Package ‘LeyLabRMisc’

August 29, 2021

Type Package

Title Ley Lab misc R functions, rmd templates, etc.

Version 0.2.0

Author Nick Youngblut

Maintainer Nick Youngblut <nyoungb2@gmail.com>

Description Ley Lab misc R functions, rmd templates, etc.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

R topics documented:

`.ancombc_tidy` *tidy ancom-bc output*

Description

tidy ancom-bc output

Usage

```
.ancombc_tidy(var, L)
```

`.get_brewer_palette` *getting RColorBrewer entire palette*

Description

getting RColorBrewer entire palette

Usage

```
.get_brewer_palette()
```

`.HFE` *supporting function for HFE*

Description

supporting function for HFE

Usage

```
.HFE(
  brk,
  class_level,
  corr_cutoff = 0.5,
  freqCut = 99/1,
  uniqueCut = 1,
  quiet = TRUE
)
```

`.tidy_PCoA` *Convert PCoA object to a tidy dataframe*

Description

Convert PCoA object to a tidy dataframe

Usage

```
.tidy_PCoA(pcoa, k = 3)
```

Arguments

`pcoa`	A pcoa object generated by cmdscale
`k`	The number of PCs to keep

Value

A data.frame of PCoA points for the top k PCs

`.well384_index` *making 384-well plate index*

Description

making 384-well plate index

Usage

```
.well384_index()
```

Value

named vector (Well –> location); column-wise location

`.well96_index` *making a 96-well plate index*

Description

making a 96-well plate index

Usage

```
.well96_index()
```

Value

named vector (Well –> location); column-wise location

`ancombc_tidy` *Tidy ANCOM-BC output*

Description

Create a tidy table of ANCOM-BC output

Usage

```
ancombc_tidy(ancombc_out)
```

Arguments

`ancombc_out`	output object from the ancombc() function

Value

a tibble of tidy data

`ancombc_unbiased_abundances` *Get unbiased abundances from ANCOM-BC output*

Description

See https://bioconductor.org/packages/release/bioc/vignettes/ANCOMBC/inst/doc/ANCOMBC.html for more info on unbiased abundances.

Usage

```
ancombc_unbiased_abundances(ancombc_out, phyloseq_obj)
```

Arguments

`ancombc_out`	output object from the ancombc() function
`phyloseq_ojb`	phyloseq object used as input for ancombc() function

Value

a data.frame of abundances

`as.Num` *convert to numeric while avoiding factor conversion issues*

Description

convert to numeric while avoiding factor conversion issues

Usage

```
as.Num(x)
```

Arguments

`x`	an iterable

Value

a numeric object

bash_job *bash job using conda env*

Description

The conda setup is assumed to be in your ~/.bashrc If print_output == TRUE: the stdout/stderr will be printed instead of returned Else: the stdout/stderr with be returned by the function stderr/stdout is printed unless print_output==FALSE

Usage

```
bash_job(
  cmd,
  conda_env = NULL,
  stdout = TRUE,
  stderr = TRUE,
  print_output = TRUE,
  return_output = FALSE,
  log_file = NULL,
  verbose = TRUE,
  wait = TRUE
)
```

Arguments

`cmd`	The bash command in a string format
`conda_env`	The conda env to use
`stdout`	Print the stdout from the command?
`stderr`	Print the stderr from the command?
`print_output`	Pretty printing of the output to the console?
`return_output`	Return the bash command output?
`log_file`	Write stdout to log file (stderr written to log_file.err)
`verbose`	Write status messages?
`wait`	Wait for the process to finish?

Examples

```
# simple
bash_job('ls -thlc')
# write to log file
bash_job('ls -thlc', log_file='log.txt')
# use conda env
bash_job('conda list', conda_env='base')
```

`beta2mtx` *Convert tidy beta diversity table to a wide distance matrix*

Description

The input should have the columns: Measure, SampleX, SampleY, Value

Usage

```
beta2mtx(dt)
```

Arguments

`dt`	data.table, data.frame, or tibble

Details

The output can be used for creating a PCoA. dendrogram, etc

Value

a symmetric matrix of distances

`calculate_rarefaction_curves` *Function for rarefaction analysis*

Description

Running estimate_richness_phy() at multiple subsampling depths

Usage

```
calculate_rarefaction_curves(psdata, measures, depths, parallel = FALSE)
```

Arguments

`psdata`	phyloseq object
`measures`	Which diversity measures (see vegan package)
`depths`	Which sequencing depths? Example: c(10, 100, 1000)

Value

A dataframe

calc_alpha_div — *Calculate common alpha-diversity metrics You need the "vegan" package installed to your R project and loaded for this code to run*

Description

Faith's Phylogenetic Diversity ("PD") can be calculated only if a tree is provided. The tree can have extra tips, but there must be tip labels for all taxa in the provided table.

Usage

```
calc_alpha_div(df, tree = NULL, index = c("nobs", "shannon", "PD"))
```

Arguments

`df`	sample x taxon abundance table (usual format for vegan)
`tree`	tree with tips matching taxa in the abundance table (only needed for PD)
`index`	which of the indices to calculate? (nobs = no. of observations, shannon = Shannon Index, PD = Faith's PD)

Value

a data.frame of alpha diversity values (and sample names)

calc_beta_div — *beta-diversity calculation*

Description

A wrapper around vegan::vegdist and rbiom (rbiom used for UniFrac calculations). For unifrac: "wunifrac" = weighted unifrac, "unifrac" = unweighted unifrac. The function returns a tidy dataframe of PCoA axes (PC1 & PC2), percent variance explained for each PC.

