Package 'samovaR'

June 4, 2025

Type Package

Title R package for generating model metagenomes with specified properties

Version 0.9.0

Description There is a fundamental problem in modern metagenomics: there are huge differences between methodological approaches that strongly influence the results, while remaining outside the attention of researchers. We propose an approach that utilizes de novo generation of the artificial metagenomes - SamovaR.

URL https://github.com/ctlab/samovar/

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Depends R (>= 3.5.0)

Imports tidyverse,
plotly,
httr,
jsonlite,
xml2,
tsne,
Matrix,
scclust,
distances,
htmlwidgets,
progress,
methods,
dplyr,
ggplot2,
stringr,
tibble,
tidyr,
ggnewscale,
yaml

Suggests shiny,
knitr,
rmarkdown,

testthat (>= 3.0.0),
withr,
phyloseq

VignetteBuilder knitr

Config/testthat/edition 3

Contents

`annotation2samovar` *Process annotation data.frame to SamovaR*

Description

Process annotation data.frame to SamovaR

Usage

```
annotation2samovar(data)
```

Arguments

data	Processed abundance table. Row names: sequence IDs, Column names: • annotators: (starting with taxID_); • true: for true annotation • length: length of sequence • sample

Value

list of samovar data objects

Examples

```
library(samovaR)

data <- read_annotation_dir("data/test_annotations/")
gglist <- viz_annotation(data)

samovar_list <- annotation2samovar(data)
```

---

build_samovar *Build samovar object*

Description

Samovar network is a 2D-oriented graph with metadata and abundances of species per sample. Oriented graph could be used for network prediction, or for better generation some network could be used as initial (to be implemented) For better understanding of building database and using it in generation, visit github source

Usage

```
build_samovar(
  samovar_data,
  dist_function = function(x) dist(x),
  network = F,
  k_means = F,
  min_cluster_size = F,
  plot_log = T
)
```

Arguments

samovar_data	samovar data after preprocessing stages
network	FALSE or graph that can be used for generation. To be implemented
plot_log	Logical or path for log plots output
distance_function	function used for measuring distances between species based on samples
min_min_cluster_size	FALSE or minimum number of species per cluster

`max_min_cluster_size`
: FALSE or minimum number of species per cluster

`concotion_pour` *Build samovar object*

Description

Samovar network is a 2D-oriented graph with metadata and abundances of species per sample. Oriented graph could be used for network prediction, or for better generation some network could be used as initial (to be implemented) For better understanding of building database and using it in generation, visit github source

Usage

```
concotion_pour(
  samovar_data,
  inner_method = "glm",
  inter_method = "glm",
  inner_model = "gaussian",
  inter_model = "gaussian",
  probability_calculation = "oriented",
  cluster_connection = "mean",
  ...
)
```

Arguments

`samovar_data`	samovar data after preprocessing stages
`inner_method`	Character, glm, other to be implemented (bootstrap, bsPCA)
`inter_method`	Character, glm, other to be implemented (bootstrap, bsPCA)
`inner_model`	Character, model processed by glm(). For glm mode only. quasipoisson by default
`inter_model`	Character, model processed by glm(). For glm mode only. quasipoisson by default
`cluster_connection`	Character (mean, median), or function. The way of cluster connection. If function, way of summarize all samples of species cluster
`...`	Additional arguments, passed
`network`	FALSE or graph that can be used for generation. To be implemented
`cooccurrence`	Character, co-occurrence calculation. If "simple", calculated as: P(A|B) = sum(A&B)/sum(A|B). If "oriented", calculated as P(A|B) = P(A&B|B) If "compositional", calculated as P(A|B) = P(A&B|B), and than sampled one of represented conditions of occurence

Examples

```
library(samovaR)
library(tidyverse)

# download data
teatree <- GMrepo_type2data(number_to_process = 2000)

# filter
tealeaves <- teatree %>%
  teatree_trim(treshhold_species = 3, treshhold_samples = 3, treshhold_amount = 10^(-3))

# normalizing
## if you build teatree by your own, rescaling stage when building via teatree$rescale()
## or assigning teatree$min_value and teatree$max_value is required
## good approximation to normal distribution is required for glm generating methods
teabag <- tealeaves %>%
  tealeaves_pack(normalization_function = function(x) log10(x+1))

# clustering
concotion <- teabag %>%
  teabag_brew(min_cluster_size = 4, max_cluster_size = 6)
# remember: if you want to refilter, it is better to re-do welding stage to avoid crashes in future!

# building samovar
samovar <- concotion %>%
  concotion_pour()
```

