

# Package: MicrobiomeR (via r-universe)

October 5, 2026

**Title** Analyze Microbiome Data

**Version** 0.8.0

**Maintainer** Shaurita Hutchins <shaurita.hutchins@gmail.com>

**License** MIT + file LICENSE

**biocViews** Metagenomics, Microbiome, Sequencing, SystemsBiology

**Description** An R package for microbiome analysis combining functions from phyloseq, metacoder, and microbiome into an easy to use and full featured microbiome analysis package.

**Roxygen** list(markdown = TRUE)

**Depends** R (>= 4.1.0)

**Remotes** ropensci/taxa, grunwaldlab/metacoder, joey711/phyloseq, microbiome/microbiome, jonclayden/shades

**Imports** ape, biomformat, cli, crayon, DT, data.table, diptest, dplyr (>= 0.8.0), forcats, ggplot2, ggpubr, ggthemes, ggrepel, glue, htmlwidgets, htmltools, leaflet, lifecycle, magrittr, metacoder, microbiome (>= 1.5.27), openxlsx, phyloseq, plotly, purrr, rlang, rstudioapi, scales, scico, shades, SparseArray, stringr, taxa, tibble, tidyr, vegan, vctrs, viridis, yaml

**Suggests** covr, knitr, rmarkdown, testthat

**Encoding** UTF-8

**RoxygenNote** 7.3.3

**LazyData** true

**LazyDataCompression** xz

**URL** <https://github.com/vallenderlab/MicrobiomeR>,  
<https://vallenderlab.github.io/MicrobiomeR/>

**BugReports** <https://github.com/vallenderlab/MicrobiomeR/issues>

**VignetteBuilder** knitr

**StagedInstall** false

**Config/pak/sysreqs** libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev make libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev libx11-dev zlib1g-dev

**Repository** <https://sdhutchins.r-universe.dev>

**Date/Publication** 2026-09-05 21:37:14 UTC

**RemoteUrl** <https://github.com/vallenderlab/MicrobiomeR>

**RemoteRef** HEAD

**RemoteSha** 951564731f0e93334d43fc55a54a49b0da9c66c6

## Contents

|                                    |    |
|------------------------------------|----|
| agglomerate_taxmap . . . . .       | 3  |
| alpha_diversity_measures . . . . . | 4  |
| alpha_diversity_plot . . . . .     | 5  |
| as_analyzed_format . . . . .       | 7  |
| as_basic_format . . . . .          | 8  |
| as_custom_format . . . . .         | 9  |
| as_MicrobiomeR_format . . . . .    | 10 |
| as_phyloseq_format . . . . .       | 10 |
| as_raw_format . . . . .            | 11 |
| combination_palette . . . . .      | 12 |
| convert_proportions . . . . .      | 13 |
| correlation_data . . . . .         | 14 |
| correlation_plot . . . . .         | 15 |
| correlation_plots . . . . .        | 16 |
| cov_filter . . . . .               | 17 |
| create_phyloseq . . . . .          | 19 |
| create_pub_table . . . . .         | 21 |
| create_taxmap . . . . .            | 21 |
| get_color_palette . . . . .        | 22 |
| heat_tree_parameters . . . . .     | 24 |
| heat_tree_plots . . . . .          | 25 |
| is_analyzed_format . . . . .       | 26 |
| is_basic_format . . . . .          | 27 |
| is_phyloseq_format . . . . .       | 28 |
| is_raw_format . . . . .            | 29 |
| melt_taxmap . . . . .              | 30 |
| metacoder_comp_func_1 . . . . .    | 30 |
| MicrobiomeR_Formats . . . . .      | 31 |
| MicrobiomeR_Workflow . . . . .     | 32 |
| order_metacoder_data . . . . .     | 32 |
| ordination_plot . . . . .          | 33 |
| otu_id_filter . . . . .            | 34 |
| otu_prevalence_filter . . . . .    | 36 |
| otu_proportion_filter . . . . .    | 38 |
| output_dir . . . . .               | 39 |

|                                         |           |
|-----------------------------------------|-----------|
| parse_taxonomy_silva_128 . . . . .      | 41        |
| permanova . . . . .                     | 42        |
| phyloseq_silva_2 . . . . .              | 43        |
| phyloseq_silva_3 . . . . .              | 44        |
| pick_new_outgroup . . . . .             | 44        |
| plot_limits . . . . .                   | 45        |
| raw_silva_2 . . . . .                   | 46        |
| raw_silva_3 . . . . .                   | 46        |
| root_by_longest_edge . . . . .          | 47        |
| root_phyloseq_tree . . . . .            | 48        |
| sample_id_filter . . . . .              | 49        |
| save_alpha_diversity_plots . . . . .    | 50        |
| save_correlation_plots . . . . .        | 52        |
| save_heat_tree_plots . . . . .          | 53        |
| save_ordination_plots . . . . .         | 54        |
| save_stacked_barplots . . . . .         | 55        |
| save_top_coefficients_barplot . . . . . | 56        |
| scico_palette . . . . .                 | 57        |
| stacked_barplot . . . . .               | 58        |
| stacked_barplots . . . . .              | 59        |
| taxa_prevalence_filter . . . . .        | 59        |
| taxon_id_filter . . . . .               | 61        |
| top_coefficients_barplot . . . . .      | 63        |
| transformer . . . . .                   | 64        |
| transposer . . . . .                    | 66        |
| treatment_matrix . . . . .              | 67        |
| validate_MicrobiomeR_format . . . . .   | 68        |
| viridis_magma_palette . . . . .         | 69        |
| viridis_palette . . . . .               | 70        |
| vlookup . . . . .                       | 71        |
| which_format . . . . .                  | 72        |
| <b>Index</b>                            | <b>74</b> |

---

|                    |                                   |
|--------------------|-----------------------------------|
| agglomerate_taxmap | <i>Agglomerate taxmap Objects</i> |
|--------------------|-----------------------------------|

---

## Description

A function similar to the `phyloseq::tax_glom` function, that assembles abundance data at a specified rank. This removes subtaxa and reassigns the values at the specified rank.

## Usage

```
agglomerate_taxmap(obj, rank, validated = FALSE)
```

**Arguments**

|           |                                                                                           |
|-----------|-------------------------------------------------------------------------------------------|
| obj       | A Taxmap object.                                                                          |
| rank      | The rank that will be agglomerated to.                                                    |
| validated | This parameter provides a way to override validation steps. Use carefully. Default: FALSE |

**Details**

This function helps analyzing taxonomic data at intermediate ranks by agglomerating the observation data and the taxmap object.

**Value**

A taxmap object that has been agglomerated at the specified rank.

**See Also**

[filter\\_taxa](#)

[validate\\_MicrobiomeR\\_format](#)

Other Advanced Metacoder Filters: [cov\\_filter\(\)](#), [otu\\_prevalence\\_filter\(\)](#), [otu\\_proportion\\_filter\(\)](#), [taxa\\_prevalence\\_filter\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  # This example uses data that are no longer available in the MicrobiomeR package,
  # however, they can be easily generated with \link{MicrobiomeR}{as_basic_format}.
  library(MicrobiomeR)
  basic_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "basic_format")
  phylum_obj <- agglomerate_taxmap(obj = basic_silva, rank = "Phylum")
  class_obj <- agglomerate_taxmap(obj = basic_silva, rank = "Class")
  order_obj <- agglomerate_taxmap(obj = basic_silva, rank = "Order")
}

## End(Not run)
```

---

alpha\_diversity\_measures

*Alpha Diversity Measures*

---

**Description**

This function generates various alpha diversity measures include Shannon, Fisher, Coverage, GiniSimpson, and InverseSimpson.

**Usage**

```
alpha_diversity_measures(obj, group = "TreatmentGroup")
```

**Arguments**

|       |                                                                                                                |
|-------|----------------------------------------------------------------------------------------------------------------|
| obj   | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> .                              |
| group | The "TreatmentGroup" or similar grouping from your metadata to denote sample groups, Default: 'TreatmentGroup' |

**Value**

Returns a list of alpha diversity measures with metadata.

**See Also**

View [alpha,meta](#) to learn more about generating diversity measures with the microbiome package.

**Examples**

```
## Not run:
if (interactive()) {
  library(MicrobiomeR)
  data <- analyzed_silva
  measures <- alpha_diversity_measures(data)
  measures$Shannon
}

## End(Not run)
```

---

alpha\_diversity\_plot    *Alpha Diversity Plot*

---

**Description**

Plot the alpha diversity using a violin plot. `alpha_diversity_plots` generates plots for all alpha diversity measures.

**Usage**

```
alpha_diversity_plot(
  obj,
  measure = "Shannon",
  group = "TreatmentGroup",
  select_otu_table = NULL,
  title = NULL
)

alpha_diversity_plots(
```

```
obj,
measure = c("Shannon", "GiniSimpson", "InverseSimpson"),
group = "TreatmentGroup"
)
```

### Arguments

|                  |                                                                                                                                                                 |
|------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obj              | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> .                                                                               |
| measure          | Select an alpha diversity measure such as Shannon, Fisher, Coverage, GiniSimpson, and InverseSimpson, Default: 'Shannon'                                        |
| group            | The "TreatmentGroup" or similar grouping or column from your metadata to denote sample groups, Default: 'TreatmentGroup'                                        |
| select_otu_table | DEPRECATED. Choose an otu table to analyze, Default: 'otu_proportions'                                                                                          |
| title            | The title of the plot, Default: NULL                                                                                                                            |
| measures         | A list of alpha diversity measures such as Shannon, Fisher, Coverage, GiniSimpson, and InverseSimpson, Default: 'c("Shannon", "GiniSimpson", "InverseSimpson")' |

### Details

Alpha diversity helps to determine the species richness (the number of different species in a sample) or evenness (similar abundance level). We prefer to use Shannon as it is better for data generated using the QIIME pipeline.

### Value

Returns an alpha diversity plot.

Returns a melted dataframe.

### See Also

[alpha\\_diversity\\_measures](#), [diversity](#), [ggviolin](#)

Other Visualizations: [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

Other Visualizations: [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

### Examples

```
## Not run:
if (interactive()) {
  library(MicrobiomeR)
```

```
data <- analyzed_silva
plot <- alpha_diversity_plot(obj = data,
                             measure = "Shannon")

plot
}

## End(Not run)
```

---

|                    |                                       |
|--------------------|---------------------------------------|
| as_analyzed_format | <i>As Analyzed MicrobiomeR Format</i> |
|--------------------|---------------------------------------|

---

## Description

Converts a Taxmap object to the analyzed\_format.

## Usage

```
as_analyzed_format(
  obj,
  cols = NULL,
  groups = NULL,
  combinations = NULL,
  out_names = NULL,
  comp_func = metacoder_comp_func_1
)
```

## Arguments

|              |                                                                                    |
|--------------|------------------------------------------------------------------------------------|
| obj          | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> .  |
| cols         | Column names used for <a href="#">calc_taxon_abund</a> . Default: NULL             |
| groups       | Group names used for <a href="#">compare_groups</a> . Default: NULL                |
| combinations | Combinations of treatments used in <a href="#">compare_groups</a> . Default: NULL  |
| out_names    | Column names of the output used for <a href="#">calc_obs_props</a> . Default: NULL |
| comp_func    | A Comparison based function used in <a href="#">compare_groups</a> . Default: NULL |

## Details

See the [MicrobiomeR\\_Formats](#) documentation.

## Value

A Taxmap object in the "analyzed\_format".

**See Also**

[is\\_phyloseq\\_format](#), [is\\_raw\\_format](#), [is\\_basic\\_format](#), [is\\_analyzed\\_format](#), [as\\_raw\\_format](#), [as\\_basic\\_format](#), [order\\_metacoder\\_data](#)

[compare\\_groups](#)

[taxonomy\\_table](#), [taxon\\_ids](#)

Other Formatting: [MicrobiomeR\\_Formats](#), [as\\_MicrobiomeR\\_format\(\)](#), [as\\_basic\\_format\(\)](#), [as\\_custom\\_format\(\)](#), [as\\_phyloseq\\_format\(\)](#), [as\\_raw\\_format\(\)](#), [melt\\_taxmap\(\)](#), [treatment\\_matrix\(\)](#)

---

|                 |                                    |
|-----------------|------------------------------------|
| as_basic_format | <i>As Basic MicrobiomeR Format</i> |
|-----------------|------------------------------------|

---

**Description**

Converts a Taxmap object to the basic\_format.

**Usage**

```
as_basic_format(obj, cols = NULL, out_names = NULL)
```

**Arguments**

|           |                                                                                    |
|-----------|------------------------------------------------------------------------------------|
| obj       | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> .  |
| cols      | Column names used for <a href="#">calc_taxon_abund</a> . Default: NULL             |
| out_names | Column names of the output used for <a href="#">calc_obs_props</a> . Default: NULL |

**Details**

See the [MicrobiomeR\\_Formats](#) documentation.

**Value**

A Taxmap object in the "basic\_format".

**See Also**

[is\\_phyloseq\\_format](#), [is\\_raw\\_format](#), [is\\_basic\\_format](#), [as\\_raw\\_format](#), [order\\_metacoder\\_data](#)  
[calc\\_taxon\\_abund](#), [calc\\_obs\\_props](#)

Other Formatting: [MicrobiomeR\\_Formats](#), [as\\_MicrobiomeR\\_format\(\)](#), [as\\_analyzed\\_format\(\)](#), [as\\_custom\\_format\(\)](#), [as\\_phyloseq\\_format\(\)](#), [as\\_raw\\_format\(\)](#), [melt\\_taxmap\(\)](#), [treatment\\_matrix\(\)](#)

---

|                  |                                     |
|------------------|-------------------------------------|
| as_custom_format | <i>As Custom MicrobiomeR Format</i> |
|------------------|-------------------------------------|

---

## Description

A function for formatting Taxmap objects in the MicrobiomeR format. This function attempts to give more customization than the as\_\*\_format functions.

## Usage

```
as_custom_format(obj, format, change_name_list = NULL, ...)
```

## Arguments

|                  |                                                                                                                                                  |
|------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|
| obj              | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> .                                                                |
| format           | The name of the format to convert the object to.                                                                                                 |
| change_name_list | A list with names of the tables in the observation data, that have matching values that are used to change the names of the table. Default: NULL |
| ...              | An optional list of parameters to use in the as_*_format function specified                                                                      |

## Details

This function is meant to be more helpful for customizing the Taxmap object.

## Value

A Taxmap object that we have tried to format with all of our heart.

## See Also

[create\\_taxmap](#), [which\\_format](#), [order\\_metacoder\\_data](#), [as\\_MicrobiomeR\\_format](#)

Other Formatting: [MicrobiomeR\\_Formats](#), [as\\_MicrobiomeR\\_format\(\)](#), [as\\_analyzed\\_format\(\)](#), [as\\_basic\\_format\(\)](#), [as\\_phyloseq\\_format\(\)](#), [as\\_raw\\_format\(\)](#), [melt\\_taxmap\(\)](#), [treatment\\_matrix\(\)](#)

---

as\_MicrobiomeR\_format *As MicrobiomeR Format*


---

## Description

Converts a Taxmap object to the specified format.

## Usage