Usage

```
calc_beta_div(
  df,
  tree = NULL,
 method = c("wunifrac", "unifrac", "manhattan", "euclidean", "canberra", "clark",
    "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn",
    "mountford", "raup", "binomial", "chao", "cao", "mahalanobis"),
  threads = 1
)
```

Arguments

`df`	sample x taxon dataframe. Colnames (taxa) must match the tree tip labels if the tree is provided
`tree`	phylogeny with tips matching the df colnames (only needed for wunifrac & unifrac methods)
`method`	distance method (vegdist distances; wunifrac=Weighted Unifrac; unifrac=Unweighted Unifrac)
`threads`	threads used for UniFrac calculations with rbiom

Details

Unifrac is calculated with the https://github.com/cmmr/rbiom package (requires bioconductor packages).

If the goal is PCoA, then see the "tidy_PCoA" function.

Value

data.frame

`calc_PCoA` *Wrapper for cmdscale*

Description

Simple wrapper for cmdscale to provide data.frame formatted table. If the distance matrices contain NAs, the samples containing NAs will be removed (with a warning).

Usage

```
calc_PCoA(dist_mtx, k = 2)
```

Arguments

`dist_mtx`	distance matrix object

Value

data.frame

`cat_file` *pretty printing of a text file via cat*

Description

This is most useful for working with IRkernl in Jupyter notebooks

Usage

```
cat_file(file_name)
```

Arguments

`file_name`	the name of the file to print

`clustermq_get_logs` *Get/read clustermq cluster job log files*

Description

If you use "log_file = clustermq_logfile()" in your template, then you can use this function to get the log file paths or directly read the contents of the log files.

Usage

```
clustermq_get_logs(lines = 0, logfile_dir = NULL)
```

Arguments

`lines`	The number of lines of each log file to read. If 0, then the log file paths will be returned; if >0 then the first N lines will be printed; if <0 then the last N lines will be printed.
`logfile_dir`	The base directory containing all of the logfiles. If not provided, then this is obtained by getOption('clustermq.logfile')

Value

logfile paths or NULL

Examples

```
clustermq_setup()
tmpl = list(job_mem = '8G', log_file = clustermq_logfile())
fx = function(x, y) x * 2 + y
Q(fx, x=1:3, const=list(y=10), n_jobs=10, job_size=1, template=tmpl)
clustermq_get_logs()            # getting log file paths
clustermq_get_logs(lines=10)    # reading the first 10 lines
clustermq_get_logs(lines=-10)   # reading the list 10 lines
```

clustermq_logfile *Set a path for clustermq cluster job log files*

Description

Log files are optional for clustermq. The must be set in the template. This function will create a unique directory within the "base_dir". It will also return a path that you MUST use for the "log_file" parameter in the Q template. Moreover, the function will set the "clustermq.logfile" option to that directory (used by clustermq_get_logs).

Usage

```
clustermq_logfile(base_dir = "/ebio/abt3_scratch/")
```

Arguments

base_dir	The base directory will the logfiles will be located.

Details

The function requires the uuid package.

Value

logfile path

Examples

```
clustermq_setup()
tmpl = list(job_mem = '8G', log_file = clustermq_logfile())
fx = function(x, y) x * 2 + y
Q(fx, x=1:3, const=list(y=10), n_jobs=10, job_size=1, template=tmpl)
```

clustermq_setup *Set clustermq options*

Description

These options must be set before running clustermq

Usage

```
clustermq_setup(
  scheduler = c("sge", "multicore"),
  template = file.path(Sys.getenv("HOME"), ".clustermq.tmpl")
)
```

Arguments

`scheduler`	The clustermq.scheduler option. Use "multicore" for local jobs.
`template`	The clustermq.template option. It defaults to ~.clustermq.tmpl

Examples

```
clustermq_setup()              # sge job
clustermq_setup('multicore')   # local job
```

`condaInfo` *"conda list" in R*

Description

This is most useful for working with IRkernl in Jupyter notebooks

Usage

```
condaInfo(conda_env)
```

Arguments

`conda_env`	The name of the conda env to list

`df.dims` *Changing number of rows/columns shown when printing a data frame*

Description

This is most useful for working with IRkernl in Jupyter notebooks

Usage

```
df.dims(nrows = 4, ncols = 20)
```

Arguments

`nrows`	number of rows to print
`ncols`	number of columns to print

`dfhead` *A simple dataframe summary*

Description

A simple dataframe summary

Usage

```
dfhead(df, n = 3)
```

Arguments

`df`	dataframe object
`n`	Number of lines to print

Value

a dataframe object

`dist_format` *creating a string with distance & percent explained*

Description

creating a string with distance & percent explained

Usage

```
dist_format(dist, PC1_perc_exp, PC2_perc_exp, label1 = 1, label2 = 2)
```

Arguments

`dist`	str, distance metric
`PC1_perc_exp`	float, percent variance explained for PC1
`PC2_perc_exp`	float, percent variance explained for PC2
`label1`	First PC label
`label2`	Seconda PC label

Value

str, formatted as "metric, <PC1_perc_exp>

`ena_get_filereport` *Get file reports via ENA Portal API*

Description

Works at least with sample and run accessions.

