

# Package ‘HTSSIP’

May 23, 2017

**Type** Package

**Title** High Throughput Sequencing of Stable Isotope Probing Data Analysis

**Version** 1.1.1

**Maintainer** Nicholas Youngblut <nyoungb2@gmail.com>

**Description** Functions for analyzing high throughput sequencing stable isotope probing (HTS-SIP) data.  
Analyses include high resolution stable isotope probing (HR-SIP), multi-window high resolution stable isotope probing (MW-HR-SIP), and quantitative stable isotope probing (q-SIP).

**License** GPL-2 | file LICENSE

**LazyData** TRUE

**Depends** R (>= 3.1.2)

**Imports** magrittr (>= 1.5), stringr (>= 1.0.0), plyr (>= 1.8.4), dplyr (>= 0.5.0), tidyr (>= 0.5.1), ggplot2 (>= 2.1.0), DESeq2 (>= 1.10.1), phyloseq (>= 1.14.0), coenocliner (>= 0.2.2), lazyeval(>= 0.2.0)

**RoxygenNote** 6.0.1

**Suggests** testthat, knitr, rmarkdown, doParallel

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Nicholas Youngblut [aut, cre],  
Samuel Barnett [ctb]

**Repository** CRAN

**Date/Publication** 2017-05-23 21:58:52 UTC

## R topics documented:

|                            |   |
|----------------------------|---|
| as.Num . . . . .           | 3 |
| BD_shift . . . . .         | 3 |
| calc_atom_excess . . . . . | 4 |

|                                   |    |
|-----------------------------------|----|
| calc_Gi . . . . .                 | 5  |
| calc_Mheavymax . . . . .          | 5  |
| data-physeq_rep3 . . . . .        | 6  |
| data-physeq_rep3_qPCR . . . . .   | 6  |
| data-physeq_S2D1 . . . . .        | 6  |
| data-physeq_S2D1_1 . . . . .      | 7  |
| data-physeq_S2D2 . . . . .        | 7  |
| data-physeq_S2D2_1 . . . . .      | 7  |
| delta_BD . . . . .                | 8  |
| DESeq2_l2fc . . . . .             | 9  |
| evaluate_matches . . . . .        | 10 |
| expr_param_extract . . . . .      | 10 |
| extract_expressions . . . . .     | 11 |
| extract_formats . . . . .         | 12 |
| filter_l2fc . . . . .             | 12 |
| format_metadata . . . . .         | 13 |
| fraction_overlap . . . . .        | 13 |
| get_treatment_params . . . . .    | 14 |
| gradient_sim . . . . .            | 15 |
| HRSIP . . . . .                   | 16 |
| HTSSIP . . . . .                  | 17 |
| HTSSIP_sim . . . . .              | 18 |
| match_brace . . . . .             | 19 |
| match_placeholders . . . . .      | 19 |
| max_BD_range . . . . .            | 20 |
| OTU_qPCR_trans . . . . .          | 20 |
| overlap_wmean_dist . . . . .      | 21 |
| parse_dist . . . . .              | 22 |
| perc_overlap . . . . .            | 22 |
| phyloseq2df . . . . .             | 23 |
| phyloseq2table . . . . .          | 24 |
| phyloseq_list_ord_dfs . . . . .   | 25 |
| phyloseq_ord_plot . . . . .       | 26 |
| phyloseq_subset . . . . .         | 27 |
| physeq_format . . . . .           | 27 |
| physeq_list_betaDiv . . . . .     | 28 |
| physeq_list_ord . . . . .         | 29 |
| qPCR_sim . . . . .                | 30 |
| qSIP_atom_excess . . . . .        | 31 |
| qSIP_atom_excess_format . . . . . | 32 |
| qSIP_bootstrap . . . . .          | 32 |
| SIP_betaDiv_ord . . . . .         | 33 |
| stringterpolate . . . . .         | 34 |
| tss . . . . .                     | 35 |

---

|        |                              |
|--------|------------------------------|
| as.Num | <i>conversion to numeric</i> |
|--------|------------------------------|

---

**Description**

Conducts conversion: as.character → as.numeric

**Usage**

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

**Arguments**

|   |              |
|---|--------------|
| x | single value |
|---|--------------|

**Value**

numeric

---

|          |   |
|----------|---|
| BD_shift | <i>Assessing the magnitude of BD shifts with 16S rRNA community data by calculating the beta diversity between unlabeled control and labeled treatment gradient fraction communities.</i> |
|----------|---|

---

**Description**

This function is meant to compare 16S rRNA sequence communities of gradient fractions from 2 gradients: a labeled treatment (eg., 13C-labeled DNA) and its corresponding unlabeled control. First, the beta-diversity (e.g, weighted-Unifrac) is calculated pairwise between fraction communities.

**Usage**

```
BD_shift(physeq, method = "unifrac", weighted = TRUE, fast = TRUE,
  normalized = FALSE, parallel = FALSE)
```

**Arguments**

|            |   |
|------------|---|
| physeq     | phyloseq object                           |
| method     | See phyloseq::distance                    |
| weighted   | Weighted Unifrac (if calculating Unifrac) |
| fast       | Fast calculation method                   |
| normalized | Normalized abundances                     |
| parallel   | Calculate in parallel                     |

## Details

The sample\_data table of the user-provided phyloseq object MUST contain the buoyant density (BD) of each sample (a "Buoyant\_density" column in the sample\_data table). The BD information is used to identify overlapping gradient fractions (gradient fractions usually only partially overlap in BD between gradients) between the labeled treatment gradient and the control gradient. Beta diversity between overlapping fractions is calculated. Then, to standardize the values relative to the unlabeled control (1 beta-diversity value for each control gradient fraction), the mean beta diversity of overlapping labeled treatment gradients is calculated for each unlabeled control, and the percent overlap of each labeled treatment fraction is used to weight the mean.