```
as_MicrobiomeR_format(obj, format, ...)
```

## Arguments

|        |                                                                                                      |
|--------|------------------------------------------------------------------------------------------------------|
| obj    | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> .                    |
| format | The name of the format to convert the object to.                                                     |
| ...    | An optional list of parameters to use in the as_*_format function specified by the format parameter. |

## Details

See the [MicrobiomeR\\_Formats](#) documentation.

## Value

A Taxmap object in the specified format.

## See Also

[which\\_format](#), [as\\_raw\\_format](#), [as\\_basic\\_format](#), [as\\_analyzed\\_format](#), [as\\_phyloseq\\_format](#), [create\\_taxmap](#), [order\\_metacoder\\_data](#)

Other Formatting: [MicrobiomeR\\_Formats](#), [as\\_analyzed\\_format\(\)](#), [as\\_basic\\_format\(\)](#), [as\\_custom\\_format\(\)](#), [as\\_phyloseq\\_format\(\)](#), [as\\_raw\\_format\(\)](#), [melt\\_taxmap\(\)](#), [treatment\\_matrix\(\)](#)

---

as\_phyloseq\_format *As Phyloseq MicrobiomeR Format*


---

## Description

Converts the Taxmap object to the phyloseq\_format.

**Usage**

```
as_phyloseq_format(
  obj,
  otu_table = "otu_abundance",
  tax_data = "otu_annotations",
  sample_data = "sample_data",
  phy_tree = "phy_tree"
)
```

**Arguments**

|             |                                                                                   |
|-------------|-----------------------------------------------------------------------------------|
| obj         | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> . |
| otu_table   | The name of the observation table with OTU data. Default: NULL                    |
| tax_data    | The name of the observation table with taxonomic annotations. Default: NULL       |
| sample_data | The name of the observation table with metadata. Default: NULL                    |
| phy_tree    | The name of the observation data with the phylogenetic tree. Default: NULL        |

**Details**

See the [MicrobiomeR\\_Formats](#) documentation.

**Value**

A Taxmap object in the phyloseq\_format.

**See Also**

[create\\_taxmap](#), [order\\_metacoder\\_data](#)

Other Formatting: [MicrobiomeR\\_Formats](#), [as\\_MicrobiomeR\\_format\(\)](#), [as\\_analyzed\\_format\(\)](#), [as\\_basic\\_format\(\)](#), [as\\_custom\\_format\(\)](#), [as\\_raw\\_format\(\)](#), [melt\\_taxmap\(\)](#), [treatment\\_matrix\(\)](#)

---

as\_raw\_format

*As Raw MicrobiomeR Format*


---

**Description**

Converts a Taxmap object to the raw\_format.

**Usage**

```
as_raw_format(obj)
```

**Arguments**

|     |                                                                                   |
|-----|-----------------------------------------------------------------------------------|
| obj | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> . |
|-----|-----------------------------------------------------------------------------------|

**Details**

See the [MicrobiomeR\\_Formats](#) documentation.

**Value**

A Taxmap object in the "raw\_format".

**See Also**

`is_phyloseq_format`, `is_raw_format`, `order_metacoder_data` @importFrom crayon silver red green

Other Formatting: `MicrobiomeR_Formats`, `as_MicrobiomeR_format()`, `as_analyzed_format()`, `as_basic_format()`, `as_custom_format()`, `as_phyloseq_format()`, `melt_taxmap()`, `treatment_matrix()`

---

|                     |                               |
|---------------------|-------------------------------|
| combination_palette | <i>Combine Color Palettes</i> |
|---------------------|-------------------------------|

---

**Description**

This function uses dynamic arguments (...) in order to combine multiple color palettes together.

**Usage**

```
combination_palette(...)
```

**Arguments**

... You can use any name for your arguments, but the values must be a named list. The list can only have 4 named members:

**palette** This is a palette function that returns a vector of colors.

**args** This is another named list used for the palette function parameters.

**range** This is a range (*1:10*) used to subset the color palette vector.

**rev** This is a logical (*TRUE/FALSE*) used to reverse the color palette.

You can add as many parameters you want in order to combine as many color palettes as you want.

**Details**

This function allows you to combine a varying number of color palettes and gives you the ability to subset and reverse the palettes that are supplied.

**Value**

The output of this function is another function (`grDevices::colorRampPalette`), which takes a number to generate an interpolated color palette as a character vector.

**See Also**[colorRamp](#)Other Color Palettes: [get\\_color\\_palette\(\)](#), [scico\\_palette\(\)](#), [viridis\\_magma\\_palette\(\)](#), [viridis\\_palette\(\)](#)**Examples**

```
## Not run:
if(interactive()){
# Below is the code for the viridis_magma_palette function.
# It's a good example of how to use the combination_palette function.
viridis_magma_palette <- function(viridis_number = 800,
                                viridis_range = 300:viridis_number,
                                viridis_rev = TRUE,
                                magma_number = 500,
                                magma_range = 0:magma_number,
                                magma_rev = FALSE,
                                ...) {
  if (!missing(...)){
    v_args = list(n=viridis_number, ...)
    m_args = list(n=magma_number, ...)
  } else {
    v_args = list(n=viridis_number)
    m_args = list(n=magma_number)
  }
  crp <- combination_palette(viridis =
    list(palette = viridis::viridis,
        args = v_args,
        range = viridis_range,
        rev = viridis_rev),
    magma =
    list(palette = viridis::magma,
        args = m_args,
        range = magma_range,
        rev = magma_rev)
  )
  return(crp)
}
}

## End(Not run)
```

---

convert\_proportions      *Convert Proportions*


---

**Description**

Convert Proportions

**Usage**

```
convert_proportions(melted_df, tax_level)
```

**Arguments**

`melted_df` A "melted" dataframe from the metacoder object's data.  
`tax_level` The taxonomic level, Default: 'Phylum'

**See Also**

Other Data Manipulators: [metacoder\\_comp\\_func\\_1\(\)](#), [transformer\(\)](#), [transposer\(\)](#), [vlookup\(\)](#)

---

|                  |                                  |
|------------------|----------------------------------|
| correlation_data | <i>Get Correlation Plot Data</i> |
|------------------|----------------------------------|

---

**Description**

Get the correlation plot data comparing all of the treatment groups.

**Usage**

```
correlation_data(obj, primary_rank, secondary_rank = TRUE, wp_value = 0.05)
```

**Arguments**

`obj` An object to be converted to a Taxmap object with [create\\_taxmap](#).  
`primary_rank` A primary rank used to label the points.  
`secondary_rank` The secondary rank used to color the points, Default: TRUE  
`wp_value` The wilcoxon p-value cutoff/threshold, Default: 0.05

**Details**

DETAILS

**Value**

Data for the correlation plots.

**See Also**

[agglomerate\\_taxmap](#), [vlookup](#)

[data mask programming](#), [mutate](#), [filter](#), [arrange](#)

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  #EXAMPLE1
}

## End(Not run)
```

---

|                  |                         |
|------------------|-------------------------|
| correlation_plot | <i>Correlation Plot</i> |
|------------------|-------------------------|

---

**Description**

Create a correlation plot from a metacoder/taxmap object.

**Usage**

```
correlation_plot(
  obj,
  primary_rank,
  secondary_rank = TRUE,
  wp_value = 0.05,
  pal_func = NULL,
  trans = "logit"
)
```

**Arguments**

|                |                                                                                                                                                                                                                                                                                                   |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obj            | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> .                                                                                                                                                                                                                 |
| primary_rank   | The primary rank used to label the points.                                                                                                                                                                                                                                                        |
| secondary_rank | The secondary rank used to color the points. Can be an integer specifying the number of supertaxon ranks above the primary rank or the name of a supertaxon rank. Default: TRUE                                                                                                                   |
| wp_value       | The Wilcoxon P-Value used to represent significant points. Default: 0.05                                                                                                                                                                                                                          |
| pal_func       | A palette function that returns <code>grDevices::colorRampPalette</code> .                                                                                                                                                                                                                        |
| trans          | Either the name of a transformation object, or the object itself given to <a href="#">scale_continuous</a> . Built-in transformations include "asn", "atanh", "boxcox", "exp", "identity", "log", "log10", "log1p", "log2", "logit", "probability", "probit", "reciprocal", "reverse" and "sqrt". |

**Details**

Correlation plots help to better explain the heat tree findings.

**Value**

A 1:1 correlation plot built with `ggplot2`.

**See Also**

[create\\_taxmap](#), [validate\\_MicrobiomeR\\_format](#), [correlation\\_data](#), [plot\\_limits](#), [get\\_color\\_palette](#)  
[ggplot](#), [aes](#), [geom\\_polygon](#), [geom\\_point](#), [labs](#), [scale\\_continuous](#), [scale\\_manual](#), [guide\\_legend](#),  
[geom\\_abline](#)  
[geom\\_label\\_repel](#)  
[fct\\_reorder](#)

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plots\(\)](#),  
[heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#),  
[save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#),  
[stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  # This example uses data that are no longer available in the MicrobiomeR package,
  # however, they can be easily generated with \code{\link{MicrobiomeR}{as_analyzed_format}}.
  library(MicrobiomeR)
  analyzed_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "analyzed_format")
  correlation_plot(analyzed_silva, primary_rank = "Class", secondary_rank = "Phylum")
}

## End(Not run)
```

---

|                   |                                       |
|-------------------|---------------------------------------|
| correlation_plots | <i>Get Multiple Correlation Plots</i> |
|-------------------|---------------------------------------|

---

**Description**

This function allows the user to create a list of multiple correlation plots.

**Usage**

```
correlation_plots(obj, primary_ranks, secondary_ranks = TRUE, ...)
```

**Arguments**

|                              |                                                                                                                                                                                 |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>obj</code>             | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> .                                                                                               |
| <code>primary_ranks</code>   | A vector of primary ranks used to label the points.                                                                                                                             |
| <code>secondary_ranks</code> | The secondary rank used to color the points. Can be an integer specifying the number of supertaxon ranks above the primary rank or the name of a supertaxon rank. Default: TRUE |
| <code>...</code>             | An optional list of parameters to use in the <code>correlation_plot</code> function.                                                                                            |

Details

This function makes it easier to generate multiple correlation plots at once.

Value

A list object containing correlation plots. A pairwise comparison returns a nested list.

See Also

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

Examples

```
## Not run:
if(interactive()){
  # This example uses data that are no longer available in the MicrobiomeR package,
  # however, they can be easily generated with \code{\link{MicrobiomeR}{as_analyzed_format}}.
  library(MicrobiomeR)
  analyzed_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "analyzed_format")
  corr_plots <- correlation_plots(analyzed_silva, primary_ranks = c("Phylum", "Class", "Order"),
                                secondary_ranks = c("Phylum", "Class", "Order", "Family", "Genus"))

  # Show a plot
  corr_plots$Class$Phylum
}

## End(Not run)
```

---

|            |                                        |
|------------|----------------------------------------|
| cov_filter | <i>Coefficient of Variation Filter</i> |
|------------|----------------------------------------|

---

Description

This function filters OTUs that have a variance higher than the specified CoV.

Usage

```
cov_filter(obj, coefficient_of_variation, validated = FALSE)
```

Arguments

- obj                    A Taxmap object.
- coefficient\_of\_variation            The maximum CoV that an OTU can have.
- validated            This parameter provides a way to override validation steps. Use carefully. Default: FALSE

## Details

This function helps remove OTUs that have an unusually high variance using the coefficient of variation.

## Value

Returns a taxmap object that contains otu\_ids that have passed the above filter.

## See Also

[validate\\_MicrobiomeR\\_format](#), [otu\\_id\\_filter](#)

[summarise\\_if](#)

Other Advanced Metacoder Filters: [agglomerate\\_taxmap\(\)](#), [otu\\_prevalence\\_filter\(\)](#), [otu\\_proportion\\_filter\(\)](#), [taxa\\_prevalence\\_filter\(\)](#)

## Examples

```
## Not run:
if(interactive()){
# Use the cov_filter towards the end of your analysis
library(MicrobiomeR)
library(metacoder)
library(taxa)

# Convert Phyloseq object to taxmap object
metacoder_obj <- as_MicrobiomeR_format(obj = phyloseq_obj, format = "raw_format")

# Remove Archaea from the taxmap object
metacoder_obj <- filter_taxa(
  obj = metacoder_obj,
  taxon_names == "Archaea",
  subtaxa = TRUE,
  invert = TRUE)

# Ambiguous Annotation Filter - Remove taxonomies with ambiguous names
metacoder_obj <- filter_ambiguous_taxa(metacoder_obj, subtaxa = TRUE)

# Low Sample Filter - Remove the low samples
metacoder_obj <- sample_id_filter(obj = metacoder_obj,
  .f_filter = ~sum(.),
  .f_condition = ~.>= 20, validated = TRUE)

# Master Threshold Filter - Add the otu_proportions table and then filter OTUs based on min %
metacoder_obj <- otu_proportion_filter(
  obj = metacoder_obj,
  otu_percentage = 0.00001
)

# Taxa Prevalence Filter
# The default minimum abundance is 5 and the sample percentage is 0.5 (5%).
# Phylum
metacoder_obj <- taxa_prevalence_filter(
```

```

      obj = metacoder_obj,
      rank = "Phylum"
    )
  # Class
  metacoder_obj <- taxa_prevalence_filter(
    obj = metacoder_obj,
    rank = "Class",
    validated = TRUE
  )

  # OTU prevalence filter
  metacoder_obj <- otu_prevalence_filter(obj = metacoder_obj, validated = TRUE)

  # Coefficient of Variation Filter - Filter OTUs based on the coefficient of variation
  metacoder_obj <- cov_filter(obj = metacoder_obj,
                              coefficient_of_variation = 3,
                              validated = TRUE)

}

## End(Not run)

```

---

create\_phyloseq

*Get Phyloseq Object*


---

## Description

Create a phyloseq object using a biom file, phylogenetic tree file, and a metadata file. Alternatively, an .Rdata file can be used. For now this function takes output from Qiime. The data provided by this package was processed by the NIH's Nephele pipeline.

## Usage

