

```
attach(airquality)
Month <- factor(Month, labels = month.abb[5:9])
ptt.rslt <- pairwise.t.test(Ozone, Month)
detach(airquality)
get_groups(ptt.rslt, alpha = 0.05, rm.subset = FALSE)
```

get_plot_limits *Get ggplot Plot Limits*

Description

Gets the ranges for the width and height of a ggplot panel.

Usage

```
get_plot_limits(plot)
```

Arguments

plot	A plot created with ggplot2

Details

Sometimes when adding text to a ggplot, the text is cutoff if it extends beyond the limits of plot panel. This function provides information enabling the user to extend the panel limits so that the text is not cutoff.

Value

A list: xmin, xmax, ymin, ymax giving the coordinates of the limits of a ggplot panel.

Examples

```
library(ggplot2)
data(iris)
plt <- ggplot(data=iris, aes(x=Species, y=Petal.Length)) + geom_boxplot()
get_plot_limits(plt)
```

goods *Calculate Good's Coverage*

Description

Calculates Good's coverage from a community data matrix with samples as rows and OTUs as columns.

Usage

```
goods(com)
```

Arguments

com	a vegan compatible community data matrix.

Value

A table with the headings number of singletons, number of sequences, and Good's coverage for each sample in rows.

References

Good, I. J. 1953. The Population Frequencies of Species and the Estimation of Population Parameters. Biometrika 40:237-264.

Examples

```
{
data(dune, package = "vegan")
goods(dune)
}
```

`hash_dna_seqs` *Hash DNA sequences*

Description

Renames sequences in a multi-fasta file with the MD5 hash of each sequence.

Usage

```
hash_dna_seqs(seqs)
```

Arguments

`seqs`	A list of DNA sequences

Details

Taxa names for the ASV table returned by the R version of DADA2 are the sequences themselves. This function renames them with the MD5 hash of each sequences so that the results are directly comparable to QIIME2/DADA2 results.

Value

DNA sequences renamed with their hashes.

Examples

```
seqs <- ">ASV1
AGTTTGATCATGGCTCAGATTGAACGCTGGCGGCAGGCCTAACACATGCAAGTCGAACGGTAACAGGAAG"
hash_dna_seqs(seqs)
```

its.root *An ITS Experiment Level phyloseq Object*

Description

Based on ITS2 sequences amplified from corn roots.

Usage

```
its.root
```

Format

A phyloseq object with otu_table, sample_data and tax_table. The sample_data variables are:

P Phosporous level, H or L

Genotype One of three: 2, 3, and C

Label A code for treatments: 2HR, 2LR, 3HR, 3LR, CHR, CLR

log_arc_sine *log_arc_sine*

Description

Transform a percentage to the log of its arc sine.

Usage

```
log_arc_sine(x)
```

Arguments

x	A percentage.

Value

The common logarithm of the arcsine transformation of x.

Examples

```
log_arc_sine(x = 30.1)
```

make_comparisons *Make Multiple Comparisons on Transformed Data*

Description

Makes multiple comparisons of the relative abundances of taxa between treatment groups using the pairwise.t.test. Data may be transformed by a user supplied function. Three are included in this package.

Usage

```
make_comparisons(
  expt,
  taxrank = "Phylum",
  grps = "Treatment",
  transformation = "none",
  pc.filter = 0.01,
  p.adjust.method = "BH",
  pool.sd = FALSE
)
```

Arguments

expt	Experiment level phyloseq object.
taxrank	Rank for which to make comparisons.
grps	Factor in sample data for which to make comparisons.
transformation	Transformation function to use.
pc.filter	Minimum percentage of total counts to include rank in result.
p.adjust.method	Adjustment method for multiple comparisons.
pool.sd	Logical, whether or not to pool standard deviations.

Details

This is essentially a wrapper around the functions comp_prepare_phyloseq(), comp_prepare_otu_table(), comp_means_sd(), comp_make_f_tests(), comp_comparisons() and comp_assemble(). Transformation may be "none" or a user-supplied function name in quotation marks or any of the built-it transformations ("arc_sine", "log_arc_sine", or "sqrt_arc_sine"). The "sqrt_arc_sine" has generally proven most effective.

