Package ‘yatah’

May 8, 2026

Title Yet Another TAxonomy Handler

Version 1.0.0

Description Provides functions to manage taxonomy when lineages are described with strings and ranks separated with special patterns like ``|*__" or ``;*__".

License GPL-3

URL `https://abichat.github.io/yatah/`,
`https://github.com/abichat/yatah/`

BugReports `https://github.com/abichat/yatah/issues`

Depends R (>= 2.10)

Imports ape, lifecycle, purrr, stats, stringr

Suggests dplyr, knitr, rmarkdown, spelling, testthat (>= 2.1.0)

VignetteBuilder knitr

Encoding UTF-8

Language en-US

LazyData true

RoxygenNote 7.3.1

NeedsCompilation no

Author Antoine Bichat [aut, cre] (ORCID:
<`https://orcid.org/0000-0001-6599-7081`>)

Maintainer Antoine Bichat <`antoine.bichat@proton.me`>

Repository CRAN

Date/Publication 2024-04-13 16:20:02 UTC

Contents

`abundances` *Abundance table for 199 samples.*

Description

A dataset containing the abundances of 1585 lineages among 199 patients.

Usage

```
abundances
```

Format

A data.frame with 1585 rows and 200 variables:

lineages lineage (string)

XXX abundance of each lineage in the sample XXX (double)

Source

Zeller et al., 2014 (doi:10.15252/msb.20145645), Pasolli et al., 2017 (doi:10.1038/nmeth.4468).

Examples

```
dim(abundances)
abundances[1:5, 1:7]
```

`all_ranks` *Ranks handled by* `yatah`

Description

Ranks handled by `yatah`

Usage

```
all_ranks
```

Format

An object of class `character` of length 8.

Examples

```
all_ranks
```

`get_all_clades` *Extract all clades present in the lineages*

Description

Extract all clades present in the lineages

Usage

```
get_all_clades(lineage, simplify = TRUE)
```

Arguments

`lineage`	string. Vector of lineages.
`simplify`	logical. Should the output be a vector or a dataframe?

Details

If a clade correspond to different ranks (e.g. Actinobacteria is both a phylum and a clade), it will be displayed only one time when `simplify` is set to `TRUE`. It is also the case for different clades with same name and same rank when `simplify` is set to `FALSE`.

Value

The clades present in the lineage. Vector of ordered strings or data.frame.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae"
lineage2 <- "k__Bacteria|p__Firmicutes|c__Clostridia"
lineage3 <- "k__Bacteria|p__Actinobacteria|c__Actinobacteria"
get_all_clades(c(lineage1, lineage2, lineage3))
get_all_clades(c(lineage1, lineage2, lineage3), simplify = FALSE)
```

`get_clade` *Extract the clade of a desired rank in a lineage*

Description

Extract the clade of a desired rank in a lineage

Usage

```
get_clade(lineage, rank = yatah::all_ranks, same = TRUE)
```

Arguments

`lineage`	string. Vector of lineages.
`rank`	The desired rank of the clade.
`same`	logical. Does the lineage have the same depth? Default to `TRUE`.

Value

A string.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae"
lineage2 <- "k__Bacteria|p__Firmicutes|c__Clostridia"
get_clade(c(lineage1, lineage2), "phylum")
```

`get_last_clade` *Extract the last clade of a lineage*

Description

Extract the last clade of a lineage

Usage

```
get_last_clade(lineage, same = TRUE)
```

Arguments

`lineage`	string. Vector of lineages.
`same`	logical. Does the lineage have the same depth? Default to `TRUE`.

Value

A string. The last clades of the given lineages.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae"
lineage2 <- "k__Bacteria|p__Firmicutes|c__Clostridia"
get_last_clade(c(lineage1, lineage2))
```

`get_last_rank` *Extract the last rank of a lineage*

Description

Extract the last rank of a lineage

Usage

```
get_last_rank(lineage, same = TRUE)
```

Arguments

`lineage`	string. Vector of lineages.
`same`	logical. Does the lineage have the same depth? Default to `TRUE`.

Value

A string. The last rank of the given lineages.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae"
lineage2 <- "k__Bacteria|p__Firmicutes|c__Clostridia"
get_last_rank(c(lineage1, lineage2))
```

`is_clade` *Test if a lineage belongs to a clade*

Description

Test if a lineage belongs to a clade

Usage

```
is_clade(lineage, clade, rank = c(".", yatah::all_ranks))
```

Arguments

`lineage`	string. Vector of lineages.
`clade`	string.
`rank`	string. One of `c("kingdom", "phylum", "class","order", "family", "genus", "species", "strain")` with partial matching.

Details

If `rank` is set to `.`, clade is looked for among all ranks.