Usage

```
ena_get_filereport(
  accession,
  fields = c("accession", "run_accession", "nominal_length", "read_count",
   "base_count", "library_selection", "environment_biome", "environment_feature",
   "environment_material", "instrument_model", "instrument_platform", "library_name",
   "library_strategy", "sample_accession", "sample_collection", "sampling_platform",
    "sequencing_method", "project_name"),
  base_url = "https://www.ebi.ac.uk/ena/portal/api/",
  ...
)
```

Arguments

`accession`	Sample/run accession
`fields`	Fields to return. Use ena_get_search_fields() to list all possible fields.
`base_url`	ENA API base url
`...`	Parameters passed to httr::GET

Details

ENA Portal API: https://www.ebi.ac.uk/ena/portal/api/

Value

data.frame

Examples

```
df = ena_get_filereport('ERR479486')
df = ena_get_filereport('SRS2472312')
```

ena_get_search_fields *Get possible search fields*

Description

ENA Portal API: https://www.ebi.ac.uk/ena/portal/api/

Usage

```
ena_get_search_fields(
  section = c("read_run", "study", "sample"),
  base_url = "https://www.ebi.ac.uk/ena/portal/api/",
  ...
)
```

Arguments

section	Section to query (eg., read_run)
base_url	ENA API base url
...	Parameters passed to httr::GET

Value

data.frame

Examples

```
df = ena_get_search_fields()
df = ena_get_search_fields('sample')
```

estimate_rarified_richness *Helper Function for rarefaction analysis*

Description

Helper Function for rarefaction analysis

Usage

```
estimate_rarified_richness(psdata, measures, depth)
```

Arguments

`psdata`	phyloseq object
`measures`	Which diversity measures
`depth`	The sampling depth

Value

molten alpha diversity object

`estimate_richness_phy` *phyloseq::estimate_richness, but includes Faith's PD*

Description

See physeq::estimate richness for full details

Usage

```
estimate_richness_phy(physeq, split = TRUE, measures = NULL)
```

Arguments

`physeq`	Phyloseq object
`split`	Splitting the OTU table
`measures`	Which diversity measures (Faith's PD = "FaithPD)

Value

Dataframe

`expand.grid.lower` *expand.grid(), but just lower-triange comparisions*

Description

This is useful when you want pairwise comparisons, but you don't need the reciprical ('a' <=> 'b' & 'b' <=> 'a').

Usage

```
expand.grid.lower(x, y, diag = FALSE)
```

Arguments

`x`	a vector
`y`	a vector
`diag`	include same-same comparisons ('a' <=> 'b')?

Value

a data.frame of all non-reciprical comparisons

Examples

```
expand.grid.lower(1:3, 1:3)
expand.grid.lower(1:3, 1:3, diag=TRUE)
```

`extract_pltdt` *Extract data from ggplot object*

Description

The data is written to files

Usage

```
extract_pltdt(plot_object, output_path)
```

Arguments

`plot_object`	A ggplot object
`output_path`	Where to write the output

`fig_uuid` *create UUID for figure file name*

Description

create UUID for figure file name

Usage

```
fig_uuid(full = FALSE)
```

Arguments

`full`	Full length uuid or trimmed to just 24 char?

Value

character object

`files_to_list` *convert a vector of file paths into a named list*

Description

convert a vector of file paths into a named list

Usage

```
files_to_list(files, label_index = -1)
```

Arguments

`files`	Vector of file paths (eg., by using "list_files()")
`label_index`	Which item in the path to return? 1-indexing. If <1, samples selected from the end.

Value

list of files

Examples

```
files = c('/path/to/project/Sample1/table.txt', '/path/to/project/Sample2/table.txt')
files_to_list(files, -1)
files = c('/path/to/project/Sample1.txt', '/path/to/project/Sample2.txt')
files_to_list(files, 0)
```

`Fread` *Simple wrapper around data.table::fread*

Description

Simple wrapper around data.table::fread

Usage

```
Fread(
  infile = NULL,
  cmd = NULL,
  sep = "\t",
  check.names = TRUE,
  tmpdir = file.path("/ebio", "abt3_scratch", Sys.info()[["user"]], "R_tmp"),
  ...
)
```

Arguments

`infile`	Input file name
`cmd`	Command instead of input file (eg., "gunzip -c INFILE")
`sep`	Value delimiter
`check.names`	Format check column names
`...`	Passed to data.table::fread
`tmp_dir`	Temp file directory. Scratch directory by default

Value

data.table

`hello` *Hello, World!*

Description

Prints 'Hello, world!'.