GMrepo_run *GMrepo run data class*

Description

GMrepo run data class

Slots

`metadata` metadata DataFrame

`run` character

GMrepo_run2data *Get data from GMrepo_run object*

Description

Get data from GMrepo_run object

Usage

```
GMrepo_run2data(
  run,
  number_to_out = F,
  at_level = "species",
  QC_filter = "QCStatus"
)
```

Arguments

number_to_out	False by default, maximum number of obtained data
at_level	"species" by default. level to obtain classification from GMrepo
QC_filter	QCStatus by default. Perform auto QC filtering based on metadata column, or False for no checking.
runs	GMrepo_run object got by GMrepo_type2run or created by user with `new('GMrepo_run', metadata = data.frame(), run = run_list)`

Examples

```
library(samovaR)
library(tidyverse)

# get data from GMrepo
data_GMrepo <- GMrepo_type2data(mesh_ids = "D006262", number_to_process = 1000)

# equal to:
run_GMrepo <- GMrepo_type2run(mesh_ids = "D006262", number_to_process = 1000)
data_GMrepo <- GMrepo_run2data(run_GMrepo)

# filter runs before obtaining data (OOP updating data!)
run_GMrepo$filter("checking", 1)

# view
data_GMrepo

# access to metadata
data_GMrepo$run

# access to data
data_GMrepo$data

# access to runs
data_GMrepo$run

# access to taxa
data_GMrepo$species
```

GMrepo_type2data *Get data from GMrepo*

Description

Wrapper around GMrepo_type2run and GMrepo_run2data functions

Usage

```
GMrepo_type2data(
  mesh_ids = c("D006262"),
  number_to_process = F,
  number_to_out = F,
  at_level = "species",
  QC_filter = "QCStatus"
)
```

Arguments

mesh_ids	Character. All types of meshID to use. List of relations between meshID and phenotype could be obtained using GMrepo_meshID(). Health meshID by default
number_to_process	False by default, or maximum number of runs per meshID
number_to_out	False by default, maximum number of obtained data
at_level	"species" by default. level to obtain classification from GMrepo
QC_filter	QCStatus by default. Perform auto QC filtering based on metadata column, or False for no checking.

Examples

```
library(samovaR)
library(tidyverse)

# get data from GMrepo
data_GMrepo <- GMrepo_type2data(mesh_ids = "D006262", number_to_process = 1000)

# equal to:
run_GMrepo <- GMrepo_type2run(mesh_ids = "D006262", number_to_process = 1000)
data_GMrepo <- GMrepo_run2data(run_GMrepo)

# filter runs before obtaining data (OOP updating data!)
run_GMrepo$filter("checking", 1)

# view
data_GMrepo

# access to metadata
data_GMrepo$run

# access to data
data_GMrepo$data

# access to runs
data_GMrepo$run

# access to taxa
data_GMrepo$species
```

GMrepo_type2run *Get runs from GMrepo by meshID*

Description

Get runs from GMrepo by meshID

Usage

```
GMrepo_type2run(mesh_ids = c("D006262"), number_to_process = F)
```

Arguments

mesh_ids	Character. All types of meshID to use. List of relations between meshID and phenotype could be obtained using GMrepo_meshID()
number_to_process	False by default, or maximum number of runs per meshID

Examples

```
library(samovaR)
library(tidyverse)

# get data from GMrepo
data_GMrepo <- GMrepo_type2data(mesh_ids = "D006262", number_to_process = 1000)

# equal to:
run_GMrepo <- GMrepo_type2run(mesh_ids = "D006262", number_to_process = 1000)
data_GMrepo <- GMrepo_run2data(run_GMrepo)

# filter runs before obtaining data (OOP updating data!)
run_GMrepo$filter("checking", 1)

# view
data_GMrepo

# access to metadata
data_GMrepo$run

# access to data
data_GMrepo$data

# access to runs
data_GMrepo$run

# access to taxa
data_GMrepo$species
```

log_plot *Print a log plot*

Description

Print a log plot

Usage

```
log_plot(plot_log, postfix, gg, mode = "ggplot", write = F)
```

minmaxscale *Misc functions*

Description

Misc functions

Usage

```
minmaxscale(x)
```

phyloseq2samovar *Convert phyloseq object to samovar object*

Description

Convert phyloseq object to samovar object

Usage

```
phyloseq2samovar(phyloseq_data)
```

Arguments

phyloseq_data	A phyloseq object

Value

A samovar object

progress_function *Progress bar*

Description

Progress bar

Usage

```
progress_function(iters)
```

read_annotation_dir *Read annotations produced with samovar pipeline from the directory*

Description

Read annotations produced with samovar pipeline from the directory

Usage

```
read_annotation_dir(data_dir, sample_name_position = 0, ...)
```

Arguments

data_dir	Path to abundance table. Row names: sequence IDs, column names: annotators; true for true annotation
sample_name_position	Position to split the file basename by "." to extract sample names. 0 by default
...	Parameters processed by read.csv()