## Value

a data.frame object of weighted mean distances

## Examples

```
data(physeq_S2D2)
## Not run:
# Subsetting phyloseq by Substrate and Day
params = get_treatment_params(physeq_S2D2, c('Substrate', 'Day'))
params = dplyr::filter(params, Substrate!='12C-Con')
ex = "(Substrate=='12C-Con' & Day=='${Day}') | (Substrate=='${Substrate}' & Day == '${Day}')"
physeq_S2D2_l = phyloseq_subset(physeq_S2D2, params, ex)

# Calculating BD_shift on 1 subset (use lapply function to process full list)
wmean1 = BD_shift(physeq_S2D2_l[[1]])

ggplot(wmean1, aes(BD_min.x, wmean_dist)) +
  geom_point()

# Calculating BD_shift on all subsets
lapply(physeq_S2D2_l, BD_shift)

## End(Not run)
```

---

|                  |                                       |
|------------------|---------------------------------------|
| calc_atom_excess | <i>Calculate atom fraction excess</i> |
|------------------|---------------------------------------|

---

## Description

See Hungate et al., 2015 for more details

## Usage

```
calc_atom_excess(Mlab, Mlight, Mheavymax, isotope = "13C")
```

**Arguments**

|           |  |
|-----------|--|
| Mlab      | The molecular wight of labeled DNA                             |
| Mlight    | The molecular wight of unlabeled DNA                           |
| Mheavymax | The theoretical maximum molecular weight of fully-labeled DNA  |
| isotope   | The isotope for which the DNA is labeled with ('13C' or '18O') |

**Value**

numeric value: atom fraction excess (A)

---

|         |   |
|---------|---|
| calc_Gi | <i>Calculate G+C from unlabeled buoyant density</i> |
|---------|---|

---

**Description**

See Hungate et al., 2015 for more details

**Usage**

```
calc_Gi(Wlight)
```

**Arguments**

|        |  |
|--------|--|
| Wlight | A vector with >=1 weighted mean BD from 'light' gradient fractions |
|--------|--|

**Value**

numeric value (fractional G+C; Gi)

---

|                |  |
|----------------|--|
| calc_Mheavymax | <i>Calculate the theoretical maximum molecular weight of fully-labeled DNA</i> |
|----------------|--|

---

**Description**

See Hungate et al., 2015 for more details

**Usage**

```
calc_Mheavymax(Mlight, isotope = "13C", Gi = NA)
```

**Arguments**

|         |  |
|---------|--|
| Mlight  | The molecular wight of unlabeled DNA                           |
| isotope | The isotope for which the DNA is labeled with ('13C' or '18O') |
| Gi      | The G+C content of unlabeled DNA                               |

Value

numeric value: maximum molecular weight of fully-labeled DNA

---

|                  |                                    |
|------------------|------------------------------------|
| data-physeq_rep3 | (Data) A simulated HTS-SIP dataset |
|------------------|------------------------------------|

---

Description

- 6 gradients:
- \*12C-control, replicate 1
  - \*12C-control, replicate 2
  - \*12C-control, replicate 3
  - \*13C-treatment, replicate 1
  - \*13C-treatment, replicate 2
  - \*13C-treatment, replicate 3

---

|                       |  |
|-----------------------|--|
| data-physeq_rep3_qPCR | (Data) qPCR data associated with the physeq_rep3 HTS-SIP dataset |
|-----------------------|--|

---

Description

The dataset contains simulated 16S rRNA copies for each gradient fraction of each gradient (measured via qPCR).

See Also

physeq\_rep3

---

|                  |  |
|------------------|--|
| data-physeq_S2D1 | (Data) A subset of full HTS-SIP dataset (Substrates=2, Days=1) |
|------------------|--|

---

Description

1 of the 2 'substrates' is the 12C-control; the other is the 13C-labeled substrate

---

|                    |  |
|--------------------|--|
| data-physeq_S2D1_1 | (Data) A subset of full HTS-SIP dataset (Substrates=2, Days=1) |
|--------------------|--|

---

**Description**

1 of the 2 'substrates' is the 12C-control; the other is the 13C-labeled substrate.

**Details**

The dataset has been parsed into a list of corresponding 13C-treatment vs 12C-control comparisons. Each comparison is of all gradient fraction samples of the treatment vs the gradient fraction samples of the control.

---

|                  |  |
|------------------|--|
| data-physeq_S2D2 | (Data) A subset of full HTS-SIP dataset (Substrates=2, Days=2) |
|------------------|--|

---

**Description**

1 of the 2 'substrates' is the 12C-control; the other is the 13C-labeled substrate.

---

|                    |  |
|--------------------|--|
| data-physeq_S2D2_1 | (Data) A subset of full HTS-SIP dataset (Substrates=2, Days=2) |
|--------------------|--|

---

**Description**

1 of the 2 'substrates' is the 12C-control; the other is the 13C-labeled substrate.

**Details**

The dataset has been parsed into a list of corresponding 13C-treatment vs 12C-control comparisons. Each comparison is of all gradient fraction samples of the treatment vs the gradient fraction samples of the control.

---

|          |                             |
|----------|-----------------------------|
| delta_BD | <i>delta_BD calculation</i> |
|----------|-----------------------------|

---

### Description

Calculate delta\_BD as described in [Pepe-Ranney et al., 2016](#).

### Usage

```
delta_BD(physeq, control_expr, n = 20, BD_min = NULL, BD_max = NULL)
```

### Arguments

|              |  |
|--------------|--|
| physeq       | Phyloseq object  |
| control_expr | An expression for identifying unlabeled control samples in the phyloseq object (eg., "Substrate=='12C-Con'")   |
| n            | How many evenly-spaced buoyant density values to use for linear interpolation of abundances.   |
| BD_min       | The minimum BD value of the BD range used for OTU abundance interpolation. If NULL, then BD_min will be the minimum of all BD values in the phyloseq object. |
| BD_max       | The maximum BD value of the BD range used for OTU abundance interpolation. If NULL, then BD_max will be the maximum of all BD values in the phyloseq object. |

### Details

Basically, the abundance of each OTU is interpolated at specific BD values in order to have abundance values at consistent points across gradients (gradient fraction BDs normally vary from gradient to gradient). The center of mass (CM) is calculated from these interpolated values, which is the weighted mean BD with interpolated OTU abundances used as weights (ie., where in the density gradient contains the 'center' of the OTU abundance distribution). Delta\_BD is then calculated by subtracting the CM for the unlabeled control gradient from the labeled treatment gradient.

The delta\_BD calculation will be a comparison between unlabeled control and labeled treatment samples. These samples are distinguished from each other with the 'control\_expr' parameter. NOTE: if multiple gradients fall into the control or treatment category, they will be treated as one gradient (which may be OK if you want to combine replicate gradients).

NaN values may occur due low abundances.

The BD range used for interpolation is set by the min/max of all buoyant density values in the phyloseq object (standardize across).

### Value

data.frame with delta\_BD values for each OTU. 'CM' stands for 'center of mass'.

**Examples**

```

data(physeq_S2D2_1)
# just selecting 1 treatment-control comparison
physeq = physeq_S2D2_1[[1]]

## Not run:
# calculating delta_BD
df = delta_BD(physeq, control_expr='Substrate=="12C-Con"')
head(df)

# In this example, the replicate gradients will be combined for treatments/controls
data(physeq_rep3)
df = delta_BD(physeq_rep3, control_expr='Treatment=="12C-Con"')
head(df)

## End(Not run)

```

DESeq2\_l2fc

*Calculating log2 fold change for HTS-SIP data.***Description**

The phyloseq object will be filtered to 1) just OTUs that pass the sparsity cutoff 2) just samples in the user-defined 'heavy' fractions. The log2 fold change (l2fc) is calculated between labeled treatment and control gradients.