Value

logical.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae"
lineage2 <- "k__Bacteria|p__Firmicutes|c__Clostridia"
is_clade(c(lineage1, lineage2), clade = "Verrucomicrobia", rank = "phylum")
is_clade(c(lineage1, lineage2), clade = "Clostridia")
```

`is_lineage` *Test if a string is a lineage*

Description

Test if a string is a lineage

Usage

```
is_lineage(string)
```

Arguments

`string`	string to be tested as lineage.

Details

Alphanumeric character, hyphen, dots, square brackets and non-consecutive underscores are allowed in clades names.

Value

A logical.

Examples

```
is_lineage("k__Bacteria|p__Firmicutes|c__Clostridia|o__Clostridiales")
```

`is_rank` *Test if a lineage goes down to a specified rank*

Description

Test if a lineage goes down to a specified rank

Usage

```
is_rank(lineage, rank = yatah::all_ranks)

is_at_least_rank(lineage, rank = yatah::all_ranks)
```

Arguments

`lineage`	string. Vector of lineages.
`rank`	string. One of `c("kingdom", "phylum", "class","order", "family", "genus", "species", "strain")` with partial matching.

Value

logical.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae"
lineage2 <- "k__Bacteria|p__Firmicutes|c__Clostridia"
is_rank(c(lineage1, lineage2), "class")
is_rank(c(lineage1, lineage2), "order")
is_at_least_rank(c(lineage1, lineage2), "phylum")
is_at_least_rank(c(lineage1, lineage2), "order")
```

`taxtable` *Taxonomic table*

Description

Compute taxonomic table from lineages.

Usage

```
taxtable(lineage)
```

Arguments

`lineage`	string. Vector of lineages.

Details

Duplicated lineages are removed.

Value

A data.frame with columns corresponding to different ranks.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae"
lineage2 <- "k__Bacteria|p__Firmicutes|c__Clostridia"
lineage3 <- "k__Bacteria|p__Firmicutes|c__Bacilli"
taxtable(c(lineage1, lineage2, lineage3))
```

`taxtree` *Taxonomic tree*

Description

Compute taxonomic tree from taxonomic table.

Usage

```
taxtree(table, collapse = TRUE, lineage_length = 1, root = "")
```

Arguments

`table`	dataframe.
`collapse`	logical. Should node with one child be vanished? Default to TRUE.
`lineage_length`	double. Lineage length from the root to the leaves. Default to 1.
`root`	character. Name of the root if there is no natural root.

Value

A phylo object.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae"
lineage2 <- "k__Bacteria|p__Firmicutes|c__Clostridia"
lineage3 <- "k__Bacteria|p__Firmicutes|c__Bacilli"
table <- taxtable(c(lineage1, lineage2, lineage3))
taxtree(table)
```

`trim_common` *Trim lineages until the shallowest common rank.*

Description

Trim lineages until the shallowest common rank.

Usage

```
trim_common(lineage, remove_void = TRUE, only_tail = TRUE)
```

Arguments

`lineage`	string. Vector of lineages.
`remove_void`	Should void ranks be removed? Default to `TRUE`.
`only_tail`	Logical to be passed to `trim_void()`. Used only if `remove_void` is set to `TRUE`.

Value

The trimmed lineages, with same depth.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae"
lineage2 <- "k__Bacteria|p__Firmicutes"
lineage3 <- "k__Bacteria|p__|c__Clostridia"
trim_common(c(lineage1, lineage2, lineage3), remove_void = FALSE)
trim_common(c(lineage1, lineage2, lineage3), only_tail = FALSE)
```

`trim_rank` *Trim lineages until a specified rank*

Description

Trim lineages until a specified rank

Usage

```
trim_rank(lineage, rank = yatah::all_ranks, same = TRUE)
```

Arguments

`lineage`	string. Vector of lineages.
`rank`	string. One of `c("kingdom", "phylum", "class","order", "family", "genus", "species", "strain")` with partial matching.
`same`	logical. Does the lineage have the same depth? Default to `TRUE`.

Details

Returns `NA` if a lineage is not as deep as the specified rank.

Value

The trimmed lineages. Depth could be different among them.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae"
lineage2 <- "k__Bacteria|p__Firmicutes|c__Clostridia"
trim_rank(c(lineage1, lineage2), rank = "phylum")
trim_rank(c(lineage1, lineage2), rank = "genus")
```

`trim_void` *Trim void ranks in lineages*

Description

Trim void ranks in lineages

Usage

```
trim_void(lineage, same = TRUE, only_tail = FALSE)
```

Arguments

`lineage`	string. Vector of lineages.
`same`	logical. Does the lineage have the same depth? Default to `TRUE`.
`only_tail`	Logical. If `FALSE` (default), void ranks amid lineages and subranks are removed. If `TRUE`, only final void ranks are removed.

Details

If there is a void rank amid a lineage, deeper ranks will be removed. See the example with `lineage3`.

Value

The trimmed lineages. Depth could be different among them.

Examples

```
lineage1 <- "k__Bacteria|p__Verrucomicrobia|c__Verrucomicrobiae|o__|f__"
lineage2 <- "k__Bacteria|p__Firmicutes|c__"
lineage3 <- "k__Bacteria|p__|c__Verrucomicrobiae|o__|f__"
trim_void(c(lineage1, lineage2, lineage3), same = FALSE)
trim_void(c(lineage1, lineage2, lineage3), same = FALSE, only_tail = TRUE)
```

Index