Usage

```
hello()
```

Examples

```
hello()
```

`itol_boxplot` *create itol boxplot file*

Description

https://itol.embl.de/help.cgi#boxplot

Usage

```
itol_boxplot(
  df,
  dataset_label,
  out_file,
  out_dir = NULL,
  key_color = "#ff0000",
  WIDTH = 200
)
```

Arguments

`df`	Dataframe, in which the rownames should correspond with the tree labels; the columns must specify: minimum,q1,median,q3,maximum,extreme_value1,extreme_value2
`dataset_label`	What to label the itol dataset
`out_file`	Name of the output file
`out_dir`	Where to write the output
`key_color`	The color for the legend key
`WIDTH`	Maximum width

`itol_colorstrip` *create itol colorstrip file*

Description

https://itol.embl.de/help.cgi#strip

Usage

```
itol_colorstrip(df, dataset_label, out_file, out_dir = NULL, legend = NULL)
```

Arguments

`df`	Dataframe, in which the rownames should correspond with the tree labels; the plotting parameter should be column 1
`dataset_label`	What to label the itol dataset
`out_file`	Name of the output file
`out_dir`	Where to write the output
`legend`	Custom legend (see the function description)

Details

Custom Legend: requires a data.frame with the number of rows equaling the number of unique values in the legend.

- "shapes" => numeric (see the itol docs)
- "colors" => hexidecimal (see this website for examples)
- "labels" => legend labels

Examples

```
# creating a custom legend
legend = data.frame(unique(iris$Species),
colors = c('#00FF00', '#FFCC33', '#FF0000'),
shapes = rep(1, length(unique(iris$Species))))
legend
```

`itol_externalshape` *create itol external shape file*

Description

https://itol.embl.de/help.cgi#shapes

Usage

```
itol_externalshape(
  df,
  dataset_label,
  out_file,
  out_dir = NULL,
  legend = NULL,
  WIDTH = 200
)
```

Arguments

`df`	Dataframe, in which the rownames should correspond with the tree labels; other columns should be values corresponding to symbol size
`dataset_label`	What to label the itol dataset
`out_file`	Name of the output file
`out_dir`	Where to write the output
`legend`	Specify particular legend (see `itol_colorstrip`)

`itol_heatmap` *create itol heatmap file*

Description

https://itol.embl.de/help.cgi#heatmap

Usage

```
itol_heatmap(
  df,
  dataset_label,
  out_file,
  out_dir = NULL,
  tree = NULL,
  dist_method = "bray",
  color_scheme = c("color", "bw")
)
```

Arguments

`df`	Dataframe, in which the rownames should correspond with the tree labels; all columns should be numeric values for the heatmap
`dataset_label`	What to label the itol dataset
`out_file`	Name of the output file
`out_dir`	Where to write the output
`tree`	Tree object used for ordering the heatmap columns; if NULL, the dist_method will be used to create the tree
`dist_method`	vegan::vegdist method for creating the correlation dendrogram
`color_scheme`	Heatmap color scheme. color = blue-orange-yellow; bw=white-grey-black

---

`itol_multibar` *create itol multi-bar file*

Description

https://itol.embl.de/help.cgi#multibar

Usage

```
itol_multibar(
  df,
  dataset_label,
  out_file,
  out_dir = NULL,
  legend = NULL,
  WIDTH = 200,
  COLOR = "#ff0000"
)
```

Arguments

`df`	Dataframe, in which the rownames should correspond with the tree labels
`dataset_label`	What to label the itol dataset
`out_file`	Name of the output file
`out_dir`	Where to write the output
`legend`	A list that includes shapes, colors, and labels (see `itol_colorstrip`)
`WIDTH`	Bar width
`COLOR`	Legend color

`itol_simplebar` *create itol simple-bar file*

Description

https://itol.embl.de/help.cgi#bar

Usage

```
itol_simplebar(
  df,
  dataset_label,
  out_file,
  out_dir = NULL,
  legend = NULL,
  WIDTH = 200
)
```

Arguments

`df`	Dataframe, the rownames should correspond with the tree labels
`dataset_label`	What to label the itol dataset
`out_file`	Name of the output file
`out_dir`	Where to write the output
`legend`	Specify particular legend (see `itol_colorstrip`)
`WIDTH`	Bar width

`itol_symbol` *create itol symbol file*

Description

https://itol.embl.de/help.cgi#symbols

Usage

```
itol_symbol(
  df,
  dataset_label,
  out_file,
  out_dir = NULL,
  MAXIMUM_SIZE = 50,
  COLOR = "#ff0000"
)
```

Arguments

`df`	Dataframe, in which the rownames should correspond with the tree internal node labels, and other columns should be: symbol,size,color,fill,position,(label)
`dataset_label`	What to label the itol dataset
`out_file`	Name of the output file
`out_dir`	Where to write the output
`MAXIMUM_SIZE`	The max size of the symbols
`COLOR`	Legend color

`list_files` *list.files with full.names=TRUE & recursive=TRUE*

Description

list.files with full.names=TRUE & recursive=TRUE

Usage

```
list_files(path, pattern = NULL, full.names = TRUE, recursive = TRUE, ...)
```

Arguments

`path`	a character vector of full path names; the default corresponds to the working directory,
`pattern`	an optional regular expression. Only file names which match the regular expression will be returned.
`full.names`	a logical value. If TRUE, the directory path is prepended to the file names to give a relative file path. If FALSE, the file names (rather than paths) are returned
`recursive`	logical. Should the listing recurse into directories?