```

create_phyloseq(
  biom_file = NULL,
  tree_file = NULL,
  metadata_file = NULL,
  treatment_group = NULL,
  parse_func = NULL,
  rdata_file = NULL,
  save_rooted_tree = FALSE,
  recursive_save = FALSE
)

```

## Arguments

|           |                                                                                                                                        |
|-----------|----------------------------------------------------------------------------------------------------------------------------------------|
| biom_file | A file in the <b>BIOM format</b> . Default: NULL                                                                                       |
| tree_file | A phylogenetic tree file. This function will generate a rooted tree file if your phylogenetic tree isn't already rooted. Default: NULL |

|                  |                                                                                                                                                                                                                               |
|------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| metadata_file    | Sample metadata in tab delimited format. It's very important that you make sure that you have a <b>sample_id</b> and <b>TreatmentGroup</b> column in your metadata file. You might otherwise run into problems. Default: NULL |
| treatment_group  | The column number or name in the metadata file that contains the treatment group names. Default: NULL                                                                                                                         |
| parse_func       | The parse function used to parse taxonomy strings from Greengenes or SILVA database. Default: NULL                                                                                                                            |
| rdata_file       | A .Rdata file. Default: NULL                                                                                                                                                                                                  |
| save_rooted_tree | A logical that determines whether or not the rooted tree is saved. Default: FALSE                                                                                                                                             |
| recursive_save   | If the directory doesn't exist create the parent directories that don't exist as well. Default: FALSE                                                                                                                         |

## Details

This function heavily relies on the phyloseq package to import data into R. It also requires you to use an absolute or relative path to your data files or for your data files to be in the working directory.

It's very important that you make sure that you have a **sample\_id** and **TreatmentGroup** column in your metadata file.

## Value

A phyloseq object, and a phylogenetic tree file if one does not already exist.

## See Also

[import\\_biom](#), [phyloseq::sample\\_data\(\)](#), [phyloseq::phy\\_tree\(\)](#), [import\\_qiime\\_sample\\_data](#), [merge\\_phyloseq](#)

[root](#), [is.rooted](#), [read.tree](#), [write.tree](#)

[root\\_by\\_longest\\_edge](#)

Other Data Importers: [create\\_taxmap\(\)](#)

## Examples

```
## Not run:
if(interactive()){
  library(MicrobiomeR)
  # Get the datafiles from the package
  biom_file <- system.file("extdata", "silva_OTU.biom", package = "MicrobiomeR")
  tree_file <- system.file("extdata", "silva.tre", package = "MicrobiomeR")
  md_file <- system.file("extdata", "nephele_metadata2.txt", package = "MicrobiomeR")
  parse_func <- parse_parse_taxonomy_silva_128
}
# Create a phyloseq object from the data files.
phy_obj <- create_phyloseq(biom_file = biom_file,
                          tree_file = tree_file,
```

```

                                metadata_file = md_file,
                                parse_func = parse_func)

## End(Not run)

```

---

|                  |                                 |
|------------------|---------------------------------|
| create_pub_table | <i>Create Publication Table</i> |
|------------------|---------------------------------|

---

### Description

Creates an image of a table from a dataframe.

### Usage

```
create_pub_table(dataframe)
```

### Arguments

dataframe      A dataframe object.

### Value

The text table as an object that can be saved to png/tiff/jpg/pdf.

### Examples

```

## Not run:
if(interactive()){
# This example uses data that are no longer available in the MicrobiomeR package,
# however, they can be easily generated with \code{\link{MicrobiomeR}{as_basic_format}}.
library(MicrobiomeR)
data <- MicrobiomeR::basic_silva$data$taxa_abundance
create_pub_table(data)
}

## End(Not run)

```

---

|               |                          |
|---------------|--------------------------|
| create_taxmap | <i>Get Taxmap Object</i> |
|---------------|--------------------------|

---

### Description

A function that handles the conversion of objects to Taxmap (**taxa::taxmap**) objects.

### Usage

```
create_taxmap(obj)
```

**Arguments**

**obj** An object that contains the data being analyzed. Can be one of the following:

- Phyloseq Object** An object generated from the phyloseq package.
- Taxmap Object** An object generated from the metacoder or taxa package.
- RData file** An RData file generated from the base::save function. Can have an extension of .RData or .rda.

**Details**

This function is used to convert data to metacoder/taxmap objects for *microbiome* analysis. This function is used at the beginning of every other function to support multiple types of input for the obj parameter in those functions. For phyloseq inputs, this function uses [parse\\_phyloseq](#) when possible and falls back to an internal parser for known upstream metacoder namespace breakage.

**Value**

The output generated is a **taxa::taxmap** object.

**See Also**

[parse\\_phyloseq](#)

[fileutils](#)

Other Data Importers: [create\\_phyloseq\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  library(MicrobiomeR)
  phy_obj <- MicrobiomeR::phyloseq_silva_2
  mc_obj <- create_taxmap(phy_obj)
}

## End(Not run)
```

---

get\_color\_palette

*Get a Color Palette*


---

**Description**

Get a color palette with a specific number of colors.

**Usage**

```
get_color_palette(
  pal_func = viridis_magma_palette,
  color_no = 20,
  display = TRUE,
  rev = FALSE,
  ...
)
```

**Arguments**

|          |                                                                                                                                                                                                 |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| pal_func | A function that <i>returns the output function</i> from <a href="#">colorRamp</a> or a <i>the output of the output function</i> from <a href="#">colorRamp</a> . Default: viridis_magma_palette |
| color_no | The number of colors in the palette. Default: 20                                                                                                                                                |
| display  | A logical for displaying a pie chart of the palette. Default: TRUE                                                                                                                              |
| rev      | A logical that will reverse the order of the color palette.                                                                                                                                     |
| ...      | Parameters for the <i>pal_func</i> .                                                                                                                                                            |

**Details**

This function is meant to be a plugin style function for users to created palettes.

**Value**

Returns a color palette in the form of a character vector.

**See Also**

Other Color Palettes: [combination\\_palette\(\)](#), [scico\\_palette\(\)](#), [viridis\\_magma\\_palette\(\)](#), [viridis\\_palette\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  # This example uses data that are no longer available in the MicrobiomeR package,
  # however, they can be easily generated with \code{\link{MicrobiomeR}{as_analyzed_format}}.
  library(MicrobiomeR)
  analyzed_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "analyzed_format")
  data <- analyzed_silva$data$stats_tax_data$Phylum
  data_len <- length(unique(data))
  custom_pal <- get_color_palette(color_no=data_len)
  print(custom_pal)
}

## End(Not run)
```

---

heat\_tree\_parameters    *Get Heat Tree Parameters*

---

## Description

This function gets the parameters used for the heat\_tree\_plots function.

## Usage

```
heat_tree_parameters(obj, title, treatment_no, ...)
```

## Arguments

|              |                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obj          | A Taxmap object.                                                                                                                                                                                                                                                                                                                                                                                                                    |
| title        | The title used in the heat_tree plot.                                                                                                                                                                                                                                                                                                                                                                                               |
| treatment_no | The number of treatment groups in the data.                                                                                                                                                                                                                                                                                                                                                                                         |
| ...          | Any of the heat tree parameters list below can be used to change the way the heat_tree output is displayed. However, this function acts as a default list of parameters. The members of the default list will be overridden by the dot parameters. Any variable in obj\$data\$stats_tax_data can be used to manipulate the heat tree parameters. Function calls from the taxa package must be done explicitly on the Taxmap object. |

## Value

A list used with do.call and the metacoder::heat\_tree function.

## See Also

[heat\\_tree](#)

[n\\_obs](#), [taxon\\_names](#)

[list\\_modify](#)

[enquos](#), [is\\_quosure](#), [eval\\_tidy](#)

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

---

|                 |                            |
|-----------------|----------------------------|
| heat_tree_plots | <i>Get Heat Tree Plots</i> |
|-----------------|----------------------------|

---

## Description

A function for getting multiple heat\_tree plots per rank.

## Usage

```
heat_tree_plots(obj, rank_list = NULL, title = TRUE, seed = 1, ...)
```

## Arguments

|           |                                                                                                                                                                                                             |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obj       | An object to be converted to a Taxmap object with <a href="#">create_taxmap</a> .                                                                                                                           |
| rank_list | A vector of ranks used to generate heat_trees. Default: NULL                                                                                                                                                |
| title     | Can be a logical, NULL, or a string. The string can utilize {rank} to dynamically display the rank in the title via <a href="#">glue</a> .                                                                  |
| seed      | A single integer seed used to make heat-tree layouts reproducible without modifying the caller's RNG state. Use NULL to defer to the current RNG stream.                                                    |
| ...       | Any of the <a href="#">heat_tree</a> parameters can be used to change the way the heat_tree output is displayed. Please see the <a href="#">heat_tree_parameters</a> documentation for further explanation. |

## Details

By default, node size represents the number of retained OTUs assigned to each taxon or any of its descendants in the source otu\_abundance table. Counts are pooled across samples after upstream filtering and do not represent read abundance.

## Value

A list of heat\_tree plots.

## See Also

[heat\\_tree](#)

[create\\_taxmap](#), [validate\\_MicrobiomeR\\_format](#), [heat\\_tree\\_parameters](#)

[filter\\_obs](#)

[crayon](#)

[theme](#), [margin](#), [labs](#)

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plot\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

## Examples

```
## Not run:
if(interactive()){
  # This example uses data that are no longer available in the MicrobiomeR package,
  # however, they can be easily generated with \link{MicrobiomeR}{as_analyzed_format}.
  library(MicrobiomeR)
  analyzed_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "analyzed_format")
  h_trees <- heat_tree_plots(analyzed_silva, rank_list = c("Phylum", "Class"))
  h_trees$Class
}

## End(Not run)
```

---

|                    |                                       |
|--------------------|---------------------------------------|
| is_analyzed_format | <i>Is Analyzed MicrobiomeR Format</i> |
|--------------------|---------------------------------------|

---

## Description

This function returns a logical based on whether or not the object is in the analyzed\_format.

## Usage

```
is_analyzed_format(obj)
```

## Arguments

obj                      A Taxmap object.

## Details

The "analyzed\_format" is Level 3. in the [MicrobiomeR\\_Formats](#) hierarchy.

## Value

A logical (TRUE/FALSE).

## See Also

Other Validation: [is\\_basic\\_format\(\)](#), [is\\_phyloseq\\_format\(\)](#), [is\\_raw\\_format\(\)](#), [order\\_metacoder\\_data\(\)](#), [validate\\_MicrobiomeR\\_format\(\)](#), [which\\_format\(\)](#)

## Examples

```
## Not run:
if(interactive()){
  library(MicrobiomeR)
  if(is_analyzed_format(MicrobiomeR::raw_silva_2)){
    print("It's analyzed!")
  } else {
```

```

    fmt <- which_format(MicrobiomeR::raw_silva_2)
    print("It's not analyzed!")
    print(sprintf("It's %s!", fmt))
  }
}

## End(Not run)

```

---

is\_basic\_format

*Is Basic MicrobiomeR Format*


---

## Description

This function returns a logical based on whether or not the object is in the basic\_format.

## Usage

```
is_basic_format(obj)
```

## Arguments

obj                    A Taxmap object.

## Details

The "basic\_format" is Level 2. in the [MicrobiomeR\\_Formats](#) hierarchy.

## Value

A logical (TRUE/FALSE).

## See Also

Other Validation: [is\\_analyzed\\_format\(\)](#), [is\\_phyloseq\\_format\(\)](#), [is\\_raw\\_format\(\)](#), [order\\_metacoder\\_data\(\)](#), [validate\\_MicrobiomeR\\_format\(\)](#), [which\\_format\(\)](#)

## Examples

```

## Not run:
if(interactive()){
  library(MicrobiomeR)
  if(is_basic_format(MicrobiomeR::raw_silva_2)){
    print("It's basic!")
  } else {
    fmt <- which_format(MicrobiomeR::raw_silva_2)
    print("It's not analyzed!")
    print(sprintf("It's %s!", fmt))
  }
}

## End(Not run)

```

---

|                    |                                       |
|--------------------|---------------------------------------|
| is_phyloseq_format | <i>Is Phyloseq MicrobiomeR Format</i> |
|--------------------|---------------------------------------|

---

## Description

This function returns a logical based on whether or not the object is in the phyloseq\_format.

## Usage

```
is_phyloseq_format(obj)
```

## Arguments

obj                      A Taxmap object.

## Details

The "phyloseq\_format" is Level 0. in the [MicrobiomeR\\_Formats](#) hierarchy.

## Value

A logical (TRUE/FALSE).

## See Also

Other Validation: [is\\_analyzed\\_format\(\)](#), [is\\_basic\\_format\(\)](#), [is\\_raw\\_format\(\)](#), [order\\_metacoder\\_data\(\)](#), [validate\\_MicrobiomeR\\_format\(\)](#), [which\\_format\(\)](#)

## Examples

```
## Not run:
if(interactive()){
  library(MicrobiomeR)
  if(is_analyzed_format(MicrobiomeR::raw_silva_2)){
    print("It's phyloseq format!")
  } else {
    fmt <- which_format(MicrobiomeR::raw_silva_2)
    print("It's not phyloseq fomrat!")
    print(sprintf("It's %s!", fmt))
  }
}

## End(Not run)
```

---

is\_raw\_format*Is Raw MicrobiomeR Format*

---

## Description

This function returns a logical based on whether or not the object is in the raw\_format.

## Usage

```
is_raw_format(obj)
```

## Arguments

obj                    A Taxmap object.

## Details

The "raw\_format" is Level 1. in the [MicrobiomeR\\_Formats](#) hierarchy.

## Value

A logical (TRUE/FALSE).

## See Also

Other Validation: [is\\_analyzed\\_format\(\)](#), [is\\_basic\\_format\(\)](#), [is\\_phyloseq\\_format\(\)](#), [order\\_metacoder\\_data\(\)](#), [validate\\_MicrobiomeR\\_format\(\)](#), [which\\_format\(\)](#)

## Examples

```
## Not run:
if(interactive()){
  library(MicrobiomeR)
  if(is_raw_format(MicrobiomeR::raw_silva_2)){
    print("It's raw!")
  } else {
    fmt <- which_format(MicrobiomeR::raw_silva_2)
    print("It's not analyzed!")
    print(sprintf("It's %s!", fmt))
  }
}

## End(Not run)
```

---

|             |                    |
|-------------|--------------------|
| melt_taxmap | <i>Melt Taxmap</i> |
|-------------|--------------------|

---

**Description**

Melt Taxmap

**Usage**

```
melt_taxmap(obj)
```

**Arguments**

obj                    An object to be converted to a taxmap object with [create\\_taxmap](#).

**See Also**

Other Formatting: [MicrobiomeR\\_Formats](#), [as\\_MicrobiomeR\\_format\(\)](#), [as\\_analyzed\\_format\(\)](#), [as\\_basic\\_format\(\)](#), [as\\_custom\\_format\(\)](#), [as\\_phyloseq\\_format\(\)](#), [as\\_raw\\_format\(\)](#), [treatment\\_matrix\(\)](#)

---

|                       |                                         |
|-----------------------|-----------------------------------------|
| metacoder_comp_func_1 | <i>Metacoder Comparison Function #1</i> |
|-----------------------|-----------------------------------------|

---

**Description**

A comparison function for metacoder::compare\_groups "func" parameter.

**Usage**