**Usage**

```

DESeq2_l2fc(physeq, density_min, density_max, design, l2fc_threshold = 0.25,
  sparsity_threshold = 0.25, sparsity_apply = "all")

```

**Arguments**

|                    |  |
|--------------------|--|
| physeq             | Phyloseq object  |
| density_min        | Minimum buoyant density of the 'heavy' gradient fractions  |
| density_max        | Maximum buoyant density of the 'heavy' gradient fractions  |
| design             | design parameter used for DESeq2 analysis. See DESeq2::DESeq for more details.   |
| l2fc_threshold     | log2 fold change (l2fc) values must be significantly above this threshold in order to reject the hypothesis of equal counts.   |
| sparsity_threshold | All OTUs observed in less than this portion (fraction: 0-1) of gradient fraction samples are pruned. A form of independent filtering. The sparsity cutoff with the most rejected hypotheses is used. |
| sparsity_apply     | Apply sparsity threshold to all gradient fraction samples ('all') or just heavy fraction samples ('heavy')   |

**Value**

dataframe of HRSIP results

**Examples**

```
data(physeq_S2D2)
## Not run:
df_l2fc = DESeq2_l2fc(physeq_S2D2, density_min=1.71, density_max=1.75, design=~Substrate)
head(df_l2fc)

## End(Not run)
```

---

|                  |  |
|------------------|--|
| evaluate_matches | <i>Evaluate String Interpolation Matches</i> |
|------------------|--|

---

**Description**

The expression part of string interpolation matches are evaluated in a specified environment and formatted for replacement in the original string.

**Usage**

```
evaluate_matches(matches, env)
```

**Arguments**

- matches            Match data
- env                The environment in which to evaluate the expressions.

**Value**

A character vector of replacement strings.

---

|                    |  |
|--------------------|--|
| expr_param_extract | <i>Extract all quoted values in the expression used for phyloseq subsetting.</i> |
|--------------------|--|

---

**Description**

This can be useful for creating custom (shorter) labels for each subset relative to using the entire subsetting expression.

**Usage**

```
expr_param_extract(ex, collapse = NULL)
```

**Arguments**

|          |  |
|----------|--|
| ex       | Expression for subsetting the phyloseq object    |
| collapse | Similar to the collapse parameter in base::paste |

**Value**

If `length(ex) == 1`, then a vector of quoted values in the input string. If `length(ex) > 1`, then a matrix or list of quote values, 1 column/index per input string.

**Examples**

```
ex = '(Substrate=="12C-Con" & Day=="14")'
expr_param_extract(ex)

ex = '(Substrate=="12C-Con" & Day=="14") | (Substrate=="13C-Cel" & Day == "14")'
expr_param_extract(ex)

# returns a matrix
ex = c('(Substrate=="12C-Con" & Day=="14")',
      '(Substrate=="13C-Cel" & Day == "14")')
expr_param_extract(ex)

# returns a list
ex = c('(Substrate=="12C-Con" & Day=="14")',
      '(Substrate=="13C-Cel" & Day == "14")',
      '(Substrate=="13C-Cel")')
expr_param_extract(ex)
```

---

|                     |   |
|---------------------|---|
| extract_expressions | <i>Extract Expression Objects from String Interpolation Matches</i> |
|---------------------|---|

---

**Description**

An interpolation match object will contain both its wrapping `${ }` part and possibly a format. This extracts the expression parts and parses them to prepare them for evaluation.

**Usage**

```
extract_expressions(matches)
```

**Arguments**

|         |            |
|---------|------------|
| matches | Match data |
|---------|------------|

**Value**

list of R expressions

---

|                              |   |
|------------------------------|---|
| <code>extract_formats</code> | <i>Extract String Interpolation Formats from Matched Placeholders</i> |
|------------------------------|---|

---

**Description**

An expression placeholder for string interpolation may optionally contain a format valid for `sprintf`. This function will extract such or default to "s" the format for strings.

**Usage**

```
extract_formats(matches)
```

**Arguments**

`matches` Match data

**Value**

A character vector of format specifiers.

---

|                          |                          |
|--------------------------|--------------------------|
| <code>filter_l2fc</code> | <i>Filter l2fc table</i> |
|--------------------------|--------------------------|

---

**Description**

`filter_l2fc` filters a l2fc table to 'best' sparsity cutoffs & bouyant density windows.

**Usage**

```
filter_l2fc(df_l2fc, padj_cutoff = 0.1)
```

**Arguments**

`df_l2fc` data.frame of log2 fold change values  
`padj_cutoff` Adjusted p-value cutoff for rejecting the null hypothesis that l2fc values were not greater than the l2fc\_threshold.

**Value**

filtered df\_l2fc object

---

|                 |  |
|-----------------|--|
| format_metadata | <i>Format phyloseq metadata for calculating BD range overlaps.</i> |
|-----------------|--|

---

**Description**

Format phyloseq metadata for calculating BD range overlaps.

**Usage**

```
format_metadata(physeq, ex = "Substrate=='12C-Con'")
```

**Arguments**

|        |   |
|--------|---|
| physeq | Phyloseq object   |
| ex     | Expression for selecting the control samples to compare to the non-control samples. |

**Value**

a data.frame object of formatted metadata

**Examples**

```
## Not run:
data(physeq_S2D1)
ex = "Substrate=='12C-Con'"
metadata = format_metadata(physeq_S2D1, ex)

## End(Not run)
```

---

|                  |   |
|------------------|---|
| fraction_overlap | <i>Calculate the BD range overlap of gradient fractions</i> |
|------------------|---|

---

**Description**

Calculate the BD range overlap of gradient fractions

**Usage**

```
fraction_overlap(metadata)
```

**Arguments**

|          |  |
|----------|--|
| metadata | Metadata data.frame object. See format_metadata(). |
|----------|--|

**Value**

a data.frame object of metadata with fraction BD overlaps

**Examples**

```
## Not run:
data(physeq_S2D2)
ex = "Substrate=='12C-Con'"
metadata = format_metadata(physeq_S2D2, ex)
m = fraction_overlap(metadata)
head(m)

## End(Not run)
```

---

get\_treatment\_params    *Get parameters for subsetting the phyloseq dataset*

---

**Description**

This function is needed if you want to make multiple subsets of the phyloseq object in order to make specific comparisons between isotopically labeled-treatments and

**Usage**

```
get_treatment_params(physeq, exp_params, treatment = NULL)
```

**Arguments**

|            |   |
|------------|---|
| physeq     | Phyloseq object   |
| exp_params | a vector listing the columns in the phyloseq sample_data table that can subset the phyloseq dataset in order to make the specific labeled-treatment vs labeled-control comparisons that you would like to make. |
| treatment  | This is an expression used to filter out the control-specific parameters (if needed).   |

**Details**

their corresponding controls (eg., from the same time point).

