

# Package ‘LeyLabRMisc’

August 2, 2020

**Type** Package

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

**Version** 0.1.6

**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** 7.1.0

## R topics documented:

.HFE . . . . .	3
.well384_index . . . . .	3
.well96_index . . . . .	3
as.Num . . . . .	4
bash_job . . . . .	4
calculate_rarefaction_curves . . . . .	5
calc_alpha_div . . . . .	5
calc_beta_div . . . . .	6
calc_PCoA . . . . .	7
cat_file . . . . .	7
clustermq_get_logs . . . . .	8
clustermq_logfile . . . . .	8
clustermq_setup . . . . .	9
condaInfo . . . . .	10
df.dims . . . . .	10
dfhead . . . . .	10
dist_format . . . . .	11
estimate_rarified_richness . . . . .	11
estimate_richness_phy . . . . .	12
extract_pltdt . . . . .	12

fig_uuid . . . . .	13
files_to_list . . . . .	13
Fread . . . . .	14
hello . . . . .	14
itol_boxplot . . . . .	15
itol_colorstrip . . . . .	15
itol_externalshape . . . . .	16
itol_heatmap . . . . .	17
itol_multibar . . . . .	18
itol_simplebar . . . . .	18
itol_symbol . . . . .	19
list_files . . . . .	20
make_dir . . . . .	20
mlr_boruta_filter . . . . .	21
mlr_getNestedTuneResultsOptPathDf . . . . .	21
ml_tax_HFE . . . . .	22
overlap . . . . .	22
p.dims . . . . .	23
path_get_label . . . . .	23
phyloseq2df . . . . .	24
phyloseq_rel_abund . . . . .	24
pipelineInfo . . . . .	25
Plot . . . . .	25
qsave_obj . . . . .	26
readLinesTail . . . . .	26
read_bracken . . . . .	27
read_eggnog_mapper . . . . .	27
Robj_md5sum . . . . .	28
row_means . . . . .	29
row_sums . . . . .	29
send_email . . . . .	29
size_objects . . . . .	30
snakemakeInfo . . . . .	30
split_path . . . . .	31
summary_x . . . . .	31
taxonomy_levels . . . . .	32
tidy_pcoa . . . . .	32
unique_n . . . . .	33
well2index . . . . .	34
write_table . . . . .	34

---

<code>.HFE</code>	<i>supporting function for HFE</i>
-------------------	------------------------------------

---

**Description**

supporting function for HFE

**Usage**

```
.HFE(brk, class_level, corr_cutoff = 0.5, quiet = TRUE)
```

---

<code>.well384_index</code>	<i>making 384-well plate index</i>
-----------------------------	------------------------------------

---

**Description**

making 384-well plate index

**Usage**

```
.well384_index()
```

**Value**

named vector (Well → location); column-wise location

---

<code>.well96_index</code>	<i>making a 96-well plate index</i>
----------------------------	-------------------------------------

---

**Description**

making a 96-well plate index

**Usage**

```
.well96_index()
```

**Value**

named vector (Well → location); column-wise location

`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 interable

**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 will be returned by the function `stderr/stdout` is printed unless `print_output==FALSE`

**Usage**`bash_job(cmd, conda_env, stdout = TRUE, stderr = TRUE, print_output = TRUE)`**Arguments**

<code>cmd</code>	The bash command in a string format
<code>conda_env</code>	The conda env to use
<code>stdout</code>	Print the stdout from the command?
<code>stderr</code>	Print the stderr from the command?
<code>quiet</code>	No printing

---

 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	<i>beta-diversity calculation</i>
---------------	-----------------------------------

---

## Description

A wrapper around `vegan::vegdist` and `rbiom` (`rbiom` used for UniFrac calculations). For unifracs: "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	<i>Wrapper for cmdscale</i>
-----------	-----------------------------

---

**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	<i>pretty printing of a text file via cat</i>
----------	---

---

**Description**

This is most useful for working with IRkernel in Jupyter notebooks

**Usage**

```
cat_file(file_name)
```

**Arguments**

file_name	the name of the file to print
-----------	-------------------------------

---

clustermq_get_logs	<i>Get/read clustermq cluster job log files</i>
--------------------	---