Value

A character vector containing the names of the files in the specified directories

make_dir *A helper function for creating a directory (recursively)*

Description

A helper function for creating a directory (recursively)

Usage

```
make_dir(dir, quiet = FALSE)
```

Arguments

dir	path for the new directory (will create recursively)
quite	quite output

mgnify_get *Get MGnify info via the API*

Description

MGnify API: https://www.ebi.ac.uk/metagenomics/api/v1/

Usage

```
mgnify_get(
  accession = NULL,
  section = c("samples", "studies", "analyses", "biomes", "experiment-types"),
  search = NULL,
  lineage = NULL,
  instrument_platform = NULL,
  instrument_model = NULL,
  query = list(),
  max_pages = 1,
  base_url = "https://www.ebi.ac.uk/metagenomics/api/v1/",
  cache_file = "mgnify_request.RDS",
  cache_break = 10,
  use_cache = TRUE,
  ...
)
```

Arguments

`accession`	Study accession (primary or secondary). If provided, just info returned for that study.
`section`	Section of the API to query.
`search`	General keyword search to filter records.
`lineage`	Filter by lineage (eg., "root:Host-associated:Human").
`query`	List of additional queries provided to httr::GET.
`max_pages`	The maximum number of pages of records to return.
`base_url`	MGnify API base url
`cache_file`	File name to cache (checkpoint) the results. Useful for big queries in case the job is interuppted.
`cache_break`	Write cache file every N pages.
`use_cache`	Read the cache file, if it exists?
`...`	Parameters passed to httr::GET
`instrutment_platform`	Sequencing instrument platform (eg., "ILLUMINA").
`instrutment_model`	Sequencing instrument model (eg., "HiSeq" or "MiSeq").

Details

This function can query any "section" of the API (eg., "studies" or "samples").

Main filtering options are listed in the function (eg., lineage). More queries can be provides as a list via the query parameter. Note that not all filtering options for work each section. To see all filtering options for each section, click the "Filters" button at https://www.ebi.ac.uk/metagenomics/api/v1/samples

To prevent accidental big queries, only 1 page of results is returned by default (max_pages).

Value

data.frame

Examples

```
mgnify_get()
mgnify_get(search='soil')
mgnify_get(section='biomes')
mgnify_get(section='experiment-types')
mgnify_get(accession = 'ERP009004', section='studies')
mgnify_get(lineage='root:Host-associated', instrument_platform = 'ILLUMINA',
           instrument_model = 'HiSeq', max_pages=8)
mgnify_get(accession = 'SRS2472313')
```

mgnify_request *GET request from the ENA*

Description

GET request from the ENA

Usage

```
mgnify_request(
  url,
  max_pages = NULL,
  query = list(),
  verbose = TRUE,
  cache_file = "mgnify_request.RDS",
  cache_break = 10,
  use_cache = TRUE,
  ...
)
```

Arguments

`url`	ENA API url
`max_pages`	Max number of pages to return. If NULL, all pages returned.
`query`	Query list passed to httr::GET
`verbose`	Verbose output?
`cache_file`	File name to cache (checkpoint) the results. Useful for big queries in case the job is interuppted.
`cache_break`	Write cache file every N pages.
`use_cache`	Read the cache file, if it exists?
`...`	Parameters passed to httr::GET

Value

data.frame

Examples

```
x = mgnify_request('https://www.ebi.ac.uk/metagenomics/api/v1/biomes', max_pages = 2)
```

`mgnify_request_get` *GET request to MGnify API*

Description

GET request to MGnify API

Usage

```
mgnify_request_get(url, page = 1, query = list(), verbose = TRUE, ...)
```

Arguments

`url`	Query url
`page`	Page of records to return
`query`	List passed to httr::GET(query=)
`verbose`	Verbose output?

Value

list(status = character, page = numberic, pages = numberic, data = data.frame)

Examples

```
mgnify_request_get('https://www.ebi.ac.uk/metagenomics/api/v1/samples', query=list(instrument_platform = 'ILLUM
```

`mlr_boruta_filter` *Custom mlr filter for Boruta*

Description

A custom mlr filter that uses Boruta to select important features This function registers the "boruta.filter" filter to be used with makeFilterWrapper and other mlr filter functions.

Usage

```
mlr_boruta_filter()
```

Details

- target str; what is the target variable in the task object (default: 'Class')
- pValue float; see Boruta docs (default: 0.01)
- maxRuns int; see Boruta docs (default: 200)
- hostHistory bool; see Boruta docs (default: FALSE)
- withTentative bool; keep tentative features (default: TRUE)
- verbose bool; list features selected? (default: FALSE)
- mustKeep vector; features that cannot be filtered (default: NULL)
- threads int; number of threads to use for Boruta (default: 1)

Value

Nothing, but "boruta.filter" filter will be registered

`mlr_getNestedTuneResultsOptPathDf` *Version of getNestedTuneResultsOptPathDf that actually works*

Description

For main docs, see ?getNestedTuneResultsOptPathDf

Usage

```
mlr_getNestedTuneResultsOptPathDf(r, trafo = FALSE)
```

Arguments

`r`	The result of resampling of a tuning wrapper
`trafo`	Should the units of the hyperparameter path be converted to the transformed scale?