Makes a data.frame of all of the parameter values that differ among the treatment-control comparisons.

For example, if you want to compare the gradient fractions from each labeled-treatment to its corresponding unlabeled-Control (both from the same time point).

**Examples**

```
data(physeq_S2D2)
# Here, the treatment/controls (12C & 13C) are listed in substrate,
# and should be matched by 'Day'. The 13C-treatments can be identified by
# the expression: "Substrate != '12C-Con'"
get_treatment_params(physeq_S2D2, c('Substrate', 'Day'), "Substrate != '12C-Con'")
```

gradient\_sim

*Simulate HTS-SIP communities for 1 density gradient***Description**

Simulate HTS-SIP communities for 1 density gradient

**Usage**

```
gradient_sim(locs, params, responseModel = "gaussian",
             countModel = "poisson", ...)
```

**Arguments**

|               |   |
|---------------|---|
| locs          | Buoyant densities of each gradient fraction   |
| params        | A matrix of parameters for <code>coenocliner::coenocline()</code> . See that function's documentation for more details. |
| responseModel | See <code>coenocliner::coenocline()</code>  |
| countModel    | See <code>coenocliner::coenocline()</code>  |
| ...           | Other parameters passed to <code>coenocliner::coenocline()</code>   |

**Value**

A data.frame of OTU counts.

**Examples**

```
# setting parameters
set.seed(2)
M = 10                                # number of species (OTUs)
ming = 1.67                           # gradient minimum...
maxg = 1.78                           # ...and maximum
nfrac = 24                            # number of gradient fractions
locs = seq(ming, maxg, length=nfrac)  # gradient fraction BD values
tol = rep(0.005, M)                  # species tolerances
h = ceiling(rlnorm(M, meanlog=11))    # max abundances
opt = rnorm(M, mean=1.7, sd=0.005)    # species optima
params = cbind(opt=opt, tol=tol, h=h)  # put in a matrix
# simulate the OTU abundances
```

```
df_OTU = gradient_sim(locs, params)
head(df_OTU)
```

HRSIP

*(MW-)HR-SIP analysis*

## Description

Conduct (multi-window) high resolution stable isotope probing (HR-SIP) analysis.

## Usage

```
HRSIP(physeq, design, density_windows = data.frame(density_min = c(1.7),
  density_max = c(1.75)), sparsity_threshold = seq(0, 0.3, 0.1),
  sparsity_apply = "all", l2fc_threshold = 0.25, padj_method = "BH",
  padj_cutoff = NULL, parallel = FALSE)
```

## Arguments

|                    |   |
|--------------------|---|
| physeq             | Phyloseq object   |
| design             | design parameter used for DESeq2 analysis. This is usually used to differentiate labeled-treatment and unlabeled-control samples. See DESeq2::DESeq for more details on the option.   |
| density_windows    | The buoyant density window(s) used for calculating log2 fold change values. Input can be a vector (length 2) or a data.frame with a 'density_min' and a 'density_max' column (each row designates a density window).  |
| sparsity_threshold | All OTUs observed in less than this portion (fraction: 0-1) of gradient fraction samples are pruned. This is a form of independent filtering. The sparsity cutoff with the most rejected hypotheses is used.  |
| sparsity_apply     | Apply sparsity threshold to all gradient fraction samples ('all') or just 'heavy' fraction samples ('heavy'), where 'heavy' samples are designated by the density_windows.  |
| l2fc_threshold     | log2 fold change (l2fc) values must be significantly above this threshold in order to reject the hypothesis of equal counts. See DESeq2 for more information.   |
| padj_method        | Method for global p-value adjustment (See p.adjust()).  |
| padj_cutoff        | Adjusted p-value cutoff for rejecting the null hypothesis that l2fc values were not greater than the l2fc_threshold. Set to NULL to skip filtering of results to the sparsity cutoff with most rejected hypotheses and filtering each OTU to the buoyant density window with the greatest log2 fold change. |
| parallel           | Process each parameter combination in parallel. See plyr::mdply() for more information.   |

**Details**

The (MW-)HR-SIP workflow is as follows:

1. For each sparsity threshold & BD window: calculate log2 fold change values (with DESeq2) for each OTU
2. Globally adjust p-values with a user-defined method (see p.adjust())
3. Select the sparsity cutoff with the most rejected hypotheses
4. For each OTU, select the BD window with the greatest log2 fold change value

**Value**

dataframe of HRSIP results

**Examples**

```
data(physeq_S2D2_1)

## Not run:
# HR-SIP on just 1 treatment-control comparison
## 1st item in list of phyloseq objects
physeq = physeq_S2D2_1[[1]]
## HR-SIP
#### Note: treatment-control samples differentiated with 'design=~Substrate'
df_l2fc = HRSIP(physeq, design=~Substrate)
head(df_l2fc)

## Same, but multiple BD windows (MW-HR-SIP) & run in parallel
#### Windows = 1.7-1.73 & 1.72-1.75
doParallel::registerDoParallel(2)
dw = data.frame(density_min=c(1.7, 1.72), density_max=c(1.73, 1.75))
df_l2fc = HRSIP(physeq_S2D1_1[[1]],
                design=~Substrate,
                density_windows=dw,
                parallel=TRUE)
head(df_l2fc)

## End(Not run)
```

---

HTSSIP

---

*HTSSIP: analyzing high throughput sequence data from nucleotide  
stable isotope probing experiments*


---

**Description**

HTSSIP provides a toolset for reproducibly analyzing HTS-SIP data. HTS-SIP data is the combination of nucleotide stable isotope probing (DNA- & RNA-SIP) and high throughput sequence data (e.g., MiSeq of 16S rRNA amplicons).

## Details

To learn more about HTSSIP, start with the vignettes: `browseVignettes(package = "HTSSIP")`

---

|            |                                   |
|------------|-----------------------------------|
| HTSSIP_sim | <i>Simulate a HTS-SIP dataset</i> |
|------------|-----------------------------------|

---

## Description

This is a simple method for simulating high throughput sequencing stable isotope probing datasets and is mainly used for package testing purposes. See SIPSim for more detailed and simulation pipeline.