Value

A list consisting of three data frames and a vector of treatment groups. The first data frame is comparison.table.giving for each row (taxon) the mean relative abundance, standard deviation, F statistic with significance indicated by asterisks, and then for each treatment group the mean+/-sd with a CLD indicating group membership. The second data frame is taxa.pc giving for each row

(taxon) the relative abundance of each treatment group. The third data frame is taxa.pc.transformed. It is like the second data frame but the data has been transformed using the specified function.

See Also

arc_sine, log_arc_sine, sqrt_arc_sine, check_var

Examples

```
{
data(its.root)
make_comparisons(its.root,
taxrank = "Phylum",
grps = "Label",
transformation = "sqrt_arc_sine",
pc.filter = 0.01,
p.adjust.method ="BH",
pool.sd = TRUE)
}
```

`make_letter_assignments` *Make Letter Assignments*

Description

Makes letter assignments for treatment groups that are not significantly different.

Usage

```
make_letter_assignments(ptt.rslt, significance = 0.05)
```

Arguments

`ptt.rslt`	Output from the pairwise.t.test function.
`significance`	Alpha level to be declared a significant difference.

Details

Letter assignments are made using Piepho's algorithm.

Value

A named vector of letter assignments. Names are treatment groups.

References

Piepho, H. P. 2004. An algorithm for a letter-based representation of all-pairwise comparisons. Journal of Computational and Graphical Statistics **13**:456-466.

Examples

```
{
data(iris, package = "datasets")
ptt.rslt <- with(iris, pairwise.t.test(Petal.Length, Species, pool.sd = FALSE))
make_letter_assignments(ptt.rslt, significance = 0.05)
}
```

`merge_2_frames` *Merge Two Data Frames*

Description

Merge two data frames by their row names.

Usage

```
merge_2_frames(one, two)
```

Arguments

one	A data frame.
two	A second data frame.

Details

Merges data frames by common row names. This function differs from merge.data.frames in that the merged data frame returned has row names and not a new column of the row names.

Value

A merged data frame.

Examples

```
{
common_rows <- paste0("ID_", 1:5)

df1 <- data.frame(
  Value_A = runif(5),
  Category = sample(c("X", "Y"), 5, replace = TRUE),
  row.names = common_rows
)

df2 <- data.frame(
  Value_B = rnorm(5),
  Flag = sample(c(TRUE, FALSE), 5, replace = TRUE),
  row.names = common_rows
)
```

```
merge_2_frames(df1, df2)
}
```

`ord_labels` *Make Ordination Axis Labels*

Description

Makes ordination axis labels that include, if appropriate, the % of the total variance explained by each axis.

Usage

```
ord_labels(ord)
```

Arguments

`ord`	A vegan ordination object.

Details

If there are no eigenvalues in ord, or if any eigenvalues are less than 0, each element of the vector returned has the form "DIMn" where n is the axis number. Otherwise, each element of the vector returned has the form Axisn xx.x % where Axis is taken from the vector of eigenvalues in ord if they are named or simply "DIM" if they are not, n is the number of the axis, and xx.x is the % of total variance explained by the axis.

For this function to work correctly, ord should be created in one of the following ways:

1. As an unconstrained ordination using vegan::rda. In this case the labels have the form PCAn xx.x %
2. As a PCoA made with stats::cmdscale. In this case the labels have the form DIMn xx.x %.
3. As a CA made with vegan::ca. In this case the labels have the form CAn xx.x %.

Value

A character vector, each element of which can be used to label the corresponding axis of an ordination plot.

See Also

`rda_labels()`

Examples

```
{
# For PCA using rda:
data("dune", package = "vegan")
dune_hel <- vegan::decostand(dune, method = "hellinger")
pca <- vegan::rda(dune_hel)
print("For the PCA case:")
print(ord_labels(pca)[1:2])
cat("\n")
# For PCoA with negative eigenvalues
d <- vegan::vegdist(dune)
pcoa <- stats::cmdscale(d, k = nrow(dune)-1, eig = TRUE, add = FALSE)
print("For the PCoA case with negative eigenvalues:")
print(ord_labels(pcoa)[1:2])
cat("\n")
# For PCoA without negative eigenvalues
pcoa <- stats::cmdscale(d, k = nrow(dune)-1, eig = TRUE, add = TRUE)
print("For the PCoA case without negative eigenvalues:")
print(ord_labels(pcoa)[1:2])
cat("\n")
# For correspondence analysis
ca_ord <- vegan::ca(dune)
(ord_labels(ca_ord))
print("For the CA case:")
print(ord_labels(ca_ord))
}
```

---

pca_labels *Make PCA Axis Labels*

Description

Makes PCA axis labels that include the % total variance explained by each axis.