```
metacoder_comp_func_1(abund_1, abund_2)
```

**Arguments**

abund\_1                A character vector of abundances.  
 abund\_2                A character vector of abundances.

**Details**

This function is used by metacoder::compare\_groups in order to compare every combination of treatment groups.

**Value**

A list of statistical results used to compare groups.

**See Also**[dip.test](#)Other Data Manipulators: [convert\\_proportions\(\)](#), [transformer\(\)](#), [transposer\(\)](#), [vlookup\(\)](#)


---

MicrobiomeR\_Formats      *The MicrobiomeR Formats*


---

**Description**

The formats are listed in order from lowest to highest level. To get a *taxmap* object into the higher formats they must be in one of the lower formats. It is also inherently important to understand the [MicrobiomeR\\_Workflow](#), which describes some of the important dependencies used to create the MicrobiomeR formats listed below:

**phyloseq\_format** Level 0. The "phyloseq\_format" is the format given to a *taxmap* object that has just been converted from a *phyloseq* object using *metacoder* via [create\\_phyloseq](#). A *taxmap* object identified by MicrobiomeR to be in this format will contain "otu\_table", "tax\_data", "sample\_data" and "phy\_tree" data in the observation tables (**metacoder\_obj\$data**), which represents the matching *phyloseq::otu\_table*, *phyloseq::tax\_table*, *phyloseq::sample\_data*, and *phyloseq::phy\_tree* data in the original *phyloseq* object.

**raw\_format** Level 1. The "raw\_format" is the format given to a *taxmap* object that has been processed with [as\\_raw\\_format](#). It contains the "otu\_abundance" and "otu\_annotations" data in the observation tables, which are just name conversions of the *otu\_table* and *tax\_data* table from the "phyloseq\_format".

**basic\_format** Level 2. The "basic\_format" is the format given to a *taxmap* object that has been processed with [as\\_basic\\_format](#). It contains the tables available in the "raw\_format" along with "taxa\_abundance", "otu\_proportions", and "taxa\_proportions" tables.

**analyzed\_format** Level 3. The "analyzed\_format" is the format given to a *taxmap* object that has been processed with [as\\_analyzed\\_format](#). It contains the tables available in the "basic\_format" along with "statistical\_data" and "stats\_tax\_data" tables.

**See Also**

Other Formatting: [as\\_MicrobiomeR\\_format\(\)](#), [as\\_analyzed\\_format\(\)](#), [as\\_basic\\_format\(\)](#), [as\\_custom\\_format\(\)](#), [as\\_phyloseq\\_format\(\)](#), [as\\_raw\\_format\(\)](#), [melt\\_taxmap\(\)](#), [treatment\\_matrix\(\)](#)

## Description

In order to understand the MicrobiomeR packages workflow it is necessary to understand a few other packages. Primarily **phyloseq**, **taxa**, and **metacoder**.

## Data Import

Data is imported with the `create_phyloseq` function or something proprietary that creates a `phyloseq` object with an `phyloseq::otu_table`, a `phyloseq::tax_table`, a `phyloseq::sample_data`, and a `phyloseq::phy_tree`. The data is preferably not filtered with phyloseq at this stage.

## Data Manipulation

Data is manipulated into a `metacoder::taxmap` object using the `metacoder::parse_phyloseq` function. Custom `MicrobiomeR_Formats` are used to validate the data for consistency. MicrobiomeR and metacoder filtering functions can be used to preprocess the data before visualization.

## Visualizations

MicrobiomeR contains useful functions for creating visualizations such as `heat_trees`, correlations plots, stacked bar charts, violin plots, and PCA plots. These functions are also coupled with save functions that organize saved output by experiment.

## Description

A function for changing the order of the observation data in a Taxmap object

## Usage

```
order_metacoder_data(obj)
```

## Arguments

`obj`                      An object to be converted to a Taxmap object with `create_taxmap`.

## Details

Changes the order of the observation tables in `metacoder_object$data` to `otu_abundance`, `otu_annotations`, `otu_proportions`, `sample_data`, `phy_tree`, `taxa_abundance`, `taxa_proportions`, `statistical_data`, and `stats_tax_data` respectively.

**Value**

A taxmap object with observation data in the proper order for downstream analysis.

**See Also**

Other Validation: [is\\_analyzed\\_format\(\)](#), [is\\_basic\\_format\(\)](#), [is\\_phyloseq\\_format\(\)](#), [is\\_raw\\_format\(\)](#), [validate\\_MicrobiomeR\\_format\(\)](#), [which\\_format\(\)](#)

---

|                 |                        |
|-----------------|------------------------|
| ordination_plot | <i>Ordination Plot</i> |
|-----------------|------------------------|

---

**Description**

This function allows for ordination (which helps us to distinguish beta diversity relationships) to be plotted as well as for the corresponding data to be returned.

Generate plots for a list of ordination methods and distances.

**Usage**

```
ordination_plot(
  obj,
  method = "PCoA",
  distance = "wunifrac",
  color = "TreatmentGroup",
  title = NULL,
  only_data = FALSE
)

ordination_plots(
  obj,
  methods = c("PCoA", "NMDS"),
  distances = c("wunifrac", "unifrac", "bray"),
  color = "TreatmentGroup",
  select_otu_table = "otu_proportions"
)
```

**Arguments**

|          |                                                                                                                        |
|----------|------------------------------------------------------------------------------------------------------------------------|
| obj      | An object to be converted to a metacoder object with <a href="#">create_taxmap</a> .                                   |
| method   | Choose an ordination method from 'PCoA', 'CCA', 'NMDS' or 'DPCoA', Default: 'PCoA'                                     |
| distance | Choose a <a href="#">distance</a> method or use a pre-computed <a href="#">dist</a> -class object, Default: 'wunifrac' |
| color    | Choose the group or factor of which colors will be mapped to, Default: 'TreatmentGroup'                                |

|                  |                                                                           |
|------------------|---------------------------------------------------------------------------|
| title            | The title of the plot, Default: NULL                                      |
| only_data        | Allows for only ordination data to be generated, Default: FALSE           |
| methods          | A list of ordination methods, Default: 'c("PCoA", "NMDS")'                |
| distances        | A list of distance methods, Default: 'c("wunifrac", "unifrac", "bray")'   |
| select_otu_table | The data table to use in the observation data. Default: "otu_proportions" |

## Value

By default, it returns an ordination plot.

Returns a melted dataframe.

## See Also

View [save\\_ordination\\_plots](#) to save your ordination plot or multiple ordination plots.

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plot\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plot\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

## Examples

```
## Not run:
if (interactive()) {
  # An example ordination plot
  library(MicrobiomeR)
  data <- analyzed_silva
  plot <- ordination_plot(obj = data)
  plot
}

## End(Not run)
```

---

otu\_id\_filter

*Filter OTU Ids from Taxmap Objects*

---

## Description

This function provides a flexible way to filter unwanted otu\_ids from the taxmap object and from the observations of a MicrobiomeR formatted object.

**Usage**

```
otu_id_filter(
  obj,
  .f_transform = NULL,
  .f_filter = NULL,
  .f_condition = NULL,
  validated = FALSE,
  ...
)
```

**Arguments**

|                           |                                                                                                                  |
|---------------------------|------------------------------------------------------------------------------------------------------------------|
| <code>obj</code>          | A Taxmap object.                                                                                                 |
| <code>.f_transform</code> | A function used for transforming the data. Default: NULL                                                         |
| <code>.f_filter</code>    | A function used for summarising the data like 'sum' or 'mean'. Default: NULL                                     |
| <code>.f_condition</code> | A function that takes the summarised data and applied a condition like <code>x &gt; 10000</code> . Default: NULL |
| <code>validated</code>    | This parameter provides a way to override validation steps. Use carefully. Default: FALSE                        |
| <code>...</code>          | An optional list of parameters to use in the <code>.f_filter</code> function specified                           |

**Details**

Get the `otu_ids` to keep by using `purrr` and the user supplied transform and filter + condition formulas. The `purrr` package allows the use of anonymous functions as described in the link below:

[https://jennybc.github.io/purrr-tutorial/ls03\\_map-function-syntax.html#anonymous\\_function,\\_formula](https://jennybc.github.io/purrr-tutorial/ls03_map-function-syntax.html#anonymous_function,_formula)

**Value**

Returns a taxmap object with `otu_ids` that pass the filters.

**See Also**

`validate_MicrobiomeR_format`, `transposer`, `transformer`  
`get_dataset`, `filter_taxa`  
`select_all`  
`map`, `keep`

Other Basic Metacoder Filters: `sample_id_filter()`, `taxon_id_filter()`

**Examples**

```
## Not run:
if(interactive()){
  # Below is the code for the otu_prevalence_filter function.
  # It's a good example of how to use the otu_id_filter function.
```

```

library(MicrobiomeR)
otu_proportion_filter <- function(obj, otu_percentage = 0.00005, validated = FALSE) {
  mo_clone <- obj$clone()
  mo_clone <- validate_MicrobiomeR_format(obj = mo_clone,
                                          valid_formats = c("raw_format", "basic_format"),
                                          force_format = TRUE,
                                          validated = validated,
                                          min_or_max = min)

  # Filter OTU ids
  mo_clone <- otu_id_filter(obj = mo_clone,
                           .f_transform = ~./sum(.),
                           .f_filter = ~mean(.),
                           .f_condition = ~.> otu_percentage)

  return(mo_clone)
}
}

## End(Not run)

```

---

otu\_prevalence\_filter *OTU Prevalence Filter*

---

## Description

This function filters observations by their prevalence across samples.

## Usage

```

otu_prevalence_filter(
  obj,
  minimum_abundance = 5,
  rel_sample_percentage = 0.5,
  validated = FALSE
)

```

## Arguments

|                       |                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------|
| obj                   | A Taxmap object.                                                                          |
| minimum_abundance     | The minimum abundance needed per observation per sample. Default: 5                       |
| rel_sample_percentage | The percentage of samples per observation that meet the minimum abundance. Default: 0.5   |
| validated             | This parameter provides a way to override validation steps. Use carefully. Default: FALSE |

## Details

The `otu_prevalence_filter` filters `taxon_ids` that do not appear more than a certain amount of times (minimum abundance) in a certain percentage of samples (`rel_sample_percentage`). The [phyloseq workflow](#) calls for a minimum abundance of 5 across %50 of the samples. This filtering method is considered unsupervised, because it solely relies on the data in this experiment (OTU ids).

## Value

Returns a taxmap object that contains `taxon_ids` that have passed the above filter.

## See Also

[validate\\_MicrobiomeR\\_format](#)

[calc\\_prop\\_samples](#)

[filter](#)

[filter\\_taxa](#)

Other Advanced Metacoder Filters: [agglomerate\\_taxmap\(\)](#), [cov\\_filter\(\)](#), [otu\\_proportion\\_filter\(\)](#), [taxa\\_prevalence\\_filter\(\)](#)

## Examples

```
## Not run:
if(interactive()){
  library(MicrobiomeR)
  library(metacoder)
  library(taxa)

  # Convert Phyloseq object to taxmap object
  metacoder_obj <- as_MicrobiomeR_format(obj = phyloseq_obj, format = "raw_format")

  # Remove Archaea from the taxmap object
  metacoder_obj <- filter_taxa(
    obj = metacoder_obj,
    taxon_names == "Archaea",
    subtaxa = TRUE,
    invert = TRUE)

  # Ambiguous Annotation Filter - Remove taxonomies with ambiguous names
  metacoder_obj <- filter_ambiguous_taxa(metacoder_obj, subtaxa = TRUE)

  # Low Sample Filter - Remove the low samples
  metacoder_obj <- sample_id_filter(obj = metacoder_obj,
    .f_filter = ~sum(.),
    .f_condition = ~.>= 20, validated = TRUE)

  # Master Threshold Filter - Add the otu_proportions table and then filter OTUs based on min %
  metacoder_obj <- otu_proportion_filter(
    obj = metacoder_obj,
    otu_percentage = 0.00001
  )
}
```

```
# OTU prevalence filter
metacoder_obj <- otu_prevalence_filter(obj = metacoder_obj, validated = TRUE)
}

## End(Not run)
```

---

otu\_proportion\_filter *OTU Proportion Filter*

---

## Description

This function filters OTU values from the observation data and the taxmap object based on a minimum proportional mean across samples per OTU.

## Usage

```
otu_proportion_filter(obj, otu_percentage = 5e-05, validated = FALSE)
```

## Arguments

|                |                                                                                           |
|----------------|-------------------------------------------------------------------------------------------|
| obj            | A Taxmap object.                                                                          |
| otu_percentage | The minimum percentage used to compare against the proportional OTU mean. Default: 5e-05  |
| validated      | This parameter provides a way to override validation steps. Use carefully. Default: FALSE |

## Details

This type of filtering is used to remove OTUs that do not have a specified mean proportion. This function must be used conservatively, hence why the default otu\_percentage is so low.

## Value

Returns a taxmap object that contains otu\_ids that have passed the above filter.

## See Also

[validate\\_MicrobiomeR\\_format](#), [otu\\_id\\_filter](#)

Other Advanced Metacoder Filters: [agglomerate\\_taxmap\(\)](#), [cov\\_filter\(\)](#), [otu\\_prevalence\\_filter\(\)](#), [taxa\\_prevalence\\_filter\(\)](#)

## Examples

```
## Not run:
if(interactive()){
# Use the otu_proportions filter early on in your analysis
library(MicrobiomeR)
library(metacoder)
library(taxa)

# Convert Phyloseq object to taxmap object
metacoder_obj <- as_MicrobiomeR_format(obj = phyloseq_obj, format = "raw_format")

# Remove Archaea from the Taxmap object
metacoder_obj <- filter_taxa(
  obj = metacoder_obj,
  taxon_names == "Archaea",
  subtaxa = TRUE,
  invert = TRUE)

# Ambiguous Annotation Filter - Remove taxonomies with ambiguous names
metacoder_obj <- filter_ambiguous_taxa(metacoder_obj, subtaxa = TRUE)

# Low Sample Filter - Remove the low samples
metacoder_obj <- sample_id_filter(obj = metacoder_obj,
  .f_filter = ~sum(.),
  .f_condition = ~.>= 20, validated = TRUE)

# Master Threshold Filter - Add the otu_proportions table and then filter OTUs based on min %
metacoder_obj <- otu_proportion_filter(
  obj = metacoder_obj,
  otu_percentage = 0.00001
)
}