## Usage

```
HTSSIP_sim(locs, params, responseModel = "gaussian", countModel = "poisson",
  meta = NULL, parallel = FALSE, ...)
```

## Arguments

|                            |   |
|----------------------------|---|
| <code>locs</code>          | Buoyant densities of each gradient fraction   |
| <code>params</code>        | A matrix of parameters for <code>coenocliner::coenocline()</code> . See that function's documentation for more details.   |
| <code>responseModel</code> | See <code>coenocliner::coenocline()</code>  |
| <code>countModel</code>    | See <code>coenocliner::coenocline()</code>  |
| <code>meta</code>          | Data.frame object of metadata to add to <code>sample_data</code> table. The data.frame object must have a 'Gradient' column, which is used for joining with <code>dplyr::left_join()</code> . |
| <code>parallel</code>      | Parallel processing. See <code>.parallel</code> option in <code>dplyr::mdply()</code> for more details.   |
| <code>...</code>           | Other parameters passed to <code>coenocliner::coenocline()</code>   |

## Value

A phyloseq object

## Examples

```
# setting parameters for tests
set.seed(2)
M = 10                                # number of species
ming = 1.67                           # gradient minimum...
maxg = 1.78                           # ...and maximum
nfrac = 24                            # number of gradient fractions
locs = seq(ming, maxg, length=nfrac)  # gradient locations
tol = rep(0.005, M)                  # species tolerances
h = ceiling(rlnorm(M, meanlog=11))    # max abundances
## creating parameter matrices for each density gradient
opt1 = rnorm(M, mean=1.7, sd=0.005)   # species optima
```

```

params1 = cbind(opt=opt1, tol=tol, h=h) # put in a matrix
opt2 = rnorm(M, mean=1.7, sd=0.005)    # species optima
params2 = cbind(opt=opt2, tol=tol, h=h) # put in a matrix
param_l = list(
  '12C-Con_rep1' = params1,
  '13C-Ce1_rep1' = params2
)
## Not run:
# simulating phyloseq object
physeq = HTSSIP_sim(locs, param_l)
physeq

## End(Not run)

```

---

match\_brace

*Utility Function for Matching a Closing Brace*


---

### Description

Given positions of opening and closing braces match\_brace identifies the closing brace matching the first opening brace.

### Usage

```
match_brace(opening, closing)
```

### Arguments

|         |   |
|---------|---|
| opening | integer: Vector with positions of opening braces. |
| closing | integer: Vector with positions of closing braces. |

### Value

Integer with the position of the matching brace.

---

match\_placeholders

*Match Expression Placeholders for String Interpolation*


---

### Description

Given a character string a set of expression placeholders are matched. They are of the form `${...}` or optionally `$(f){...}` where `f` is a valid format for `sprintf`.

### Usage

```
match_placeholders(string)
```

Arguments

string                    character: The string to be interpolated.

Value

list containing indices (regex match data) and matches, the string representations of matched expressions.

---

|              |   |
|--------------|---|
| max_BD_range | <i>Adjusting BD range size if negative.</i> |
|--------------|---|

---

Description

If BD (buoyant density) range size is negative, use BD\_to\_set value to set new BD\_max. The BD\_to\_set determines the BD\_max if BD range is negative

Usage

```
max_BD_range(BD_range, BD_min, BD_max, BD_to_set)
```

Arguments

|           |   |
|-----------|---|
| BD_range  | BD range size                           |
| BD_min    | Minimum BD value                        |
| BD_max    | Maximum BD value                        |
| BD_to_set | Value added to BD_min to set new BD_max |

Value

New max BD value

---

|                |  |
|----------------|--|
| OTU_qPCR_trans | <i>Transform OTU counts based on qPCR data</i> |
|----------------|--|

---

Description

OTU counts in the phyloseq otu\_table object will be normalized to sample totals (total sum scaling), then multiplied by the qPCR value associated with each sample. Thus, the qPCR table should have ONE value matching the OTU count table. Value matching between the OTU table & qPCR value table to set by sample\_idx().

Usage

```
OTU_qPCR_trans(physeq, qPCR, sample_idx = "Sample",  
               value_idx = "qPCR_tech_rep_mean")
```

**Arguments**

|            |  |
|------------|--|
| physeq     | A phyloseq object  |
| qPCR       | Either a list or a data.frame of qPCR data. If a list, the list should include a 'summary' tag as is produced from qPCR_sim(). If a data.frame, the table should be formatted as the 'summary' table produced from qPCR_sim(). |
| sample_idx | The qPCR table column index for matching to otu table samples.   |
| value_idx  | The qPCR table column index for qPCR values.   |

**Details**

Note: only the 'summa

**Value**

A phyloseq object with transformed OTU counts

**Examples**

```
# qPCR data simulation
data(physeq_rep3)
data(physeq_rep3_qPCR)
physeq_rep3_t = OTU_qPCR_trans(physeq_rep3, physeq_rep3_qPCR)
```

---

|                    |  |
|--------------------|--|
| overlap_wmean_dist | <i>Calculating weighted mean beta-diversities of overlapping gradient fractions.</i> |
|--------------------|--|

---

**Description**

Calculating weighted mean beta-diversities of overlapping gradient fractions.

**Usage**

```
overlap_wmean_dist(df_dist)
```

**Arguments**

|         |   |
|---------|---|
| df_dist | Filtered distance matrix in data.frame format. See parse_dist() |
|---------|---|

**Value**

a data.frame object of weighted mean distances

---

|            |  |
|------------|--|
| parse_dist | <i>Filtering out non-relevant distances in distance matrix</i> |
|------------|--|

---

**Description**

Filtering out non-relevant distances in distance matrix

**Usage**

```
parse_dist(d)
```

**Arguments**

d                      a distance matrix object

**Value**

a data.frame object of metadata with fraction BD overlaps

**Examples**

```
## Not run:
data(physeq_S2D2)
physeq_S2D2_d = phyloseq::distance(physeq_S2D2,
                                   method='unifrac',
                                   weighted=TRUE,
                                   fast=TRUE,
                                   normalized=FALSE)
physeq_S2D2_d = parse_dist(physeq_S2D2_d)
head(physeq_S2D2_d)

## End(Not run)
```

---

|              |  |
|--------------|--|
| perc_overlap | <i>Calculate the percent overlap between two ranges (x &amp; y).</i> |
|--------------|--|

---

**Description**

The fraction of overlap is relative to Range X (see examples).