Value

data.frame

`ml_tax_HFE` *Hierarchical Feature Selection*

Description

For each clade (defined by tax_level), aggregate species abundances at each taxonomic level up to the user-defined "tax_level", (optionally filter out near-zero features), then filter out taxa that correlate strongly (just one taxon is selected of those that correlate).

Usage

```
ml_tax_HFE(
  brk,
  tax_level,
  corr_cutoff = 0.7,
  threads = 2,
  freqCut = 95/1,
  uniqueCut = 5,
  quiet = TRUE
)
```

Arguments

`brk`	data.table generated by read_bracken(). Columns: Sample, Abundance, Phylum=>Species
`tax_level`	which taxonmoic level to use?
`corr_cutoff`	features with >cutoff will be filtered to just one
`freqCut`	as in caret::nearZeroVar; use NULL to skip
`uniqueCut`	as in caret::nearZeroVar; use NULL to skip

Value

data.table of filtered features

`overlap` *Determine counts of setdiff, intersect, & union of 2 vectors (or data.tables)*

Description

The output is printed text of intersect, each-way setdiff, and union. Data.table compatible! Just make sure to provide sel_col_x and/or sel_col_y

Usage

```
overlap(
  x,
  y,
  sel_col_x = NULL,
  sel_col_y = NULL,
  to_return = c("counts", "diff_x", "diff_y", "diff_fuzzy"),
  diff = c(NA, "x", "y", "int", "union", "fuzzy")
)
```

Arguments

`x`	vector1 or data.table. If data.table, sel_col_x must not be NULL
`y`	vector2 or data.table. If data.table, sel_col_y must not be NULL
`sel_col_x`	If x = data.table, which column to assess?
`sel_col_y`	If y = data.table, which column to assess?
`to_return`	(depreciated) "counts" = print overlap counts; "diff_x-or-y" = return setdiff; "diff_fuzzy" = return closest matches for those that differ (ordered best to worst)
`diff`	Alternative to "to_return". "x" or "y" = return setdiff; "int" = intersect, "union" = union

---

`p.dims` *Global change of plot size options*

Description

This is most useful for working with IRkernl in Jupyter notebooks

Usage

```
p.dims(w = 5, h = 5, res = 200)
```

Arguments

`w`	figure width
`h`	figure height
`res`	figure resolution (DPI)

`path_get_label` *splitting path and returning just one item in the vector*

Description

This is useful for merging tables in which the individual table ID is within the file path.

Usage

```
path_get_label(file_path, index)
```

Arguments

`file_path`	File path(s). If vector or list of paths provided, then a list will be returned
`index`	Which item in the path to return? 1-indexing. If <1, samples selected from the end. "O" will select the file name.

Value

string if 1 path, else list

`phyloseq2df` *Convert a sub-object of a phyloseq object to a dataframe*

Description

A helper function for converting OTU, taxonomy, and metadata to dataframes

Usage

```
phyloseq2df(physeq_obj, physeq_func, long = FALSE, flip = FALSE)
```

Arguments

`physeq_obj`	The phyloseq object
`physeq_func`	Which object do you want (’otu_table’, ’tax_table’, or ’sample_data’)
`long`	Do you want the table in "long" format ("gathered")
`flip`	Flip (transpose) the table?

Value

A tibble

`phyloseq_rel_abund` *Transform abundances to relative*

Description

A simple wrapper for transform_sample_counts()

Usage

```
phyloseq_rel_abund(physeq_obj, percent_abund = TRUE)
```

Arguments

`physeq_obj`	The phyloseq object
`percent_abund`	Fractional or percent abundance?

Value

A phyloseq object

`pipelineInfo` *pipeline sessionInfo*

Description

sessionInfo for LeyLab snakemake pipelines

Usage

```
pipelineInfo(pipeline_path, head_n = 10)
```

Arguments

`pipeline_path`	The path to the pipeline directory
`head_n`	The number of lines to print from the readme

Plot — *plot figure and save the figure grob object to a file at the same time*

Description

This is most useful for working with IRkernl in Jupyter notebooks

Usage

```
Plot(
  p,
  file = NULL,
  path = NULL,
  suffix = "",
  saveObj = TRUE,
  saveImg = FALSE,
  width = NA,
  height = NA,
  ...
)
```

Arguments

`p`	Plot object (ggplot2, base, etc)
`file`	File name to write. If NULL, the name will be based on the md5sum of the object, so the name will change if the object changes.
`path`	Path to write to. If NULL, the path will be .figures/.
`suffix`	File name suffix (eg., '.png')
`saveObj`	Write the Robj to a file?
`saveImg`	Write the image to a file?
`width`	Figure width. If NA, uses global options
`height`	Figure height. If NA, uses global options

qsave_obj — *Simple function for serializing a distance matrix or list of distance matrices*

Description

Serializing done with the "qs" R package.

Usage

```
qsave_obj(x, file, msg = "Writing file to: ", threads = 1)
```

Arguments

`x`	a distance matrix or list of distance matrices
`file`	file name to save to
`threads`	number of threads used for serializing

Value

the input distance matrix or list of distance matrices

`readLinesTail` *Read the last N lines of a file*

Description

Read the last N lines of a file

Usage

```
readLinesTail(x, n, ...)
```

Arguments

`x`	The file name
`n`	The last N lines to read
`...`	Passed to scan()

`read_bracken` *Function for reading in a bracken taxonomy table*

Description

The table will be converted to long form (sample ~ abundance). Only "_frac" or "_num" columns will be kept (see "keep_frac"). Taxonomy will be split into separate levels (see "tax_levs"). tidytable (w/ data.table) used to speed the process up.