Usage

```
pca_labels(pca)
```

Arguments

pca	Object containing the results of vegan's rda function.

Details

Each element of the vector returned has the form "PCAn xx.x%" where n is the number of the PCA axis and xx.x is the % of total variance explained by the axis.

Value

A character vector, each element of which can be used to label the corresponding axis of a PCA plot.

Examples

```
{
data(dune, package = "vegan")
dune_hel <- vegan::decostand(dune, method = "hellinger")
pca_ord <- vegan::rda(dune_hel)
pca_labels(pca_ord)
}
```

---

perm *Permutations*

Description

Retuns the number of permutaions of n things taken r at a time.

Usage

```
perm(n, r, repetition = FALSE)
```

Arguments

n	Total number of items.
r	Number of items drawn.
repetition	A logical, whether or not repetitions are allowed. FALSE by default.

Value

An integer giving how many ways m things can be drawn n at a time.

Examples

```
perm(10, 5)
perm(10, 5, repetition = TRUE)
```

plot_df *A Data File in Long Format*

Description

Used in Case 3 of the vignette make_comparisons

Usage

```
plot_df
```

Format

A data file in long format used for a ggplot. The sample_data variables are:

Treatment A code for genotype (2, 3, or C), P level (H or L) and sample type (R)

Family One of the families in Gigasporaceae

Percent Percent of total counts for family and treatment combination.

plot_transition_stats *Plot DADA2 Transition Stats*

Description

Extracts a QIIME2/DADA2 transition stats file and returns a plot.

Usage

```
plot_transition_stats(trans_stats.qza)
```

Arguments

`trans_stats.qza`
: The transitions stats file output by QIIME2 DADA2

Details

Beginning with QIIME2 version 2025.7 the DADA2 plugin requires output of a compressed (qza) file of the transition stats. This function makes a ggplot plot from the data in that file. It is useful in determining how well DADA2 has corrected read errors.

Value

A ggplot of the transition probabilities

Examples

```
qza <- system.file("extdata", "base-transition-stats.qza", package = "QsRutils")
plot_transition_stats(qza)
```

`prop_filter` *Filter OTUs by Abundance*

Description

Allows subsetting of a phyloseq object according to the relative abundance of OTUs in a minimal number of samples. Returns a logical vector of OTUs that are at least n\

Usage

```
prop_filter(x, n, m)
```

Arguments

`x`	A phyloseq object.
`n`	Minimum percentage to keep OTU.
`m`	Minimum number of samples.

Details

The functions creates a logical vector to be used in subsetting a phyloseq object according to the relative abundance of OTUs in a given number of samples. For example, if n = 1 and m = 2, then the OTUs to be kept must represent at least 1\

Value

A logical vector of OTUs to keep.

Examples

```
{
data("its.root")
prop_filter(x = its.root, n = 1, m = 5)
}
```

`QsRutils` *QsRutils: R Functions Useful for Community Ecology*

Description

The QsRutils package contains functions I have written to make some aspects of using phyloseq and vegan simpler. I originally called the package MyRutils, but that does not make much sense if I am posting it publically!

rad2deg *rad2deg*

Description

Transform an angle in radians to degrees.

Usage

```
rad2deg(x)
```

Arguments

x	Angle in radians

Value

Angle in degrees.

Examples

```
rad2deg(pi * 0.5)
```

rda_labels *Make RDA Axis Labels*

Description

Makes RDA axis labels that include the % total variance explained by each RDA axis, that is by the constrained portion of the analysis.

Usage

```
rda_labels(rda)
```

Arguments

rda	A constrained ordination object made with vegan::rda() or vegan::cca().