## End(Not run)
```

---

output\_dir

---

*Create an Output Directory*


---

## Description

A function for generating a consistent file system for data files and visualizations. If none of the parameters are set the full path generated by this function will be *root\_path/output*

## Usage

```
output_dir(
  start_path = NULL,
  experiment = NULL,
  plot_type = NULL,
```

```

    end_path = NULL,
    root_path = NULL,
    custom_path = NULL,
    overwrite = FALSE,
    mkdir = TRUE
  )

```

## Arguments

|             |                                                                                                                                                                                                      |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| start_path  | With the start_path set, the full path generated by this function will be <i>root_path/start_path/?experiment?</i> . Default: NULL                                                                   |
| experiment  | (optional) This will add a second level to the start_path file system. Default: NULL                                                                                                                 |
| plot_type   | (optional) This will add a third level to the start_path file system. With <b>ONLY</b> the plot_type set the full path generated by this function will be <i>root_path/plot_type</i> . Default: NULL |
| end_path    | (optional) This will add the final directory to the start_path file system Default: A formatted date-time string.                                                                                    |
| root_path   | The root of the new file system (start_path or not). Default: The working directory.                                                                                                                 |
| custom_path | This is an absolute path that overrides everything else. Default: NULL                                                                                                                               |
| overwrite   | A logical denoting that overwriting is acceptable. Default: FALSE                                                                                                                                    |
| mkdir       | A logical denoting weather or not the directory should be created or not. Default: TRUE                                                                                                              |

## Details

This function is incredibly useful on it's own but also for various other plotting/saving functions within the package. It helps keep data organized using a standard workflow.

## Value

Creates a directory for output and returns the path as a string.

## See Also

[glue](#)  
[case](#)

## Examples

```

## Not run:
if(interactive()){
  # Get the path to an output directory without creating
  library(MicrobiomeR)
  output_dir <- output_dir(start_path="output", experiment="microbiome-proj", mkdir=FALSE)
  print(output_dir)
}

```

```
# Create a folder for your plot types
output_dir <- output_dir(plot_type="scatter-plots")
print(output_dir)

}

## End(Not run)
```

---

parse\_taxonomy\_silva\_128

*Parse elements of a taxonomy vector*

---

## Description

These are provided as both example and default functions for parsing a character vector of taxonomic rank information for a single taxa. As default functions, these are intended for cases where the data adheres to the naming convention used by greengenes the naming convention used by greengenes and silva. (<http://greengenes.lbl.gov/cgi-bin/nph-index.cgi>) or where the convention is unknown, respectively. To work, these functions – and any similar custom function you may want to create and use – must take as input a single character vector of taxonomic ranks for a single OTU, and return a **named** character vector that has been modified appropriately (according to known naming conventions, desired length limits, etc. The length (number of elements) of the output named vector does **not** need to be equal to the input, which is useful for the cases where the source data files have extra meaningless elements that should probably be removed, like the ubiquitous “Root” element often found in greengenes/QIIME taxonomy labels. In the case of `parse_taxonomy_default`, no naming convention is assumed and so dummy rank names are added to the vector. More usefully if your taxonomy data is based on greengenes, the `parse_taxonomy_greengenes` function clips the first 3 characters that identify the rank, and uses these to name the corresponding element according to the appropriate taxonomic rank name used by greengenes (e.g. “p\_\_” at the beginning of an element means that element is the name of the phylum to which this OTU belongs). If you taxonomy data is based on SILVA, the `parse_taxonomy_silva_128` function clips the first 5 characters that identify rank, and uses these to name the corresponding element according to the appropriate taxonomic rank name used by SILVA (e.g. “D\_1\_\_” at the beginning of an element means that element is the name of the phylum to which this OTU belongs. Alternatively you can create your own function to parse this data. Most importantly, the expectations for these functions described above make them compatible to use during data import, specifically the `import_biom` function, but it is a flexible structure that will be implemented soon for all phyloseq import functions that deal with taxonomy (e.g. `import_qiime`).

## Usage

```
parse_taxonomy_silva_128(char.vec)
```

## Arguments

|                       |                                                                                                |
|-----------------------|------------------------------------------------------------------------------------------------|
| <code>char.vec</code> | (Required). A single character vector of taxonomic ranks for a single OTU, unprocessed (ugly). |
|-----------------------|------------------------------------------------------------------------------------------------|

## Details

This function is currently under PR review by phyloseq in a well supported pull request: <https://github.com/joey711/phyloseq/pull/854>. If you use this function, then please comment on the GitHub PR to encourage merging this feature.

## Value

A character vector in which each element is a different taxonomic rank of the same OTU, and each element name is the name of the rank level. For example, an element might be "Firmicutes" and named "phylum". These parsed, named versions of the taxonomic vector should reflect embedded information, naming conventions, desired length limits, etc; or in the case of `parse_taxonomy_default`, not modified at all and given dummy rank names to each element.

## See Also

`parse_taxonomy_default`  
`parse_taxonomy_greengenes`  
`parse_taxonomy_qiime`  
`import_biom`  
`import_qiime`

## Examples

```
## Not run:

> taxvec1 = c("Root", "k__Bacteria", "p__Firmicutes", "c__Bacilli", "o__Bacillales",
  "f__Staphylococcaceae")
> parse_taxonomy_default(taxvec1)
> parse_taxonomy_greengenes(taxvec1)
> taxvec2 = c("Root;k__Bacteria;p__Firmicutes;c__Bacilli;o__Bacillales;f__Staphylococcaceae")
> parse_taxonomy_qiime(taxvec2)
> taxvec3 = c("D_0__Bacteria", "D_1__Firmicutes", "D_2__Bacilli", "D_3__Staphylococcaceae")
> parse_taxonomy_silva_128(taxvec3)

## End(Not run)
```

---

permanova

*PERMANOVA*

---

## Description

Evaluate whether the group(s) has a significant effect on overall gut microbiota composition.

## Usage

```
permanova(obj, distance_method = "bray", group = "TreatmentGroup", seed = 1)
```

**Arguments**

|                 |                                                                                                                                                               |
|-----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------|
| obj             | An object to be converted to a taxmap object with <a href="#">create_taxmap</a> .                                                                             |
| distance_method | Use a desired distance method, Default: 'bray'                                                                                                                |
| group           | The group or column in the metadata to test upon, Default: 'TreatmentGroup'                                                                                   |
| seed            | A single integer seed used to make PERMANOVA permutations reproducible without modifying the caller's RNG state. Use NULL to defer to the current RNG stream. |

**Value**

Returns a list which includes permanova, anova, coefficients, and top coefficients.

**See Also**

View [top\\_coefficients\\_barplot](#) to plot the top\_coefficients returned from this function. See [adonis2](#) and [betadisper](#) to understand more about how the permanova data was generated

**Examples**

```
## Not run:
if (interactive()) {
  library(MicrobiomeR)
  data <- analyzed_silva
  p <- permanova(obj = data, group = "TreatmentGroup")
  p$permanova
}

## End(Not run)
```

---

phyloseq\_silva\_2

*Phyloseq Data with 2 Treatment Groups*


---

**Description**

Data from the Nephele-16S-Qiime-Silva microbiome pipeline in the form of a phyloseq object.

**Usage**

```
data(phyloseq_silva_2)
```

**Format**

An object of class "phyloseq"

**Source**

[Nephele-16S Qiime Pipeline](#)

**References**

Zhang et al. (2019) ([PubMed](#))

---

|                  |                                              |
|------------------|----------------------------------------------|
| phyloseq_silva_3 | <i>Phyloseq Data with 3 Treatment Groups</i> |
|------------------|----------------------------------------------|

---

**Description**

Data from the Nephele-16S-Qiime-Silva microbiome pipeline in the form of a phyloseq object.

**Usage**

```
data(phyloseq_silva_3)
```

**Format**

An object of class "phyloseq"

**Source**

[Nephele-16S Qiime Pipeline](#)

**References**

Zhang et al. (2019) ([PubMed](#))

---

|                   |                               |
|-------------------|-------------------------------|
| pick_new_outgroup | <i>Pick Outgroup for Tree</i> |
|-------------------|-------------------------------|

---

**Description**

Pick an outgroup for rooting a phylogenetic tree.

**Usage**

```
pick_new_outgroup(unrooted_tree)
```

**Arguments**

unrooted\_tree    An unrooted tree object.

**Details**

This function preprocess a phylogenetic tree for rooting by the longest edge. Please see this issue for more details <https://github.com/joey711/phyloseq/issues/597>.

**Value**

A new tree with the longest branch as the outgroup.

**See Also**

[data.table](#)

[summary.phylo](#)

Other Phylogenetic Tree Manipulators: [root\\_by\\_longest\\_edge\(\)](#), [root\\_phyloseq\\_tree\(\)](#)

---

|             |                        |
|-------------|------------------------|
| plot_limits | <i>Get Plot Limits</i> |
|-------------|------------------------|

---

**Description**

Get the limits of the plot using data values.

**Usage**

```
plot_limits(x, y)
```

**Arguments**

|   |                               |
|---|-------------------------------|
| x | Data used to plot the x axis. |
| y | Data used to plot the y axis. |

**Value**

A vector that supplies the overarching x and y limits.

**See Also**

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

---

|             |                                                  |
|-------------|--------------------------------------------------|
| raw_silva_2 | <i>"raw_format" Data with 2 Treatment Groups</i> |
|-------------|--------------------------------------------------|

---

### Description

Data from the Nephele-16S-Qiime-Silva microbiome pipeline in the form of a taxmap object. This taxmap object has been formatted into the MicrobiomeR "raw\_format".

### Usage

```
data(raw_silva_2)
```

### Format

An object of class "Taxmap" with customized observation tables:

**otu\_abundance** An abundance table with otu\_id, taxon\_id, and Sample\_\* columns.

**otu\_annotations** A taxonomy annotations table with otu\_id, taxon\_id, and multiple rank columns.

**sample\_data** A metadata table with SampleID, BarcodeSequence, LinkerPrimerSequence, ForwardFastqFile, ReverseFastqFile, TreatmentGroup, SampleName, Description

**phy\_tree** A phylogenetic tree.

### Source

[Nephele-16S Qiime Pipeline](#)

### References

Zhang et al. (2019) ([PubMed](#))

---

|             |                                                  |
|-------------|--------------------------------------------------|
| raw_silva_3 | <i>"raw_format" Data with 3 Treatment Groups</i> |
|-------------|--------------------------------------------------|

---

### Description

Data from the Nephele-16S-Qiime-Silva microbiome pipeline in the form of a taxmap object. This taxmap object has been formatted into the MicrobiomeR "raw\_format".

### Usage

```
data(raw_silva_3)
```

**Format**

An object of class "Taxmap" with customized observation tables:

**otu\_abundance** An abundance table with otu\_id, taxon\_id, and Sample\_\* columns.

**otu\_annotations** A taxonomy annotations table with otu\_id, taxon\_id, and multiple rank columns.

**sample\_data** A metadata table with SampleID, BarcodeSequence, LinkerPrimerSequence, ForwardFastqFile, ReverseFastqFile, TreatmentGroup, SampleName, Description

**phy\_tree** A phylogenetic tree.

**Source**

Nephele-16S Qiime Pipeline

**References**

Zhang et al. (2019) ([PubMed](#))

---

root\_by\_longest\_edge    *Root Phylogenetic Tree*

---

**Description**

This function roots a phylogenetic tree object by it's longest edge.

**Usage**

```
root_by_longest_edge(unrooted_tree)
```

**Arguments**

unrooted\_tree    An unrooted phylogenetic tree object.

**Details**

Please see this issue for more details <https://github.com/joey711/phyloseq/issues/597>.

**Value**

A rooted phlogenetic tree object.

**See Also**

[root\\_pick\\_new\\_outgroup](#)

Other Phylogenetic Tree Manipulators: [pick\\_new\\_outgroup\(\)](#), [root\\_phyloseq\\_tree\(\)](#)

---

|                    |                            |
|--------------------|----------------------------|
| root_phyloseq_tree | <i>Root phyloseq tree.</i> |
|--------------------|----------------------------|

---

## Description

A function for rooting and saving a phylogenetic tree.

## Usage

```
root_phyloseq_tree(
  phyloseq_object,
  tree_path,
  save_rooted_tree,
  recursive = TRUE
)
```

## Arguments

|                  |                                                                                                          |
|------------------|----------------------------------------------------------------------------------------------------------|
| phyloseq_object  | A phyloseq object that contains a phylogenetic tree.                                                     |
| tree_path        | The path to the existing or desired phylogenetic tree file.                                              |
| save_rooted_tree | A logical that determines weather or not the rooted tree is saved.                                       |
| recursive        | If the directory doesn't exist, create the parent directories that don't exist as well,<br>Default: TRUE |

## Details

This function is a helper function to get a proper phyloseq object for downstream analysis. Some analyses require a rooted tree. The function saves the rooted tree in the phyloseq object. It can also save the tree as a file if desired.

## Value

Returns a phyloseq object with a rooted tree.

## See Also

[root](#), [read.tree](#), [write.tree](#)

[phy\\_tree](#)

[fileutils](#)

Other Phylogenetic Tree Manipulators: [pick\\_new\\_outgroup\(\)](#), [root\\_by\\_longest\\_edge\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  #EXAMPLE1
}

## End(Not run)
```

sample\_id\_filter

*Filter Samples Ids from Taxmap Objects***Description**

This function provides a flexible way to filter unwanted samples from the *"otu\_abundance"* and *"sample\_data"* observations of a MicrobiomeR formatted object.

**Usage**

```
sample_id_filter(
  obj,
  .f_transform = NULL,
  .f_filter = NULL,
  .f_condition = NULL,
  validated = FALSE,
  ...
)
```

**Arguments**

|                           |                                                                                                    |
|---------------------------|----------------------------------------------------------------------------------------------------|
| <code>obj</code>          | A Taxmap object.                                                                                   |
| <code>.f_transform</code> | A function used for transforming the data. Default: NULL                                           |
| <code>.f_filter</code>    | A function used for summarising the data like 'sum' or 'mean'. Default: NULL                       |
| <code>.f_condition</code> | A function that takes the summarised data and applied a condition like $x > 10000$ . Default: NULL |
| <code>validated</code>    | This parameter provides a way to override validation steps. Use carefully. Default: FALSE          |
| <code>...</code>          | An optional list of parameters to use in the <code>.f_filter</code> function specified             |

**Details**

Get the samples to keep by using `purrr` and the user supplied transform and filter + condition formulas. The `purrr` package allows the use of anonymous functions as described in the link below:

[https://jennybc.github.io/purrr-tutorial/ls03\\_map-function-syntax.html#anonymous\\_function,\\_formula](https://jennybc.github.io/purrr-tutorial/ls03_map-function-syntax.html#anonymous_function,_formula)

**Value**

Returns a Taxmap object with samples that pass the filters.