Usage

```
read_bracken(
  infile,
  nrows = Inf,
  keep_frac = TRUE,
 tax_levs = c("Domain", "Phylum", "Class", "Order", "Family", "Genus", "Species"),
  nThread = 4,
  ...
)
```

Arguments

`infile`	Path to bracken table file
`nrows`	Number of table rows to read. If Inf, all lines will be read.
`keep_frac`	If TRUE, keep all columns ending in "_frac"; otherwise, keep "_num" columns.
`tax_levs`	Taxonomic levels to separate the taxonomy column into.
`...`	Params passed to fread()

Value

data.table

`read_eggnog_mapper` *Function for reading in eggnog-mapper annotations and returning tidy subsets of the info*

Description

Many of the data in the eggnog-mapper annotation table (eg., generated by the LLG pipeline) is encoded as comma-delimited lists within a single column (eg., KEGG pathways). This makes it challenging to "tidy" the table.

Usage

```
read_eggnog_mapper(
  infile = NULL,
  cmd = NULL,
  sep = "\t",
  nrows = Inf,
  to_keep = c("COG", "KEGG pathway", "CAZy"),
  column_names = c("query_name", "seed_eggNOG_ortholog", "seed_ortholog_evalue",
   "seed_ortholog_score", "Predicted_taxonomic_group", "Predicted_protein_name",
   "Gene_Ontology_terms", "EC_number", "KEGG_ko", "KEGG_Pathway", "KEGG_Module",
   "KEGG_Reaction", "KEGG_rclass", "BRITE", "KEGG_TC", "CAZy", "BiGG_Reaction",
   "tax_scope__eggNOG_taxonomic_level_used_for_annotation", "eggNOG_OGs", "bestOG",
   "COG_Functional_Category", "eggNOG_free_text_description")
)
```

Arguments

`infile`	Path to eggnog-annotation table file
`cmd`	command instead of input file (eg., "gunzip -c INFILE")
`sep`	table value delimiter
`nrows`	Number of table rows to read. If Inf, all lines will be read.
`to_keep`	Which functional grouping to keep (eg., KEGG pathways)?
`column_names`	The column names to use for the table (use NULL if the input table has column names)

Details

This function will read in the table and output a tidy table of one part of the table (eg., COG functional categories or KEGG pathways).

The function will also provide info on how to obtain metadata for function groupings.

Value

data.table

`Robj_md5sum` *Dump an R object as text to a temp file and get the md5sum of the file*

Description

Dump an R object as text to a temp file and get the md5sum of the file

Usage

```
Robj_md5sum(Robj)
```

Arguments

`Robj`	Any R object

Value

md5sum

`row_means` *rowMeans that works inside a dplyr::mutate() call*

Description

rowMeans that works inside a dplyr::mutate() call

Usage

```
row_means(..., na.rm = TRUE)
```

`row_sums` *rowSums that works inside a dplyr::mutate() call*

Description

rowSums that works inside a dplyr::mutate() call

Usage

```
row_sums(..., na.rm = TRUE)
```

`scale_color_all` *Great a better coloring scheme for taxon abundance barcharts*

Description

The default coloring scheme for ggplot2 makes it hard to distinguish among data points in complex bar charts (eg., taxa plots). This function is a wrapper around scale_color_continuous() which changes the color scheme used.

Usage

```
scale_color_all(..., return_hex = FALSE)
```

Arguments

`...`	Parameters passed to scale_colorl_manual()
`return_hex`	Return a vector of color hexidecimals instead of a plotting object.

Value

ScaleContinuous/ggproto object or vector

Examples

```
ggplot(mpg, aes(cty, hwy, color=class)) +
  geom_point() +
  scale_color_all()
```

`scale_fill_all` *Great a better coloring scheme for taxon abundance barcharts*

Description

The default coloring scheme for ggplot2 makes it hard to distinguish among data points in complex bar charts (eg., taxa plots). This function is a wrapper around scale_color_continuous() which changes the color scheme used.

Usage

```
scale_fill_all(..., return_hex = FALSE)
```

Arguments

`...`	Parameters passed to scale_fill_manual()
`return_hex`	Return a vector of color hexidecimals instead of a plotting object.