Details

Each element of the vector returned has the form "RDAn xx.x%" where n is the number of the RDA axis and xx.x is the % of total variance explained by the axis. The percent of total variance is for the constrained portion only.

Value

A character vector, each element of which can be used to label the corresponding axis of an RDA plot.

See Also

ord_labels()

Examples

```
{
# Using vegan::rda()
data(dune, package = "vegan")
data(dune.env, package = "vegan")
dune.Manure <- vegan::rda(dune ~ Manure, dune.env)
print("For the rda case:")
print(rda_labels(dune.Manure)[1:2])
cat("\n")
# Using vegan::cca()
data(varespec, package = "vegan")
data(varechem, package = "vegan")
vare.cca <- vegan::cca(varespec ~ Al + P*(K + Baresoil), data=varechem)
print("For the cca case:")
print(rda_labels(vare.cca)[1:2])
}
```

---

root_phyloseq_tree *Root Tree in phyloseq Object*

Description

Roots an unrooted tree in a phyloseq object

Usage

```
root_phyloseq_tree(phylo)
```

Arguments

phylo	A phyloseq object containing an unrooted tree

Details

The tree is rooted by the longest terminal branch.

Value

The same phyloseq object with a rooted tree

Examples

```
{
data("expt")
expt.rooted <- root_phyloseq_tree(expt)
ape::is.rooted(phyloseq::phy_tree(expt.rooted))
}
```

---

se *Standard error*

Description

Calculates the standard error of a numeric vector

Usage

```
se(x)
```

Arguments

x	A numeric vector

Details

NA values are ignored.

Value

The standard error of the numeric vector

Examples

```
x <- c(1,2,3,4,5, NA)
se(x)
```

sqrt_arc_sine *sqrt_arc_sine*

Description

Transform a percentage to the square root of its arc sine.

Usage

```
sqrt_arc_sine(x)
```

Arguments

x	A percentage.

Value

The square root of the arcsine transformation of x.

Examples

```
sqrt_arc_sine(30.1)
```

srs_p *srs_p*

Description

Normalize sample counts using scaling with ranked subsampling (SRS)

Usage

```
srs_p(p)
```

Arguments

p	a phyloseq object containing an OTU table

Details

This is an alternative to "rarefying" an OTU table to a constant sample size. The phyloseq object submitted must be pruned to the desired sample size before using this function.

Value

a phyloseq object including an OTU table, all sample sums equal.

Author(s)

John Quensen

References

Beule L, Karlovsky P. Improved normalization of species count data in ecology by scaling with ranked subsampling (SRS): application to microbial communities. PeerJ. 2020;8:e9593.

Examples

```
{
data("expt")
print("Sample sums before srs_p")
print(phyloseq::sample_sums(expt))
cat("\n")
expt_srs <- srs_p(p =  expt)
print("Sample sums after srs_p")
print(phyloseq::sample_sums(expt_srs))
}
```

`subset_by_refseq_lengths` *Subset physeq by refseq lengths*

Description

Subset physeq by refseq lengths

Usage

```
subset_by_refseq_lengths(p, min_len = 252, max_len = 255)
```

Arguments

`p`	An experiment level phyloseq object with reference sequences
`min_len`	The minimum reference sequence length to keep
`max_len`	The maximum references sequences length to keep

Details

Sometimes, due to sequencing errors, reference sequences have lengths less than and/or greater than the expected amplicon length. This function offers an easy way of removing such extraneous reference sequences from an experiment level phyloseq object. The default values for min_len and max_len are fro the V4 region of the 16S rRNA gene.

Value

A phyloseq object filtered to have reference sequences within the length range specified.

Examples

```
{
data("expt")

print("Refseq length range before subsetting:")
expt@refseq@ranges@width |>
  summary() |>
  print()
cat("\n")

expt_filt <- subset_by_refseq_lengths(p = expt,
                                      min_len = 400,
                                      max_len = 420)

 print("Refseq length range after subsetting:")
expt_filt@refseq@ranges@width |>
  summary() |>
  print()
}
```

`subset_dist` *Subset Distance Matrix*

Description

Subsets a distance matrix.

Usage

```
subset_dist(physeq, d.matrix)
```

Arguments

`physeq`	An experiment level phyloseq object.
`d.matrix`	A distance matrix.