**Usage**

```
perc_overlap(x.start, x.end, y.start, y.end)
```

**Arguments**

|         |                             |
|---------|-----------------------------|
| x.start | The start value for Range X |
| x.end   | The end value for Range X   |
| y.start | The start value for Range Y |
| y.end   | The end value for Range Y   |

**Value**

the percent overlap of the ranges

**Examples**

```
## Not run:
x = perc_overlap(0, 1, 0, 0.5)
stopifnot(x == 50)
x = perc_overlap(0, 0.5, 0, 1)
stopifnot(x == 100)

## End(Not run)
```

---

phyloseq2df

*phyloseq data object conversion to data.frame*


---

**Description**

Conducts conversion of 1 of the data objects in a phyloseq object (eg., tax\_table) to a dataframe

**Usage**

```
phyloseq2df(physeq, table_func)
```

**Arguments**

|            |   |
|------------|---|
| physeq     | Phyloseq object                                       |
| table_func | See <code>Phyloseq::phyloseq-class</code> for options |

**Value**

data.frame

**Examples**

```
data(physeq_S2D1)
df_otu = phyloseq2df(physeq_S2D1, table_func=phyloseq::otu_table)
head(df_otu)

df_sample = phyloseq2df(physeq_S2D1, table_func=phyloseq::sample_data)
head(df_sample)
```

---

|                |  |
|----------------|--|
| phyloseq2table | <i>Phyloseq conversion to a ggplot-formatted table</i> |
|----------------|--|

---

**Description**

Convert the OTU table (+ metadata) to a format that can be easily plotted with phyloseq

**Usage**

```
phyloseq2table(physeq, include_sample_data = FALSE, sample_col_keep = NULL,
  include_tax_table = FALSE, tax_col_keep = NULL, control_expr = NULL)
```

**Arguments**

|                     |  |
|---------------------|--|
| physeq              | Phyloseq object  |
| include_sample_data | Include sample_table information?  |
| sample_col_keep     | Which columns in the sample_data table to keep? Use NULL to keep all columns.  |
| include_tax_table   | Include tax_table information?   |
| tax_col_keep        | A vector for column names to keep. Use NULL to keep all columns.   |
| control_expr        | An expression for identifying which samples are controls. Control/non-control identification will be in the 'IS_CONTROL' column of the returned data.frame object. |

**Value**

data.frame

**Examples**

```
data(physeq_S2D1)
# Including some columns from sample metadata
df_OTU = phyloseq2table(physeq_S2D1,
  include_sample_data=TRUE,
  sample_col_keep=c('Buoyant_density', 'Substrate', 'Day'))
head(df_OTU)
```

```
## Not run:
# Including some columns from sample metadata & taxonomy
df_OTU = phyloseq2table(physeq_S2D1,
                        include_sample_data=TRUE,
                        sample_col_keep=c('Buoyant_density', 'Substrate', 'Day'),
                        include_tax_table=TRUE)

head(df_OTU)

## End(Not run)
```

---

phyloseq\_list\_ord\_dfs *Converting ordination objects to data.frames*

---

### Description

For each ordination object in a list, converts to a data.frame for easy plotting with ggplot

### Usage

```
phyloseq_list_ord_dfs(physeq_l, physeq_l_ords, parallel = FALSE)
```

### Arguments

|               |   |
|---------------|---|
| physeq_l      | A list of phyloseq objects  |
| physeq_l_ords | A list of ordination objects  |
| parallel      | Parallel processing. See <code>plyr::adply</code> for more information. |

### Value

List of data.frame objects

### Examples

```
data(physeq_S2D2_l)
## Not run:
# make a list of beta diversity distance matrix objects
physeq_S2D2_l_d = physeq_list_betaDiv(physeq_S2D2_l)
# make a list of ordinations
physeq_S2D2_l_d_ord = physeq_list_ord(physeq_S2D2_l, physeq_S2D2_l_d)
# convert ordination information to data.frame objects
physeq_S2D2_l_d_ord_df = phyloseq_list_ord_dfs(physeq_S2D2_l, physeq_S2D2_l_d_ord)

## End(Not run)
```

---

|                   |   |
|-------------------|---|
| phyloseq_ord_plot | <i>Plotting beta diversity ordination</i> |
|-------------------|---|

---

## Description

For each data.frame object in a list (converted from ordination objects), creates a ggplot figure.

## Usage

```
phyloseq_ord_plot(physeq_ord_df, title = NULL,
  point_size = "Buoyant_density", point_fill = "Substrate",
  point_alpha = 0.5, point_shape = NULL)
```

## Arguments

|               |  |
|---------------|--|
| physeq_ord_df | A list of data.frame objects (see phyloseq_list_ord_dfs)               |
| title         | Plot title   |
| point_size    | The data.frame column determining point size                           |
| point_fill    | The data.frame column determining point fill color                     |
| point_alpha   | The data.frame column (or just a single value) determining point alpha |
| point_shape   | The data.frame column (or just a single value) determining point shape |

## Value

ggplot2 object

## Examples

```
data(physeq_S2D2_1)
## Not run:
# make a list of beta diversity distance matrix objects
physeq_S2D2_1_d = physeq_list_betaDiv(physeq_S2D2_1)
# make a list of ordinations
physeq_S2D2_1_d_ord = physeq_list_ord(physeq_S2D2_1, physeq_S2D2_1_d)
# convert ordination information to data.frame objects
physeq_S2D2_1_d_ord_df = phyloseq_list_ord_dfs(physeq_S2D2_1, physeq_S2D2_1_d_ord)
# make ordination plots with ggplot2
phyloseq_ord_plot(physeq_S2D2_1_d_ord_df)

## End(Not run)
```

---

|                 |   |
|-----------------|---|
| phyloseq_subset | <i>Make a list of phyloseq object subsets</i> |
|-----------------|---|

---

**Description**

Create a list of phyloseq object subsets based on phyloseq sample data parameters (e.g., a phyloseq subset for each treatment)

**Usage**

```
phyloseq_subset(physeq, params, ex)
```

**Arguments**

|        |   |
|--------|---|
| physeq | Phyloseq object                               |
| params | data.frame of parameters supplies to ex       |
| ex     | Expression for subsetting the phyloseq object |

**Value**

A list of Phyloseq objects

**Examples**

```
data(physeq_S2D2)
# making subsets by substrate and time point
params = get_treatment_params(physeq_S2D2, c('Substrate', 'Day'))
# filtering out controls
params = dplyr::filter(params, Substrate!='12C-Con')
# making expression for subsetting labeled-unlabeled gradient comparisons
ex = "(Substrate=='12C-Con' & Day=='${Day}') | (Substrate=='${Substrate}' & Day == '${Day}')"
physeq_l = phyloseq_subset(physeq_S2D2, params, ex)
physeq_l
```

---

|               |  |
|---------------|--|
| physeq_format | <i>Checking format of phyloseq object for HTSSIP compatibility</i> |
|---------------|--|

---

**Description**

Checking format of phyloseq object for HTSSIP compatibility

**Usage**

```
physeq_format(physeq)
```

**Arguments**

physeq            Phyloseq object

**Value**

phyloseq object

**Examples**

```
# this data should be formatted for HTSSIP
data(physeq_S2D2)
physeq_format(physeq_S2D2)

# this data should NOT be correctly formatted for HTSSIP
## Not run:
library(phyloseq)
data(GlobalPatterns)
tryCatch(
  physeq_format(GlobalPatterns),
  function(e) e
)

## End(Not run)
```

---

physeq\_list\_betaDiv    *calculating beta diversity for a list of phyloseq objects*

---

**Description**

For each phyloseq object in a list, calculates beta-diversity between all samples using the phyloseq::distance function.