Value

ScaleContinuous/ggproto object or vector

Examples

```
ggplot(mpg, aes(fl, hwy, fill=model)) +
  geom_bar(stat='identity') +
  scale_fill_all()
```

`send_email` *A helper function to send an email via the mail bash cmd*

Description

A helper function to send an email via the mail bash cmd

Usage

```
send_email(
  body,
  subject = "R job complete",
  email = NULL,
  email_ext = "tuebingen.mpg.de"
)
```

Arguments

`body`	The email body
`subject`	The email subject line
`email`	The email address. If NULL, then username used
`email_ext`	The part after the "at" symbol

Value

The output of the system() call

`size_objects` *Returns the sizes of R objects*

Description

Returns the sizes of R objects

Usage

```
size_objects(Robj)
```

Arguments

`Robj`	Vector with the names of R objects as characters

Value

A list with the name of R objects as names and the formatted size of the objects

`snakemakeInfo` *snakemake conda info*

Description

snakemake conda info

Usage

```
snakemakeInfo(config_file, pipeline_dir, conda_env)
```

Arguments

`config_file`	The path to the config file
`pipeline_dir`	The path to the pipeline_directory
`conda_env`	The conda env that has snakemake installed

Value

The environment info

`split_path` *python's os.path.split() for R*

Description

python's os.path.split() for R

Usage

```
split_path(x)
```

Arguments

`x`	The full file path

Value

A vector of all path parts

`summary_x` *Summary for numeric vectors that includes sd and stderr*

Description

sd = standard deviation stderr = standard error of the mean (sd(x) / sqrt(length(x)))

Usage

```
summary_x(x, label = NULL, sel_col = NULL, rnd = 3)
```

Arguments

`x`	a numeric vector
`label`	row name label for the output. If NULL, then the label will be the input object label.
`sel_col`	If "x" is data.table or data.frame, which column to assess?
`rnd`	number of digits to round sd and stderr to

Value

a matrix

Examples

```
summary_x(iris$Sepal.Length)
```

`taxonomy_levels` *A simple function that returns a vector of taxonomy levels*

Description

This just saves some typing, since I find myself constantly typing out: c('Domain', 'Phylum', 'Class', 'Order', 'Family', 'Genus', 'Species')

Usage

```
taxonomy_levels()
```

Value

character vector of taxonomic levels

`tidy_pcoa` *PCoA on a 'long' (tidy) tibble, and a long tibble is returned*

Description

Perform PCoA in a "tidy" way. If multiple diversity metrics are provided (eg., "bray" and "jaccard"), all PCoA results will be combined into one data.frame.

Usage

```
tidy_pcoa(
  df,
  taxon_col,
  sample_col,
  abundance_col,
  dists = c("bray", "jaccard", "wunifrac", "unifrac"),
  tree = NULL,
  threads = 1,
  threads_unifrac = 1,
  k = 2,
  dist_mtx_file = NULL,
  pcoa_file = NULL
)
```

Arguments

`df`	data.frame or tibble
`taxon_col`	the column specifying taxa or OTUs (no quotes needed)
`sample_col`	the column specifying sample names (no quotes needed)
`abundance_col`	the column specifying the taxon abundances in each sample (no quotes needed)
`dists`	vector of beta-diversity distances ('wunifrac' = weighted UniFrac, 'unifrac' = unweighted Unifrac; see vegan::vegsist for others)
`tree`	phylogeny for UniFrac calculations. It can have more tips that what is in the data.frame
`threads`	number of parallel calculations of each distance metric (1 thread per distance)
`threads_unifrac`	number of threads to use for wunifrac & unifrac calculations
`k`	passed to cmdscale
`dist_mtx_file`	file name for saving the distance matrices (qs serialization; use ".qs" for the file extension)
`pcoa_file`	file name for saving the raw pcoa results

Details

Weighted/Unweighted UniFrac is calculated via the rbiom R package. All other beta-diversity metrics are calculated via the vegan R package.

Value

a tibble of PCoA info for all selected "dists"

`to_rds` *Save object as RDS, with name automatically defined*

Description

Similar to the Plot() function, but for any R object. This is useful for quickly saving data for use in other sessions. For example, if one must compile tables of all p-values for manuscript submission.

Usage

```
to_rds(obj, file = NULL, path = NULL, suffix = "")
```

Arguments

`file`	File name to write. If NULL, the name will be based on the md5sum of the object, so the name will change if the object changes.
`path`	Path to write to. If NULL, the path will be .data/.
`suffix`	File name suffix,

unique_n *Pretty print number of unique elements in a vector*

Description

The result will be cat'ed to the screen. tidytable compatable. Maje

Usage

```
unique_n(x, label = "items", sel_col = NULL, ret = FALSE)
```

Arguments

x	a vector or data.table. If data.table, sel_col must not be NULL
label	what to call the items in the vector (eg., "samples")
sel_col	If x is data.table or data.frame, which column to assess?
ret	Return the unique values?

well2index *Convert between wellID and column-num*

Description

Useful for converting between WellIDs (eg., "A2") and well position in a plate (eg., 9)

Usage

```
well2index(x, plate_type = "96-well")
```

Arguments

x	A vector of well IDs
plate_type	Either 96-well or 384-well

Value

A vector of plate positions

write_table *writing table convience function*

Description

This is most useful for working with IRkernl in Jupyter notebooks. If a data.table is provided, then fwrite is used; otherwise, write.table is used.

Usage

```
write_table(df, file, sep = "\t", quote = FALSE, row.names = FALSE, ...)
```

Arguments

`df`	data.frame or data.table to write out
`file`	Output file path
`sep`	the field separator string. Values within each row of x are separated by this string
`quote`	a logical value (TRUE or FALSE) or a numeric vector. If TRUE, any character or factor columns will be surrounded by double quotes.
`row.names`	either a logical value indicating whether the row names of x are to be written along with x, or a character vector of row names to be written.
`...`	Passed to write.table (if data.frame) or fwrite (if data.table)

Index