**See Also**

[validate\\_MicrobiomeR\\_format](#), [transformer](#)

[get\\_dataset](#)

[select\\_all](#), [select](#), [filter](#)

[map](#), [keep](#)

Other Basic Metacoder Filters: [otu\\_id\\_filter\(\)](#), [taxon\\_id\\_filter\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
# Use the sample_id_filter early on in your analysis
library(MicrobiomeR)
library(metacoder)
library(taxa)
# Convert Phyloseq object to Taxmap object
metacoder_obj <- as_MicrobiomeR_format(obj = phyloseq_silva_2, format = "raw_format")

# Remove Archaea from the Taxmap object
metacoder_obj <- filter_taxa(
  obj = metacoder_obj,
  taxon_names == "Archaea",
  subtaxa = TRUE,
  invert = TRUE)

# Ambiguous Annotation Filter - Remove taxonomies with ambiguous names
metacoder_obj <- filter_ambiguous_taxa(metacoder_obj, subtaxa = TRUE)

# Low Sample Filter - Remove the low samples
metacoder_obj <- sample_id_filter(obj = metacoder_obj,
  .f_filter = ~sum(.),
  .f_condition = ~.>= 20, validated = TRUE)
}

## End(Not run)
```

---

save\_alpha\_diversity\_plots

*Save Alpha Diversity Plots*

---

**Description**

This function saves alpha diversity plots stored in a list object to an output folder.

**Usage**

```
save_alpha_diversity_plots(
  alpha_div_plots,
  format = "tiff",
  start_path = "output",
  ...
)
```

**Arguments**

|                 |                                                                   |
|-----------------|-------------------------------------------------------------------|
| alpha_div_plots | A named list of alpha diversity plots.                            |
| format          | The format of the output image. Default: 'tiff'                   |
| start_path      | The starting path of the output directory. Default: 'output'      |
| ...             | An optional list of parameters to use in the output_dir function. |

**Details**

This function creates an appropriate output directory, where it saves publication ready plots.

**Value**

An output directory that contains alpha diversity plots.

**See Also**

[output\\_dir](#)

[ggsave](#)

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

**Examples**

```
## Not run:
if (interactive()) {
  # This example uses data that are no longer available in the MicrobiomeR package,
  # however, they can be easily generated with \code{\link{MicrobiomeR}{as_analyzed_format}}.
  library(MicrobiomeR)
  analyzed_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "analyzed_format")
  alpha_div_plots <- alpha_diversity_plots(analyzed_silva)
  # Save to \emph{./output/alpha_diversity} folder.
  save_alpha_diversity_plot(alpha_div_plots)
}

## End(Not run)
```

---

save\_correlation\_plots

*Save Correlation Plots*


---

## Description

This function saves correlation plots stored in a listlike object to an output folder.

## Usage

```
save_correlation_plots(corr, format = "tiff", start_path = "output", ...)
```

## Arguments

|            |                                                                                                        |
|------------|--------------------------------------------------------------------------------------------------------|
| corr       | A correlation plot list generated by <code>correlation_plot</code> or <code>correlation_plots</code> . |
| format     | The format of the output image. Default: 'tiff'                                                        |
| start_path | The starting path of the output directory. Default: 'output'                                           |
| ...        | An optional list of parameters to use in the <code>output_dir</code> function.                         |

## Details

This function creates an appropriate output directory, where it saves publication ready plots.

## Value

An output directory that contains correlation plots.

## See Also

[output\\_dir](#)

[ggsave](#)

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

## Examples

```
## Not run:
if(interactive()){
  # This example uses data that are no longer available in the MicrobiomeR package,
  # however, they can be easily generated with \code{\link{MicrobiomeR}{as_analyzed_silva}}.
  library(MicrobiomeR)
  analyzed_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "analyzed_format")
  corr_plot <- correlation_plot(analyzed_silva, primary_rank = "Class", secondary_rank = "Phylum")
  # Save to \emph{./output/corr_plot} folder.
  save_correlation_plots(corr_plot)
```

```

    }

## End(Not run)

```

---

save\_heat\_tree\_plots    *Save Heat Tree Plots*

---

## Description

This function saves heat tree plots stored in a list object to an output folder.

## Usage

```
save_heat_tree_plots(htrees, format = "tiff", start_path = "output", ...)
```

## Arguments

|            |                                                                   |
|------------|-------------------------------------------------------------------|
| htrees     | A named list of heat trees.                                       |
| format     | The format of the output image. Default: 'tiff'                   |
| start_path | The starting path of the output directory. Default: 'output'      |
| ...        | An optional list of parameters to use in the output_dir function. |

## Details

This function creates an appropriate output directory, where it saves publication ready plots.

## Value

An output directory that contains heat tree plots.

## See Also

[output\\_dir](#)

[ggsave](#)

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

## Examples

```
## Not run:
if(interactive()){
  # This example uses data that are no longer available in the MicrobiomeR package,
  # however, they can be easily generated with \code{\link{MicrobiomeR}{as_analyzed_format}}.
  library(MicrobiomeR)
  analyzed_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "analyzed_format")
  h_trees <- heat_tree_plots(analyzed_silva, rank_list = c("Phylum", "Class"))
  # Save to \emph{./output/heat_trees} folder.
  save_heat_tree_plots(h_trees)
}

## End(Not run)
```

---

save\_ordination\_plots *Save Ordination Plots*

---

## Description

This function saves ordination plots stored in a listlike object to an output folder.

## Usage

```
save_ordination_plots(ord_plots, format = "tiff", start_path = "output", ...)
```

## Arguments

|            |                                                                   |
|------------|-------------------------------------------------------------------|
| ord_plots  | An ordination plot list generated by ordination_plots.            |
| format     | The format of the output image. Default: 'tiff'                   |
| start_path | The starting path of the output directory. Default: 'output'      |
| ...        | An optional list of parameters to use in the output_dir function. |

## Details

This function creates an appropriate output directory, where it saves publication ready plots.

## Value

An output directory that contains ordination plots.

## See Also

[output\\_dir](#)

[ggsave](#)

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

## Examples

```
## Not run:  
if (interactive()) {  
}  
  
## End(Not run)
```

---

save\_stacked\_barplots *Save Stacked Barplots*

---

## Description

This function saves stacked barplot stored in a list object to an output folder.

## Usage

```
save_stacked_barplots(sb_plots, format = "tiff", start_path = "output", ...)
```

## Arguments

|            |                                                                   |
|------------|-------------------------------------------------------------------|
| sb_plots   | A named list of stacked barplots.                                 |
| format     | The format of the output image. Default: 'tiff'                   |
| start_path | The starting path of the output directory. Default: 'output'      |
| ...        | An optional list of parameters to use in the output_dir function. |

## Details

This function creates an appropriate output directory, where it saves publication ready plots.

## Value

An output directory that contains stacked barplot.

## See Also

[output\\_dir](#)

[ggsave](#)

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

**Examples**

```
## Not run:
if (interactive()) {
  # This example uses data that are no longer available in the MicrobiomeR package,
  # however, they can be easily generated with \code{\link{MicrobiomeR}{as_analyzed_format}}.
  library(MicrobiomeR)
  analyzed_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "analyzed_format")
  alpha_div_plots <- alpha_diversity_plots(analyzed_silva)
  # Save to \emph{./output/alpha_diversity} folder.
  save_alpha_diversity_plot(alpha_div_plots)
}

## End(Not run)
```

---

```
save_top_coefficients_barplot
      Save Top Coefficients Barplot
```

---

**Description**

Save a top coefficients barplot.

**Usage**

```
save_top_coefficients_barplot(
  plot,
  filename,
  format = "tiff",
  start_path = "output",
  ...
)
```

**Arguments**

|            |                                                             |
|------------|-------------------------------------------------------------|
| plot       | The plot object.                                            |
| filename   | The name of the file. (an extension should not be included) |
| format     | The file format, Default = "tiff"                           |
| start_path | The path of which your output should be saved.              |
| ...        | Additional parameters.                                      |

**Examples**

```
## Not run:
if (interactive()) {
  library(MicrobiomeR)
  save_top_coefficients_barplot(plot = plot, filename = "stacked_bar_phylum")
}

## End(Not run)
```

---

|               |                               |
|---------------|-------------------------------|
| scico_palette | <i>Scico Palette Function</i> |
|---------------|-------------------------------|

---

**Description**

A function that returns a color palette function based off of the scico package.

**Usage**

```
scico_palette(  
  scico_palette = "batlow",  
  scico_number = 800,  
  scico_range = 0:scico_number  
)
```

**Arguments**

|               |                                                                    |
|---------------|--------------------------------------------------------------------|
| scico_palette | The scico palette to use. Default: 'batlow'                        |
| scico_number  | The number of colors to use in the scico palette. Default: 800     |
| scico_range   | The range in the color palette to use. Default: c(0, scico_number) |

**Details**

The purpose of this function is to provide an interpolated scico palette for using with the `get_color_palette` function.

**Value**

The output of this function is another function (`grDevices::colorRampPalette`), which takes a number to generate an interpolated color palette as a character vector.

**See Also**

[colorRamp scico](#)

Other Color Palettes: [combination\\_palette\(\)](#), [get\\_color\\_palette\(\)](#), [viridis\\_magma\\_palette\(\)](#), [viridis\\_palette\(\)](#)

**Examples**

```
## Not run:  
if(interactive()){  
  library(MicrobiomeR)  
  basic_pal <- scico_palette(scico_palette="hawaii")(25)  
  better_pal <- get_color_palette(pal_func = scico_palette, color_no = 25)  
}  
  
## End(Not run)
```

---

|                 |                        |
|-----------------|------------------------|
| stacked_barplot | <i>Stacked Barplot</i> |
|-----------------|------------------------|

---

## Description

Create a stacked barplot to show relative abundance of taxa. `convert_proportions` converts the dataframe abundance values to percent 100 and returns a transformed dataframe. `melt_metacoder` melts the metacoder or phyloseq tables into a dataframe and returns a melted dataframe. `stacked_barplots` creates a stacked barplots for multiple taxonomic levels and returns a list of stacked barplots.

## Usage

```
stacked_barplot(
  obj,
  tax_level = "Phylum",
  fill = "Phylum",
  xlabel = "Samples",
  faceted = FALSE,
  title = NULL,
  palette_values = NULL
)
```

## Arguments

|                             |                                                                                   |
|-----------------------------|-----------------------------------------------------------------------------------|
| <code>obj</code>            | An object to be converted to a taxmap object with <a href="#">create_taxmap</a> . |
| <code>tax_level</code>      | The taxonomic level, Default: 'Phylum'                                            |
| <code>fill</code>           | The taxonomic level by which the bars are filled, Default: 'Phylum'               |
| <code>xlabel</code>         | The label of the x axis, Default: 'Samples'                                       |
| <code>faceted</code>        | A boolean to determine if the barplot should be faceted by TreatmentGroup         |
| <code>title</code>          | The title or name of the plot.                                                    |
| <code>palette_values</code> | A list of the colors to input to be mapped to the plot palette, Default: 'NULL'   |

## Value

Returns a stacked barplot.

## See Also

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplots\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

Examples

```
## Not run:
if (interactive()) {
  # An example stacked bar plot
  library(MicrobiomeR)
  data <- analyzed_silva
  palette <- get_color_palette(color_no = 12)
  plot <- stacked_barplot(obj = data, palette_values = palette)
  plot
}

## End(Not run)
```

---

|                  |                         |
|------------------|-------------------------|
| stacked_barplots | <i>Stacked Barplots</i> |
|------------------|-------------------------|

---

Description

Stacked Barplots

Usage

```
stacked_barplots(obj, tax_levels = c("Phylum", "Class", "Order"))
```

Arguments

- obj                   An object to be converted to a taxmap object with [create\\_taxmap](#).
- tax\_levels           The taxonomic levels, Default: 'c("Phylum", "Class", "Order")'

See Also

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [top\\_coefficients\\_barplot\(\)](#)

---

|                        |                                                |
|------------------------|------------------------------------------------|
| taxa_prevalence_filter | <i>Taxonomic Prevalence Filter (Metacoder)</i> |
|------------------------|------------------------------------------------|

---

Description

This function filters observations at a specific rank by their prevalence across samples.

## Usage

```
taxa_prevalence_filter(  
  obj,  
  rank,  
  minimum_abundance = 5,  
  rel_sample_percentage = 0.5,  
  validated = FALSE  
)
```

## Arguments

|                       |                                                                                           |
|-----------------------|-------------------------------------------------------------------------------------------|
| obj                   | A Taxmap object.                                                                          |
| rank                  | The rank being analyzed for prevalence across samples.                                    |
| minimum_abundance     | The minimum abundance needed per observation per sample. Default: 5                       |
| rel_sample_percentage | The percentage of samples per observation that meet the minimum abundance. Default: 0.5   |
| validated             | This parameter provides a way to override validation steps. Use carefully. Default: FALSE |

## Details

The `taxa_prevalence_filter` filters `taxon_ids` that do not appear more than a certain amount of times (minimum abundance) in a certain percentage of samples (`rel_sample_percentage`) at the specified agglomerated rank (`rank`). The [phyloseq workflow](#) calls for a minimum abundance of 5 across %50 of the samples. This method is considered supervised, because the filtering is done based on taxonomic annotation (`taxon_ids`), which is assigned based on a reference database (SILVA or GreenGenes).

## Value

Returns a taxmap object that contains `taxon_ids` that have passed the above filter.

## See Also

[validate\\_MicrobiomeR\\_format](#)

[calc\\_prop\\_samples](#)

[filter](#)

[filter\\_taxa](#)

Other Advanced Metacoder Filters: [agglomerate\\_taxmap\(\)](#), [cov\\_filter\(\)](#), [otu\\_prevalence\\_filter\(\)](#), [otu\\_proportion\\_filter\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  library(MicrobiomeR)
  library(metacoder)
  library(taxa)

  # Convert Phyloseq object to taxmap object
  metacoder_obj <- as_MicrobiomeR_format(obj = phyloseq_obj, format = "raw_format")

  # Remove Archaea from the taxmap object
  metacoder_obj <- filter_taxa(
    obj = metacoder_obj,
    taxon_names == "Archaea",
    subtaxa = TRUE,
    invert = TRUE)

  # Ambiguous Annotation Filter - Remove taxonomies with ambiguous names
  metacoder_obj <- filter_ambiguous_taxa(metacoder_obj, subtaxa = TRUE)

  # Low Sample Filter - Remove the low samples
  metacoder_obj <- sample_id_filter(obj = metacoder_obj,
    .f_filter = ~sum(.),
    .f_condition = ~.>= 20, validated = TRUE)

  # Master Threshold Filter - Add the otu_proportions table and then filter OTUs based on min %
  metacoder_obj <- otu_proportion_filter(
    obj = metacoder_obj,
    otu_percentage = 0.00001
  )

  # Taxa Prevalence Filter
  # The default minimum abundance is 5 and the sample percentage is 0.5 (5%).
  # Phylum
  metacoder_obj <- taxa_prevalence_filter(
    obj = metacoder_obj,
    rank = "Phylum"
  )

  # Class
  metacoder_obj <- taxa_prevalence_filter(
    obj = metacoder_obj,
    rank = "Class",
    validated = TRUE
  )
}

## End(Not run)
```

## Description

This function provides a flexible way to filter unwanted taxon\_ids from the taxmap object and from the observations of a MicrobiomeR formatted object.