**Usage**

```
physeq_list_betaDiv(physeq_l, method = "unifrac", weighted = TRUE,
  fast = TRUE, normalized = TRUE, parallel = FALSE)
```

**Arguments**

|            |   |
|------------|---|
| physeq_l   | A list of phyloseq objects                |
| method     | See phyloseq::distance                    |
| weighted   | Weighted Unifrac (if calculating Unifrac) |
| fast       | Fast calculation method                   |
| normalized | Normalized abundances                     |
| parallel   | Calculate in parallel                     |

**Value**

List of dist objects

**Examples**

```
data(physeq_S2D2_1)
## Not run:
physeq_S2D2_1_d = physeq_list_betaDiv(physeq_S2D2_1)

## End(Not run)
```

---

|                 |   |
|-----------------|---|
| physeq_list_ord | <i>calculating ordinations from a list of distance matrices</i> |
|-----------------|---|

---

**Description**

For each dist object in a provided list, the function calculates an ordination with the `phyloseq::ordinate` function.

**Usage**

```
physeq_list_ord(physeq_l, physeq_l_d, ord_method = "NMDS")
```

**Arguments**

|            |                                     |
|------------|-------------------------------------|
| physeq_l   | A list of phyloseq objects          |
| physeq_l_d | A list of dist objects              |
| ord_method | See <code>phyloseq::ordinate</code> |

**Value**

List of ordination objects

**Examples**

```
data(physeq_S2D2_1)
## Not run:
# make a list of beta diversity distance matrix objects
physeq_S2D2_1_d = physeq_list_betaDiv(physeq_S2D2_1)
# make a list of ordinations
physeq_S2D2_1_d_ord = physeq_list_ord(physeq_S2D2_1, physeq_S2D2_1_d)

## End(Not run)
```

qPCR\_sim

*Simulate qPCR values***Description**

qPCR values will be simulated for each sample in the provided phyloseq object. The error distribution for each sample is drawn from a Gaussian distribution, where the mean and standard deviation of the Gaussian distribution are set by user-defined functions. The user-defined functions that take buoyant density as input and returns a numeric value (see examples), which allows the qPCR values to increase in mean & variance at certain buoyant densities.

**Usage**

```
qPCR_sim(physeq, control_mean_fun, control_sd_fun, treat_mean_fun, treat_sd_fun,
         n_tech_rep = 3, control_expr = NULL)
```

**Arguments**

|                  |   |
|------------------|---|
| physeq           | Object of class "phyloseq"  |
| control_mean_fun | Function used for simulating the qPCR normal distribution mean for control samples.                 |
| control_sd_fun   | Function used for simulating the qPCR normal distribution standard deviation for control samples.   |
| treat_mean_fun   | Function used for simulating the qPCR normal distribution mean for treatment samples.               |
| treat_sd_fun     | Function used for simulating the qPCR normal distribution standard deviation for treatment samples. |
| n_tech_rep       | Number of technical replicates.   |
| control_expr     | Expression used to identify control samples based on sample_data.                                   |

**Value**

data.frame of qPCR values

**Examples**

```
# making functions for simulating values
## 'x' will be Buoyant_density as defined in the phyloseq object sample_data
control_mean_fun = function(x) dnorm(x, mean=1.70, sd=0.01) * 1e8
## This will set sd to scale with the mean
control_sd_fun = function(x) control_mean_fun(x) / 3
## This will 'shift' the gene copy distribution to 'heavier' BDs
treat_mean_fun = function(x) dnorm(x, mean=1.75, sd=0.01) * 1e8
treat_sd_fun = function(x) treat_mean_fun(x) / 3
# simulating qPCR values
```

```

df_qPCR = qPCR_sim(physeq_S2D2,
                    control_expr='Substrate=="12C-Con"',
                    control_mean_fun=control_mean_fun,
                    control_sd_fun=control_sd_fun,
                    treat_mean_fun=treat_mean_fun,
                    treat_sd_fun=treat_sd_fun)

# using the Cauchy distribution instead of normal distributions
control_mean_fun = function(x) dcauchy(x, location=1.70, scale=0.01) * 1e8
control_sd_fun = function(x) control_mean_fun(x) / 3
treat_mean_fun = function(x) dcauchy(x, location=1.74, scale=0.01) * 1e8
treat_sd_fun = function(x) treat_mean_fun(x) / 3
# simulating qPCR values
df_qPCR = qPCR_sim(physeq_S2D2,
                    control_expr='Substrate=="12C-Con"',
                    control_mean_fun=control_mean_fun,
                    control_sd_fun=control_sd_fun,
                    treat_mean_fun=treat_mean_fun,
                    treat_sd_fun=treat_sd_fun)

```

---

qSIP\_atom\_excess

---

*Calculate atom fraction excess using q-SIP method*


---

## Description

Calculate atom fraction excess using q-SIP method

## Usage

```

qSIP_atom_excess(physeq, control_expr, treatment_rep = NULL,
                 isotope = "13C", df_OTU_W = NULL)

```

## Arguments

|               |  |
|---------------|--|
| physeq        | A phyloseq object  |
| control_expr  | Expression used to identify control samples based on sample_data.        |
| treatment_rep | Which column in the phyloseq sample data designates replicate treatments |
| isotope       | The isotope for which the DNA is labeled with ('13C' or '18O')           |
| df_OTU_W      | Keep NULL  |

## Value

A list of 2 data.frame objects. 'W' contains the weighted mean buoyant density (W) values for each OTU in each treatment/control. 'A' contains the atom fraction excess values for each OTU. For the 'A' table, the 'Z' column is buoyant density shift, and the 'A' column is atom fraction excess.

Examples

```
# tranforming values
physeq_rep3_t = OTU_qPCR_trans(physeq_rep3, physeq_rep3_qPCR)

# BD shift (Z) & atom excess (A)
atomX = qSIP_atom_excess(physeq_rep3_t,
                        control_expr='Treatment=="12C-Con"',
                        treatment_rep='Replicate')
```

---

|                         |  |
|-------------------------|--|
| qSIP_atom_excess_format | <i>Reformat a phyloseq object of qSIP_atom_excess analysis</i> |
|-------------------------|--|

---

Description

Reformat a phyloseq object of qSIP\_atom\_excess analysis

Usage

```
qSIP_atom_excess_format(physeq, control_expr, treatment_rep)
```

Arguments

- physeq            A phyloseq object
- control\_expr    An expression for identifying unlabeled control samples in the phyloseq object (eg., "Substrate=='12C-Con'")
- treatment\_rep   Which column in the phyloseq sample data designates replicate treatments

Value

numeric value: atom fraction excess (A)

---

|                |   |
|----------------|---|
| qSIP_bootstrap | <i>Calculate bootstrap CI for atom fraction excess using q-SIP method</i> |
|----------------|---|

---

Description

Calculate bootstrap CI for atom fraction excess using q-SIP method

Usage

```
qSIP_bootstrap(atomX, isotope = "13C", n_sample = c(3, 3), n_boot = 10,
               parallel = FALSE, a = 0.1)
```

**Arguments**

|          |   |
|----------|---|
| atomX    | A list object created by <code>qSIP_atom_excess()</code>  |
| isotope  | The isotope for which the DNA is labeled with (' <sup>13</sup> C' or ' <sup>18</sup> O')                                      |
| n_sample | A vector of length 2. The sample size for data resampling (with replacement) for 1) control samples and 2) treatment samples. |
| n_boot   | Number of bootstrap replicates.   |
| parallel | Parallel processing. See <code>.parallel</code> option in <code>dplyr::mdply()</code> for more details.                       |
| a        | A numeric value. The alpha for calculating confidence intervals.  |

**Value**

A `data.frame` of atom fraction excess values (A) and atom fraction excess confidence intervals.