Details

Some distance matrices take a long time to calculate for large data sets. This is especially true of unifrac and generalized unifrac distances calculated by GUniFracs. If distances are first calculated from data in a large experiment level phyloseq object and then it is desired to perform PERMANOVA (with adonis) on a subset of that object, this function provides a means of sub-setting the distance matrix so that it does not have to be calculated again for the subset data. The arguments are the distance matrix for the original phyloseq object and the smaller phyloseq object subset from the original.

Value

A distance matrix of smaller dimensions.

References

Chen J, Bittinger K, Charlson ES et al. (2012) Associating microbiome composition with environmental covariates using generalized UniFrac distances. Bioinformatics, 28, 2106-2113.

Examples

```
{
otu <- veganotu(its.root)
d <- vegan::vegdist(otu)
print("Before subsetting:")
print(dim(as.matrix(d)))
cat("\n")
p <- phyloseq::subset_samples(its.root, P_Location == "LR")
d_sub <- subset_dist(p, d)
print("After subsetting:")
print(dim(as.matrix(d_sub)))
}
```

---

vegan_stand *Standardize a Phyloseq OTU Table*

Description

Applies any vegan decostand standardization method to a phyloseq OTU table.

Usage

```
vegan_stand(physeq, method = "hellinger", ...)
```

Arguments

physeq	A phyloseq object containing at least an OTU table.
method	A method from vegan's decostand function.
...	Other parameters passed to vegan's decostand function.

Value

Returns a phyloseq object with transformed OTU table.

Examples

```
{
data("expt")
print("Before standardization, first 5 rows and columns:")
print(veganotu(expt)[1:5, 1:5])
cat("\n")

expt_mod <- vegan_stand(expt, method = "hellinger")
print("After standardization, first 5 rows and columns:")
print(veganotu(expt_mod)[1:5, 1:5])
}
```

---

veganotu *Extract Vegan OTU Table*

Description

Extracts a vegan compatible OTU table from a phyloseq object.

Usage

```
veganotu(physeq)
```

Arguments

physeq	A phyloseq object contaning at least an OTU table.

Value

A matrix with samples in rows and OTUs in columns.

Examples

```
{
data("expt")
# Show only first 5 columns and rows:
veganotu(physeq = expt)[1:5, 1:5]
}
```

vegansam *Extract Sample Data Table*

Description

Extracts a sample data table from a phyloseq object.

Usage

```
vegansam(physeq)
```

Arguments

physeq	A phyloseq object containing sample_data.

Value

A data frame with samples in rows and factors and/or variables in columns.

Examples

```
{
data("expt")
vegansam(physeq = expt)
}
```

Index

∗ **datasets**
    expt, 19
    its.root, 26
    plot_df, 33
∗ **multivariate**
    avg_dist, 6
%wo%, 3

arc_sine, 4
asterix, 4
avg_alpha, 5
avg_dist, 6

check_primer_hits, 8
check_var, 9
cld_dunn, 10
cld_hsd, 11
clear_warnings, 11
comb, 12
comp_assemble, 13
comp_comparisons, 14
comp_make_f_tests, 15
comp_means_sd, 16
comp_prepare_otu_table, 17
comp_prepare_phyloseq, 18

deg2rad, 19

expt, 19
extract_adapter_content, 20

find_truncation_position, 20
format_taxon, 21

generate_password, 22
get_groups, 23
get_plot_limits, 24
goods, 24

hash_dna_seqs, 25

its.root, 26

log_arc_sine, 26

make_comparisons, 27
make_letter_assignments, 28
makeCluster, *6*
merge_2_frames, 29

ord_labels, 30

pca_labels, 31
perm, 32
plot_df, 33
plot_transition_stats, 33
print.check_var *(*check_var*)*, 9
prop_filter, 34

QsRutils, 34

rad2deg, 35
rda_labels, 35
rda_labels(), *30*
root_phyloseq_tree, 36
rrarefy, *7*

se, 37
sqrt_arc_sine, 38
srs_p, 38
subset_by_refseq_lengths, 39
subset_dist, 40

vegan::rarefy(), *7*
vegan::vegdist(), *7*
vegan_stand, 41
veganotu, 42
vegansam, 43
vegdist, *6, 7*