---

### 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 last 10 lines
```

---

clustermq_logfile	<i>Set a path for clustermq cluster job log files</i>
-------------------	---

---

### Description

Log files are optional for clustermq. They 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	<i>Set clustermq options</i>
-----------------	------------------------------

---

**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	<i>"conda list" in R</i>
-----------	--------------------------

---

**Description**

This is most useful for working with IRkernel in Jupyter notebooks

**Usage**

```
condaInfo(conda_env)
```

**Arguments**

conda_env	The name of the conda env to list
-----------	-----------------------------------

---

df.dims	<i>Changing number of rows/columns shown when printing a data frame</i>
---------	---

---

**Description**

This is most useful for working with IRkernel in Jupyter notebooks

**Usage**

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

**Arguments**

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

---

dfhead	<i>A simple dataframe summary</i>
--------	-----------------------------------

---

**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	<i>creating a string with distance &amp; percent explained</i>
-------------	--

---

**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>

---

estimate_rarified_richness	<i>Helper Function for rarefaction analysis</i>
----------------------------	---

---

**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	<i>phyloseq::estimate_richness, but includes Faith's PD</i>
-----------------------	---

---

**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 can calculate faith's PD (using Picante, "FaithPD")

---

extract_pltdt	<i>Extract data from ggplot object</i>
---------------	--

---

**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	<i>create UUID for figure file name</i>
----------	---

---

**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	<i>convert a vector of file paths into a named list</i>
---------------	---

---

**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.

**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	<i>Simple wrapper around data.table::fread</i>
-------	--

---

**Description**

Simple wrapper around data.table::fread

**Usage**

```
Fread(infile = NULL, cmd = NULL, sep = "\t", check.names = TRUE, ...)
```

**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

**Value**

data.table

---

hello	<i>Hello, World!</i>
-------	----------------------

---

**Description**

Prints 'Hello, world!'.

**Usage**

```
hello()
```

**Examples**

```
hello()
```

---

itol_boxplot	<i>create itol boxplot file</i>
--------------	---------------------------------

---

**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	<i>create itol colorstrip file</i>
-----------------	------------------------------------

---

**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" => hexadecimal (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	<i>create itol external shape file</i>
--------------------	--

---

**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 <a href="#">itol_colorstrip</a> )

---

itol_heatmap	<i>create itol heatmap file</i>
--------------	---------------------------------

---

**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	<i>create itol multi-bar file</i>
---------------	-----------------------------------

---

**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 <a href="#">itol_colorstrip</a> )
WIDTH	Bar width
COLOR	Legend color

---

itol_simplebar	<i>create itol simple-bar file</i>
----------------	------------------------------------

---

**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 <a href="#">itol_colorstrip</a> )
WIDTH	Bar width

---

itol_symbol	<i>create itol symbol file</i>
-------------	--------------------------------

---

**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	<i>list.files with full.names=TRUE &amp; recursive=TRUE</i>
------------	---

---

**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	<i>A helper function for creating a directory (recursively)</i>
----------	---

---

**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

---

mlr_boruta_filter	<i>Custom mlr filter for Boruta</i>
-------------------	-------------------------------------

---

**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	<i>Version of getNestedTuneResultsOptPathDf that actually works</i>
-----------------------------------	---

---

**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

---

m1_tax_HFE	<i>Hierachical Feature Selection</i>
------------	--------------------------------------

---

**Description**

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

**Usage**

```
m1_tax_HFE(brk, tax_level, corr_cutoff = 0.7, threads = 2, quiet = TRUE)
```

**Arguments**

- brk                      data.table generated by read\_bracken()
- tax\_level                which taxonmoic level to use?
- corr\_cutoff              features with >cutoff will be filtered to just one

**Value**

data.table of filtered features

---

overlap	<i>Determine counts of setdiff, intersect, &amp; union of 2 vectors (or data.tables)</i>
---------	--

---

**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")  
)
```

**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	(deprecated) "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	<i>Global change of plot size options</i>
--------	---

---

**Description**

This is most useful for working with IRkernel 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	<i>splitting path and returning just one item in the vector</i>
----------------	---

---

**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	<i>pipeline sessionInfo</i>
--------------	-----------------------------

---

**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	<i>plot figure and save the figure grob object to a file at the same time</i>
------	---

---

**Description**

This is most useful for working with IRkernel 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
path	Path to write to
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	<i>Simple function for serializing a distance matrix or list of distance matrices</i>
-----------	---