## Usage

```
taxon_id_filter(
  obj,
  .f_transform = NULL,
  .f_filter = NULL,
  .f_condition = NULL,
  validated = FALSE,
  ...
)
```

## Arguments

|              |                                                                                                    |
|--------------|----------------------------------------------------------------------------------------------------|
| obj          | A Taxmap object.                                                                                   |
| .f_transform | A function used for transforming the data. Default: NULL                                           |
| .f_filter    | A function used for summarising the data like 'sum' or 'mean'. Default: NULL                       |
| .f_condition | A function that takes the summarised data and applied a condition like $x > 10000$ . Default: NULL |
| validated    | This parameter provides a way to override validation steps. Use carefully. Default: FALSE          |
| ...          | An optional list of parameters to use in the .f_filter function specified                          |

## Details

Get the taxon\_ids to keep by using purr and the user supplied transform and filter + condition formulas. The purr package allows the use of anonymous functions as described in the link below:

[https://jennybc.github.io/purrr-tutorial/ls03\\_map-function-syntax.html#anonymous\\_function,\\_formula](https://jennybc.github.io/purrr-tutorial/ls03_map-function-syntax.html#anonymous_function,_formula)

## Value

Returns a Taxmap object with taxon\_ids that pass the filters.

## See Also

[validate\\_MicrobiomeR\\_format](#), [transposer](#), [transformer](#)

[get\\_dataset](#), [filter\\_taxa](#)

[select\\_all](#)

[map](#), [keep](#)

Other Basic Metacoder Filters: [otu\\_id\\_filter\(\)](#), [sample\\_id\\_filter\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  library(MicrobiomeR)
  library(metacoder)
  library(taxa)
  # Convert Phyloseq object to Taxmap object
  metacoder_obj <- as_MicrobiomeR_format(obj = phyloseq_obj, format = "raw_format")

  # Remove Archaea from the Taxmap object
  metacoder_obj <- filter_taxa(
    obj = metacoder_obj,
    taxon_names == "Archaea",
    subtaxa = TRUE,
    invert = TRUE)

  # Ambiguous Annotation Filter - Remove taxonomies with ambiguous names
  metacoder_obj <- filter_ambiguous_taxa(metacoder_obj, subtaxa = TRUE)

  # Low taxa filter - Remove the low taxon_ids
  metacoder_obj <- taxon_id_filter(obj = metacoder_obj,
    .f_filter = ~sum(.),
    .f_condition = ~.>= 2000, validated = TRUE)
}

## End(Not run)
```

---

top\_coefficients\_barplot

*Top Coefficients Barplot*


---

**Description**

A barplot to visualize which taxa have had the greatest impact on the overall gut composition.

**Usage**

```
top_coefficients_barplot(top_coefficients, title = NULL)
```

**Arguments**

```
top_coefficients      Coefficients for the top taxa separating the groups.
title                 The title of the plot, Default = NULL
```

**Value**

Returns a barplot of the top coefficients

**See Also**

View [permanova](#) to understand how to generate top\_coefficients

Other Visualizations: [alpha\\_diversity\\_plot\(\)](#), [correlation\\_data\(\)](#), [correlation\\_plot\(\)](#), [correlation\\_plots\(\)](#), [heat\\_tree\\_parameters\(\)](#), [heat\\_tree\\_plots\(\)](#), [ordination\\_plot\(\)](#), [plot\\_limits\(\)](#), [save\\_alpha\\_diversity\\_plots\(\)](#), [save\\_correlation\\_plots\(\)](#), [save\\_heat\\_tree\\_plots\(\)](#), [save\\_ordination\\_plots\(\)](#), [save\\_stacked\\_barplots\(\)](#), [stacked\\_barplot\(\)](#), [stacked\\_barplots\(\)](#)

**Examples**

```
## Not run:
if (interactive()) {
  library(MicrobiomeR)
  data <- analyzed_silva
  p <- permanova(obj = data)
  top_coefficients_barplot(p$top_coefficients)
}

## End(Not run)
```

transformer

*Transforming Tidy Data***Description**

This function transforms a table by rows or by columns.

**Usage**

```
transformer(
  .data,
  func,
  by = "column",
  ids = NULL,
  header_name = NULL,
  preserved_categories = TRUE,
  separated_categories = NULL,
  ...
)
```

**Arguments**

|       |                                                                                                               |
|-------|---------------------------------------------------------------------------------------------------------------|
| .data | A matrix/data_frame/tibble for transforming.                                                                  |
| func  | A function, which can be anonymous, that will be used to transform the data. (e.g. proportions: $x/sum(x)$ ). |
| by    | This denotes how the data should be transformed ( <b>column/row</b> ). Default: 'column'                      |

|                                   |                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>ids</code>                  | The column to transpose by if <code>by = 'row'</code> . Default: The first column.                                                                                                                                                                                                                                                                                  |
| <code>header_name</code>          | A name for the numeric data that will be transposed if <code>by = 'row'</code> .                                                                                                                                                                                                                                                                                    |
| <code>preserved_categories</code> | A logical denoting whether categorical data should be conserved if <code>by = 'row'</code> .<br>A value of <code>FALSE</code> will cause all categorical data except the <code>ids</code> to be dropped.<br>A value of <code>TRUE</code> will cause the categorical data to be preserved by <code>tidyr::unite</code> ing these columns. Default: <code>TRUE</code> |
| <code>separated_categories</code> | A vector containing ordered column names to use in a previously transposed and categorically preserved table if <code>by = 'row'</code> . Retransposing with this set should yield an exact replicate of the original data. Default: <code>NULL</code>                                                                                                              |
| <code>...</code>                  | Additional arguments passed on to <code>func</code> .                                                                                                                                                                                                                                                                                                               |

### Details

This function transforms the supplied data using the `func` parameter, which is used in the `purrr` package. The `purrr` package allows the use of anonymous functions as described in the link below:

[https://jennybc.github.io/purrr-tutorial/ls03\\_map-function-syntax.html#anonymous\\_function,\\_formula](https://jennybc.github.io/purrr-tutorial/ls03_map-function-syntax.html#anonymous_function,_formula)

### Value

A tibble that has been transformed.

### See Also

[transposer](#), [select\\_all](#), [select\\_modify](#)

Other Data Manipulators: [convert\\_proportions\(\)](#), [metacoder\\_comp\\_func\\_1\(\)](#), [transposer\(\)](#), [vlookup\(\)](#)

### Examples

```
## Not run:
if(interactive()){
  # This example uses data that are no longer available in the MicrobiomeR package,
  # however, they can be easily generated with \code{\link{MicrobiomeR}{as_basic_format}}.
  library(MicrobiomeR)
  basic_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "basic_format")
  data <- basic_silva$data$taxa_abundance
  # Get proportions using the anonymous functions
  tax_props <- data %>% transformer(~./sum(.))
  # Get proportions using explicit functions
  alt_tax_props <- data %>% transformer(function(x)x/sum(x))
}

## End(Not run)
```

---

transposer

---

*Transposing Tidy Data*

---

## Description

This function transposes tables containing numeric and categorical data using the tidyr package.

## Usage

```
transposer(  
  .data,  
  ids = NULL,  
  header_name,  
  preserved_categories = TRUE,  
  separated_categories = NULL  
)
```

## Arguments

|                                   |                                                                                                                                                                                                                                                                                |
|-----------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>.data</code>                | A matrix/data_frame/tibble for transposing.                                                                                                                                                                                                                                    |
| <code>ids</code>                  | The column to transpose by. Default: The first column.                                                                                                                                                                                                                         |
| <code>header_name</code>          | A name for the numeric data that will be transposed.                                                                                                                                                                                                                           |
| <code>preserved_categories</code> | A logical denoting whether categorical data should be conserved. A value of FALSE will cause all categorical data except the <i>ids</i> to be dropped. A value of TRUE will cause the categorical data to be preserved by <i>tidyr::unite</i> ing these columns. Default: TRUE |
| <code>separated_categories</code> | A vector containing ordered column names to use in a previously transposed and categorically preserved table. Retransposing with this set should yield an exact replicate of the original data. Default: NULL                                                                  |

## Details

Transposing can help with performing operations on the rows of your tibbles.

## Value

A transposed data table as a tibble.

## See Also

[is\\_tibble](#)  
[select\\_all](#), [select](#), [reexports](#), [sym](#), [one\\_of](#)  
[gather](#), [unite](#), [spread](#), [separate](#)  
[str\\_detect](#), [str\\_count](#)

Other Data Manipulators: [convert\\_proportions\(\)](#), [metacoder\\_comp\\_func\\_1\(\)](#), [transformer\(\)](#), [vlookup\(\)](#)

## Examples

```
## Not run:
if(interactive()){
# This example uses data that are no longer available in the MicrobiomeR package,
# however, they can be easily generated with \code{\link{MicrobiomeR}{as_basic_format}}.
library(MicrobiomeR)
basic_silva <- as_MicrobiomeR_format(MicrobiomeR::raw_silva_2, "basic_format")
data <- basic_silva$data$taxa_abundance
trans_data <- data %>%
  transposer(ids = "taxon_id", header_name = "samples")
retrans_data <- trans_data %>%
  transposer(ids="samples", header_name="taxon_id")
}

## End(Not run)
```

---

treatment\_matrix

*Get Treatment Matrix*


---

## Description

A function that returns a matrix with used for comparing treatment data.

## Usage

```
treatment_matrix(obj)
```

## Arguments

obj                      An object to be converted to a taxmap object with [create\\_taxmap](#).

## Details

Use this when you want to do pairwise comparisons.

## Value

A matrix with each column representing a comparison to be made.

## See Also

Other Formatting: [MicrobiomeR\\_Formats](#), [as\\_MicrobiomeR\\_format\(\)](#), [as\\_analyzed\\_format\(\)](#), [as\\_basic\\_format\(\)](#), [as\\_custom\\_format\(\)](#), [as\\_phyloseq\\_format\(\)](#), [as\\_raw\\_format\(\)](#), [melt\\_taxmap\(\)](#)

## Examples

```
## Not run:
if(interactive()){
  #EXAMPLE1
}

## End(Not run)
```

---

```
validate_MicrobiomeR_format
      Validate MicrobiomeR Format
```

---

## Description

This function validates that Taxmap objects are in a valid format MicrobiomeR format.

## Usage

```
validate_MicrobiomeR_format(
  obj,
  validated = FALSE,
  valid_formats,
  force_format = FALSE,
  min_or_max = base::max,
  ...
)
```

## Arguments

|                            |                                                                                                                                                                                                                                                                                |
|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>obj</code>           | A Taxmap object.                                                                                                                                                                                                                                                               |
| <code>validated</code>     | This parameter provides a way to override validation steps. Use carefully. Default: FALSE                                                                                                                                                                                      |
| <code>valid_formats</code> | A vector of formats that are used for validation.                                                                                                                                                                                                                              |
| <code>force_format</code>  | A logical denoting if the selected format is to be forced. Default: FALSE                                                                                                                                                                                                      |
| <code>min_or_max</code>    | A base function (min or max) that determines which format is to be forced. This is particularly useful if you provide multiple <i>valid_formats</i> . Min will choose the lowest level format, while max will choose the highest level format. Default: <code>base::max</code> |
| <code>...</code>           | An optional list of parameters to use in <code>as_MicrobiomeR_format</code> .                                                                                                                                                                                                  |

## Details

This function can provide a way to check if a taxmap object has undergone a [MicrobiomeR Style Workflow](#).

**Value**

If the object is validated, a Taxmap object.

**See Also**

[which\\_format](#), [as\\_MicrobiomeR\\_format](#)

Other Validation: [is\\_analyzed\\_format\(\)](#), [is\\_basic\\_format\(\)](#), [is\\_phyloseq\\_format\(\)](#), [is\\_raw\\_format\(\)](#), [order\\_metacoder\\_data\(\)](#), [which\\_format\(\)](#)

---

viridis\_magma\_palette *Viridis-Magma Palette Function*

---

**Description**

A function that returns a color palette function based off of the viridis package.

**Usage**

```
viridis_magma_palette(
  viridis_number = 800,
  viridis_range = 300:viridis_number,
  viridis_rev = TRUE,
  magma_number = 500,
  magma_range = 50:magma_number,
  magma_rev = FALSE,
  ...
)
```

**Arguments**

|                |                                                                                   |
|----------------|-----------------------------------------------------------------------------------|
| viridis_number | The total number of colors used to generate the viridis palette. Default: 800     |
| viridis_range  | The range of colors in the viridis palette to use. Default: 300:viridis_number    |
| viridis_rev    | A logical for reversing the viridis palette. Default: TRUE                        |
| magma_number   | The total number of colors used to generate the magma palette. Default: 500       |
| magma_range    | The range of colors in the magma palette to use. Default: 0:magma_number          |
| magma_rev      | A logical for reversing the magma palette. Default: FALSE                         |
| ...            | These dots are optionally used as both the magma and viridis function parameters. |

**Details**

The primary purpose of this function is to return a palette-function for generating viridis style color palettes. By taking the `viridis::viridis()` and the `viridis::magma()` colors, and manipulating them, this function can help create a unique set of colors that you can distinguish on a busy plot. The hopes of this function is to help improve plots that use more than 20 colors. Use the provided example to view the color palette.

**Value**

The output of this function is another function (`grDevices::colorRampPalette`), which takes a number to generate an interpolated color palette as a character vector.

**See Also**

[combination\\_palette](#)

[viridis](#), [magma](#)

[colorRamp](#)

Other Color Palettes: [combination\\_palette\(\)](#), [get\\_color\\_palette\(\)](#), [scico\\_palette\(\)](#), [viridis\\_palette\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  # Use the default values
  > pal_func <- viridis_palette()

  # Get a palette with 20 colors
  > pal <- pal_func(20)

  # Make a pie plot of the colors.
  > pie(rep(1, length(pal)), col=pal)
}

## End(Not run)
```

---

viridis\_palette

*Viridis Palette Function*

---

**Description**

A function that returns a color palette function based off of the viridis package.