**Examples**

```
# tranforming values
physeq_rep3_t = OTU_qPCR_trans(physeq_rep3, physeq_rep3_qPCR)

## Not run:
# BD shift (Z) & atom excess (A)
atomX = qSIP_atom_excess(physeq_rep3_t,
                        control_expr='Treatment=="12C-Con"',
                        treatment_rep='Replicate')

# bootstrapping in parallel
doParallel::registerDoParallel(2)
df_atomX_boot = qSIP_bootstrap(atomX, parallel=TRUE)
head(df_atomX_boot)

## End(Not run)
```

---

SIP\_betaDiv\_ord

---

*Calculating & plotting beta diversity for a list of phyloseq objects*


---

**Description**

For each phyloseq object in a list, calculates beta-diversity between all samples using the `phyloseq::distance` function.

**Usage**

```
SIP_betaDiv_ord(physeq_l, method = "unifrac", weighted = TRUE,
                fast = TRUE, normalized = TRUE, parallel = FALSE, plot = FALSE)
```

**Arguments**

|            |   |
|------------|---|
| physeq_l   | A list of phyloseq objects                |
| method     | See phyloseq::distance                    |
| weighted   | Weighted Unifrac (if calculating Unifrac) |
| fast       | Fast calculation method                   |
| normalized | Normalized abundances                     |
| parallel   | Calculate in parallel                     |
| plot       | Return a plot (instead of )               |

**Value**

If plot==FALSE, a data.frame object of beta-diversity values. If plot==TRUE, a glob object for plotting.

**Examples**

```
data(physeq_S2D2_1)
## Not run:
physeq_S2D2_1_df = SIP_betaDiv_ord(physeq_S2D2_1)
head(physeq_S2D2_1_df, n=3)

## End(Not run)
```

---

stringterpolate

*String Interpolation*


---

**Description**

String interpolation is a useful way of specifying a character string which depends on values in a certain environment. It allows for string creation which is easier to read and write when compared to using e.g. `paste` or `sprintf`. The (template) string can include expression placeholders of the form `${expression}` or `$(format){expression}`, where expressions are valid R expressions that can be evaluated in the given environment, and format is a format specification valid for use with `sprintf`.

**Usage**

```
stringterpolate(string, env = parent.frame())
```

**Arguments**

|        |   |
|--------|---|
| string | A template character string.                          |
| env    | The environment in which to evaluate the expressions. |

**Value**

An interpolated character string.

---

|     |                          |
|-----|--------------------------|
| tss | <i>Total sum scaling</i> |
|-----|--------------------------|

---

**Description**

Total sum scaling

**Usage**

```
tss(x, MARGIN = 2, na.rm = FALSE)
```

**Arguments**

|        |                                  |
|--------|----------------------------------|
| x      | data.frame of numeric values     |
| MARGIN | table margin (1=rows, 2=columns) |
| na.rm  | remove NAs?                      |

**Value**

data.frame of qPCR values

**Examples**

```
# making functions for simulating values
df = data.frame(1:5, 5:9)
df_t = tss(df)
apply(df_t, 2, sum)
```

# Index

## \*Topic **data**

- data-physeq\_rep3, [6](#)
- data-physeq\_rep3\_qPCR, [6](#)
- data-physeq\_S2D1, [6](#)
- data-physeq\_S2D1\_1, [7](#)
- data-physeq\_S2D2, [7](#)
- data-physeq\_S2D2\_1, [7](#)

as.Num, [3](#)

BD\_shift, [3](#)

calc\_atom\_excess, [4](#)

calc\_Gi, [5](#)

calc\_Mheavymax, [5](#)

data-physeq\_rep3, [6](#)

data-physeq\_rep3\_qPCR, [6](#)

data-physeq\_S2D1, [6](#)

data-physeq\_S2D1\_1, [7](#)

data-physeq\_S2D2, [7](#)

data-physeq\_S2D2\_1, [7](#)

delta\_BD, [8](#)

DESeq2\_l2fc, [9](#)

evaluate\_matches, [10](#)

expr\_param\_extract, [10](#)

extract\_expressions, [11](#)

extract\_formats, [12](#)

filter\_l2fc, [12](#)

format\_metadata, [13](#)

fraction\_overlap, [13](#)

get\_treatment\_params, [14](#)

gradient\_sim, [15](#)

HRSIP, [16](#)

HTSSIP, [17](#)

HTSSIP-package (HTSSIP), [17](#)

HTSSIP\_sim, [18](#)

match\_brace, [19](#)

match\_placeholders, [19](#)

max\_BD\_range, [20](#)

OTU\_qPCR\_trans, [20](#)

overlap\_wmean\_dist, [21](#)

parse\_dist, [22](#)

paste, [34](#)

perc\_overlap, [22](#)

phyloseq2df, [23](#)

phyloseq2table, [24](#)

phyloseq\_list\_ord\_dfs, [25](#)

phyloseq\_ord\_plot, [26](#)

phyloseq\_subset, [27](#)

physeq\_format, [27](#)

physeq\_list\_betaDiv, [28](#)

physeq\_list\_ord, [29](#)

physeq\_rep3 (data-physeq\_rep3), [6](#)

physeq\_rep3\_qPCR  
(data-physeq\_rep3\_qPCR), [6](#)

physeq\_S2D1 (data-physeq\_S2D1), [6](#)

physeq\_S2D1\_1 (data-physeq\_S2D1\_1), [7](#)

physeq\_S2D2 (data-physeq\_S2D2), [7](#)

physeq\_S2D2\_1 (data-physeq\_S2D2\_1), [7](#)

qPCR\_sim, [30](#)

qSIP\_atom\_excess, [31](#)

qSIP\_atom\_excess\_format, [32](#)

qSIP\_bootstrap, [32](#)

SIP\_betaDiv\_ord, [33](#)

sprintf, [12](#), [19](#), [34](#)

stringterpolate, [34](#)

tss, [35](#)