---

**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	<i>Read the last N lines of a file</i>
---------------	--

---

**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"),
  ...
)
```

**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	<i>Dump an R object as text to a temp file and get the md5sum of the file</i>
-------------	---

---

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	<i>rowMeans that works inside a dplyr::mutate() call</i>
-----------	--

---

**Description**

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

**Usage**

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

---

row_sums	<i>rowSums that works inside a dplyr::mutate() call</i>
----------	---

---

**Description**

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

**Usage**

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

---

send_email	<i>A helper function to send an email via the mail bash cmd</i>
------------	---

---

**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	<i>Returns the sizes of R objects</i>
--------------	---------------------------------------

---

**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	<i>snakemake conda info</i>
---------------	-----------------------------

---

**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	<i>python's os.path.split() for R</i>
------------	---------------------------------------

---

**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	<i>Summary for numeric vectors that includes sd and stderr</i>
-----------	--

---

**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

---

taxonomy_levels	<i>A simple function that returns a vector of taxonomy levels</i>
-----------------	---

---

### Description

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

### Usage

```
taxonomy_levels()
```

---

tidy_pcoa	<i>PCoA on a 'long' (tidy) tibble, and a long tibble is returned</i>
-----------	--

---

### 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

<code>df</code>	data.frame or tibble
<code>taxon_col</code>	the column specifying taxa or OTUs (no quotes needed)
<code>sample_col</code>	the column specifying sample names (no quotes needed)
<code>abundance_col</code>	the column specifying the taxon abundances in each sample (no quotes needed)
<code>dists</code>	vector of beta-diversity distances ('wunifrac' = weighted UniFrac, 'unifrac' = unweighted Unifrac; see <code>vegan::vegdist</code> for others)



tree	phylogeny for UniFrac calculations. It can have more tips than 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"

---

unique_n	<i>Pretty print number of unique elements in a vector</i>
----------	---

---

### Description

The result will be cat'ed to the screen. tidytable compatible. Major

### 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	<i>Convert between wellID and column-num</i>
------------	--

---

**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	<i>writing table convience function</i>
-------------	---

---

**Description**

This is most useful for working with IRkernel 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

.HFE, [3](#)  
.well1384\_index, [3](#)  
.well196\_index, [3](#)  
  
as.Num, [4](#)  
  
bash\_job, [4](#)  
  
calc\_alpha\_div, [5](#)  
calc\_beta\_div, [6](#)  
calc\_PCoA, [7](#)  
calculate\_rarefaction\_curves, [5](#)  
cat\_file, [7](#)  
clustermq\_get\_logs, [8](#)  
clustermq\_logfile, [8](#)  
clustermq\_setup, [9](#)  
condaInfo, [10](#)  
  
df.dims, [10](#)  
dfhead, [10](#)  
dist\_format, [11](#)  
  
estimate\_rarified\_richness, [11](#)  
estimate\_richness\_phy, [12](#)  
extract\_pltdt, [12](#)  
  
fig\_uuid, [13](#)  
files\_to\_list, [13](#)  
Fread, [14](#)  
  
hello, [14](#)  
  
itol\_boxplot, [15](#)  
itol\_colorstrip, [15](#), [17–19](#)  
itol\_externalshape, [16](#)  
itol\_heatmap, [17](#)  
itol\_multibar, [18](#)  
itol\_simplebar, [18](#)  
itol\_symbol, [19](#)  
  
list\_files, [20](#)  
  
make\_dir, [20](#)  
ml\_tax\_HFE, [22](#)  
mlr\_boruta\_filter, [21](#)  
mlr\_getNestedTuneResultsOptPathDf, [21](#)  
  
overlap, [22](#)  
  
p.dims, [23](#)  
path\_get\_label, [23](#)  
phyloseq2df, [24](#)  
phyloseq\_rel\_abund, [24](#)  
pipelineInfo, [25](#)  
Plot, [25](#)  
  
qsave\_obj, [26](#)  
  
read\_bracken, [27](#)  
read\_eggnog\_mapper, [27](#)  
readLinesTail, [26](#)  
Robj\_md5sum, [28](#)  
row\_means, [29](#)  
row\_sums, [29](#)  
  
send\_email, [29](#)  
size\_objects, [30](#)  
snakemakeInfo, [30](#)  
split\_path, [31](#)  
summary\_x, [31](#)  
  
taxonomy\_levels, [32](#)  
tidy\_pcoa, [32](#)  
  
unique\_n, [33](#)  
  
well2index, [34](#)  
write\_table, [34](#)