**Usage**

```
viridis_palette(
  viridis_palette = "viridis",
  viridis_number = 800,
  viridis_range = 0:viridis_number
)
```

**Arguments**

viridis\_palette

The viridis palette to use. Default: 'viridis'

viridis\_number The number of colors to use in the viridis palette. Default: 800

viridis\_range The range in the color palette to use. Default: c(0, viridis\_number)

**Details**

The purpose of this function is to provide an interpolated viridis palette for using with the `get_color_palette` function.

**Value**

The output of this function is another function (`grDevices::colorRampPalette`), which takes a number to generate an interpolated color palette as a character vector.

**See Also**

[colorRamp](#)

[viridis](#)

Other Color Palettes: [combination\\_palette\(\)](#), [get\\_color\\_palette\(\)](#), [scico\\_palette\(\)](#), [viridis\\_magma\\_palette\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  library(MicrobiomeR)
  basic_pal <- viridis::viridis(n=25, option="magma")
  better_pal <- get_color_palette(pal_func = viridis_palette, color_no = 25)
}

## End(Not run)
```

---

vlookup

---

*Mock Excel "VlookUp" Function*


---

**Description**

A function that mimics excels vlookup, but for R's dataframe.

**Usage**

```
vlookup(lookup_vector, df, match_var, return_var)
```

**Arguments**

|                            |                                             |
|----------------------------|---------------------------------------------|
| <code>lookup_vector</code> | A vector of items to look up.               |
| <code>df</code>            | The dataframe to search.                    |
| <code>match_var</code>     | The column name to search in the dataframe. |
| <code>return_var</code>    | The column of data to return when matched.  |

**Details**

A function that works like the VLOOKUP function in excel. This function was borrowed from <https://www.r-bloggers.com/an-r-vlookup-not-so-silly-idea/>.

**Value**

A vector that contains the items of interest.

**See Also**

Other Data Manipulators: [convert\\_proportions\(\)](#), [metacoder\\_comp\\_func\\_1\(\)](#), [transformer\(\)](#), [transposer\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  #EXAMPLE1
}

## End(Not run)
```

---

which\_format

*Which MicrobiomeR Format*

---

**Description**

A function for looking at a Taxmap object and returning the identified MicrobiomeR format.

**Usage**

```
which_format(obj)
```

**Arguments**

obj                    A Taxmap object.

**Details**

This function is used to get basic information about the format of the taxmap object that is supplied by the user.

**Value**

If the format is verified it returns a character string denoting the identified format.

**See Also**

Other Validation: [is\\_analyzed\\_format\(\)](#), [is\\_basic\\_format\(\)](#), [is\\_phyloseq\\_format\(\)](#), [is\\_raw\\_format\(\)](#), [order\\_metacoder\\_data\(\)](#), [validate\\_MicrobiomeR\\_format\(\)](#)

**Examples**

```
## Not run:
if(interactive()){
  library(MicrobiomeR)
  fmt <- which_fmt(MicrobiomeR::raw_silva_2)
  print(fmt)
}

## End(Not run)
```

# Index

## \* Advanced Metacoder Filters

- agglomerate\_taxmap, 3
- cov\_filter, 17
- otu\_prevalence\_filter, 36
- otu\_proportion\_filter, 38
- taxa\_prevalence\_filter, 59

## \* Basic Metacoder Filters

- otu\_id\_filter, 34
- sample\_id\_filter, 49
- taxon\_id\_filter, 61

## \* Color Palettes

- combination\_palette, 12
- get\_color\_palette, 22
- scico\_palette, 57
- viridis\_magma\_palette, 69
- viridis\_palette, 70

## \* Data Importers

- create\_phyloseq, 19
- create\_taxmap, 21

## \* Data Manipulators

- convert\_proportions, 13
- metacoder\_comp\_func\_1, 30
- transformer, 64
- transposer, 66
- vlookup, 71

## \* Formatting

- as\_analyzed\_format, 7
- as\_basic\_format, 8
- as\_custom\_format, 9
- as\_MicrobiomeR\_format, 10
- as\_phyloseq\_format, 10
- as\_raw\_format, 11
- melt\_taxmap, 30
- MicrobiomeR\_Formats, 31
- treatment\_matrix, 67

## \* Phylogenetic Tree Manipulators

- pick\_new\_outgroup, 44
- root\_by\_longest\_edge, 47
- root\_phyloseq\_tree, 48

## \* Project Management

- output\_dir, 39

## \* Validation

- is\_analyzed\_format, 26
- is\_basic\_format, 27
- is\_phyloseq\_format, 28
- is\_raw\_format, 29
- order\_metacoder\_data, 32
- validate\_MicrobiomeR\_format, 68
- which\_format, 72

## \* Visualizations

- alpha\_diversity\_plot, 5
- correlation\_data, 14
- correlation\_plot, 15
- correlation\_plots, 16
- heat\_tree\_parameters, 24
- heat\_tree\_plots, 25
- ordination\_plot, 33
- plot\_limits, 45
- save\_alpha\_diversity\_plots, 50
- save\_correlation\_plots, 52
- save\_heat\_tree\_plots, 53
- save\_ordination\_plots, 54
- save\_stacked\_barplots, 55
- stacked\_barplot, 58
- stacked\_barplots, 59
- top\_coefficients\_barplot, 63

## \* datasets

- phyloseq\_silva\_2, 43
- phyloseq\_silva\_3, 44
- raw\_silva\_2, 46
- raw\_silva\_3, 46

adonis2, 43

aes, 16

agglomerate\_taxmap, 3, 14, 18, 37, 38, 60

alpha, 5

alpha\_diversity\_measures, 4, 6

alpha\_diversity\_plot, 5, 14, 16, 17, 24, 25,  
34, 45, 51–55, 58, 59, 64

- alpha\_diversity\_plots
  - (alpha\_diversity\_plot), 5
- arrange, 14
- as\_analyzed\_format, 7, 8–12, 30, 31, 67
- as\_basic\_format, 8, 8–12, 30, 31, 67
- as\_custom\_format, 8, 9, 10–12, 30, 31, 67
- as\_MicrobiomeR\_format, 8, 9, 10, 11, 12, 30, 31, 67–69
- as\_phyloseq\_format, 8, 9, 10, 10, 12, 30, 31, 67
- as\_raw\_format, 8–10, 11, 11, 30, 31, 67
- betadisper, 43
- calc\_obs\_props, 7, 8
- calc\_prop\_samples, 37, 60
- calc\_taxon\_abund, 7, 8
- case, 40
- colorRamp, 13, 23, 57, 70, 71
- combination\_palette, 12, 23, 57, 70, 71
- compare\_groups, 7, 8
- convert\_proportions, 13, 31, 65, 67, 72
- correlation\_data, 6, 14, 16, 17, 24, 25, 34, 45, 51–55, 58, 59, 64
- correlation\_plot, 6, 14, 15, 17, 24, 25, 34, 45, 51–55, 58, 59, 64
- correlation\_plots, 6, 14, 16, 16, 24, 25, 34, 45, 51–55, 58, 59, 64
- cov\_filter, 4, 17, 37, 38, 60
- crayon, 25
- create\_phyloseq, 19, 22, 31, 32
- create\_pub\_table, 21
- create\_taxmap, 5–11, 14–16, 20, 21, 25, 30, 32, 33, 43, 58, 59, 67
- data.table, 45
- dip.test, 31
- dist, 33
- distance, 33
- diversity, 6
- enquos, 24
- eval\_tidy, 24
- fct\_reorder, 16
- fileutils, 22, 48
- filter, 14, 37, 50, 60
- filter\_obs, 25
- filter\_taxa, 4, 35, 37, 60, 62
- gather, 66
- geom\_abline, 16
- geom\_label\_repel, 16
- geom\_point, 16
- geom\_polygon, 16
- get\_color\_palette, 13, 16, 22, 57, 70, 71
- get\_dataset, 35, 50, 62
- ggplot, 16
- ggsave, 51–55
- ggviolin, 6
- glue, 25, 40
- guide\_legend, 16
- heat\_tree, 24, 25
- heat\_tree\_parameters, 6, 14, 16, 17, 24, 25, 34, 45, 51–55, 58, 59, 64
- heat\_tree\_plots, 6, 14, 16, 17, 24, 25, 34, 45, 51–55, 58, 59, 64
- import\_biom, 20, 41, 42
- import\_qiime, 41, 42
- import\_qiime\_sample\_data, 20
- is.rooted, 20
- is\_analyzed\_format, 8, 26, 27–29, 33, 69, 72
- is\_basic\_format, 8, 26, 27, 28, 29, 33, 69, 72
- is\_phyloseq\_format, 8, 12, 26, 27, 28, 29, 33, 69, 72
- is\_quosure, 24
- is\_raw\_format, 8, 12, 26–28, 29, 33, 69, 72
- is\_tibble, 66
- keep, 35, 50, 62
- labs, 16, 25
- list\_modify, 24
- magma, 70
- map, 35, 50, 62
- margin, 25
- melt\_taxmap, 8–12, 30, 31, 67
- merge\_phyloseq, 20
- meta, 5
- metacoder::parse\_phyloseq, 32
- metacoder::taxmap, 32
- metacoder\_comp\_func\_1, 14, 30, 65, 67, 72
- MicrobiomeR\_Formats, 7–12, 26–30, 31, 32, 67
- MicrobiomeR\_Workflow, 31, 32
- modify, 65

- mutate, [14](#)
- n\_obs, [24](#)
- one\_of, [66](#)
- order\_metacoder\_data, [8–12](#), [26–29](#), [32](#), [69](#), [72](#)
- ordination\_plot, [6](#), [14](#), [16](#), [17](#), [24](#), [25](#), [33](#), [45](#), [51–55](#), [58](#), [59](#), [64](#)
- ordination\_plots(ordination\_plot), [33](#)
- otu\_id\_filter, [18](#), [34](#), [38](#), [50](#), [62](#)
- otu\_prevalence\_filter, [4](#), [18](#), [36](#), [38](#), [60](#)
- otu\_proportion\_filter, [4](#), [18](#), [37](#), [38](#), [60](#)
- output\_dir, [39](#), [51–55](#)
- parse\_phyloseq, [22](#)
- parse\_taxonomy\_default, [42](#)
- parse\_taxonomy\_greengenes, [42](#)
- parse\_taxonomy\_qiime, [42](#)
- parse\_taxonomy\_silva\_128, [41](#)
- permanova, [42](#), [64](#)
- phy\_tree, [48](#)
- phyloseq::otu\_table, [32](#)
- phyloseq::phy\_tree, [32](#)
- phyloseq::phy\_tree(), [20](#)
- phyloseq::sample\_data, [32](#)
- phyloseq::sample\_data(), [20](#)
- phyloseq::tax\_glom, [3](#)
- phyloseq::tax\_table, [32](#)
- phyloseq\_silva\_2, [43](#)
- phyloseq\_silva\_3, [44](#)
- pick\_new\_outgroup, [44](#), [47](#), [48](#)
- plot\_limits, [6](#), [14](#), [16](#), [17](#), [24](#), [25](#), [34](#), [45](#), [51–55](#), [58](#), [59](#), [64](#)
- raw\_silva\_2, [46](#)
- raw\_silva\_3, [46](#)
- read.tree, [20](#), [48](#)
- reexports, [66](#)
- root, [20](#), [47](#), [48](#)
- root\_by\_longest\_edge, [20](#), [45](#), [47](#), [48](#)
- root\_phyloseq\_tree, [45](#), [47](#), [48](#)
- sample\_id\_filter, [35](#), [49](#), [62](#)
- save\_alpha\_diversity\_plots, [6](#), [14](#), [16](#), [17](#), [24](#), [25](#), [34](#), [45](#), [50](#), [52–55](#), [58](#), [59](#), [64](#)
- save\_correlation\_plots, [6](#), [14](#), [16](#), [17](#), [24](#), [25](#), [34](#), [45](#), [51](#), [52](#), [53–55](#), [58](#), [59](#), [64](#)
- save\_heat\_tree\_plots, [6](#), [14](#), [16](#), [17](#), [24](#), [25](#), [34](#), [45](#), [51](#), [52](#), [53](#), [54](#), [55](#), [58](#), [59](#), [64](#)
- save\_ordination\_plots, [6](#), [14](#), [16](#), [17](#), [24](#), [25](#), [34](#), [45](#), [51–53](#), [54](#), [55](#), [58](#), [59](#), [64](#)
- save\_stacked\_barplots, [6](#), [14](#), [16](#), [17](#), [24](#), [25](#), [34](#), [45](#), [51–54](#), [55](#), [58](#), [59](#), [64](#)
- save\_top\_coefficients\_barplot, [56](#)
- scale\_continuous, [15](#), [16](#)
- scale\_manual, [16](#)
- scico, [57](#)
- scico\_palette, [13](#), [23](#), [57](#), [70](#), [71](#)
- select, [50](#), [65](#), [66](#)
- select\_all, [35](#), [50](#), [62](#), [65](#), [66](#)
- separate, [66](#)
- spread, [66](#)
- stacked\_barplot, [6](#), [14](#), [16](#), [17](#), [24](#), [25](#), [34](#), [45](#), [51–55](#), [58](#), [59](#), [64](#)
- stacked\_barplots, [6](#), [14](#), [16](#), [17](#), [24](#), [25](#), [34](#), [45](#), [51–55](#), [58](#), [59](#), [64](#)
- str\_count, [66](#)
- str\_detect, [66](#)
- summarise\_if, [18](#)
- summary.phylo, [45](#)
- sym, [66](#)
- taxa\_prevalence\_filter, [4](#), [18](#), [37](#), [38](#), [59](#)
- taxon\_id\_filter, [35](#), [50](#), [61](#)
- taxon\_ids, [8](#)
- taxon\_names, [24](#)
- taxonomy\_table, [8](#)
- theme, [25](#)
- top\_coefficients\_barplot, [6](#), [14](#), [16](#), [17](#), [24](#), [25](#), [34](#), [43](#), [45](#), [51–55](#), [58](#), [59](#), [63](#)
- transformer, [14](#), [31](#), [35](#), [50](#), [62](#), [64](#), [67](#), [72](#)
- transposer, [14](#), [31](#), [35](#), [62](#), [65](#), [66](#), [72](#)
- treatment\_matrix, [8–12](#), [30](#), [31](#), [67](#)
- unite, [66](#)
- validate\_MicrobiomeR\_format, [4](#), [16](#), [18](#), [25–29](#), [33](#), [35](#), [37](#), [38](#), [50](#), [60](#), [62](#), [68](#), [72](#)
- viridis, [70](#), [71](#)
- viridis\_magma\_palette, [13](#), [23](#), [57](#), [69](#), [71](#)
- viridis\_palette, [13](#), [23](#), [57](#), [70](#), [70](#)
- vlookup, [14](#), [31](#), [65](#), [67](#), [71](#)
- which\_format, [9](#), [10](#), [26–29](#), [33](#), [69](#), [72](#)
- write.tree, [20](#), [48](#)