Value

samovar object

Examples

```
library(samovaR)

data <- read_annotation_dir("data/test_annotations/")
gglist <- viz_annotation(data)

samovar_list <- annotation2samovar(data)
```

`read_samovar` *Build samovar data object from file or environment*

Description

Build samovar data object from file or environment

Usage

```
read_samovar(data, metadata = F, ...)
```

Arguments

`data`	path to abundance matrix Row names: organisms/OTUs/ASVs Column names: samples
`metadata`	Data.frame or path to metadata file
`...`	Parameters processed by read.csv()

Value

samovar object

Examples

```
library(samovaR)

## build samovar directly from the table
data <- matrix(
  runif(25),
  dimnames = list(
    rownames = paste0("sp_", 1:5),
    colnames = paste0("sample_", 1:5)
  ),
  nrow = 5)

samovar_data <- table2samovar(data)
print(samovar_data)

# or read abundance table
tf <- tempfile()
write.csv(data, tf)

samovar_data <- read_samovar(tf)
print(samovar_data)
```

samovar2phyloseq *Convert samovar object to phyloseq object*

Description

Convert samovar object to phyloseq object

Usage

```
samovar2phyloseq(samovar_data)
```

Arguments

samovar_data	A samovar object

Value

A phyloseq object

samovar_base *samovar base class*

Description

samovar base class

Slots

samovar_base	samovar_data object
method	method to obtain samovar_base
inner_cluster_graph_method	list of graphs in matrix form of inner cluster connections
inter_cluster_graph_method	list of graphs in matrix form of inter cluster connections
inner_cluster_graph_prob	list of co-occurrence probabilities in matrix form of inner cluster members
inter_cluster_graph_prob	list of co-occurrence probabilities in matrix form between clusters
properties	concotion_pour() properties

samovar_boil *Generate artificial data*

Description

Use pre-built samovar_data with its parameters

Usage

```
samovar_boil(
  samovar_base,
  N = 1,
  init_sp = F,
  init_ab = F,
  avoid_zero_generations = T,
  seed = 42
)
```

Arguments

samovar_base	samovar data after preprocessing and building stages
N	number of artificial samples to generate
init_sp	species vector for initializing data generation, or FALSE for usage most common taxa, auto for choosing random taxa
init_ab	species amount vector (values from 0 to 1) for initializing data generation, or FALSE for mean initial taxa assignment, or auto for usage from known edf for each species from init_sp
avoid_zero_generations	logical, avoid zero-based generations or not. FALSE might results in under-distributed communities, while TRUE in over-represented with species from different clusters possibly come from different samples groups
seed	initial seed for the seeds generation

Examples

```
library(samovaR)
library(tidyverse)

# download and prepare data
samovar_raw <- GMrepo_type2data(number_to_process = 2000)

samovar_data <- samovar_raw %>%
  samovar_preprocess()

# Similar to:
## samovar_raw %>%
##   teatree_trim() %>%
##   tealeaves_pack() %>%
##   teabag_brew() %>%
##   concotion_pour()
```

```
# generate
new_data <- samovar_data %>%
  samovar_boil(N = 100)
```

`samovar_data` *samovar data class*

Description

samovar data class

Slots

`description` metadata DataFrame

`data` DataFrame with species abundances. No NA pass

`run` character, runs

`species` character, runs

`normalization_function` normalization function for samples

`reverse_normalization_function` reverse normalization function

`min_value` minimal value after scaling

`max_value` maximal value after scaling

`cluster` character vector, enumerated clusters for each species

`cluster_size` named numeric, cluster sizes per cluster

`samovar_preprocess` *Preprocess SAMOVAR data*

Description

Wrapper for SAMOVAR preprocess commands

Usage

```
samovar_preprocess(
  samovar_data,
  metadata_filter = F,
  treshhold_amount = 10^(-5),
  treshhold_samples = 1,
  treshhold_species = 1,
  drop_species = F,
  drop_unclassified = T,
  normalization_function = function(x) log10(x + 1),
  plot_log = T,
  dist_function = function(x) dist(x),
  network = F,
  min_cluster_size = 2,
  max_cluster_size = 100,
```

```
  inner_method = "glm",
  inter_method = "glm",
  inner_model = "gaussian",
  inter_model = "gaussian",
  probability_calculation = "oriented",
  cluster_connection = "mean",
  ...
)
```

Arguments

samovar_data	samovar data object
metadata_filter	False, character or data.frame with 2 columns: first contain metadata names for filtering, and second values per column
treshhold_amount	Minimum value to conclude as not the noise.
treshhold_samples	Minimum number of representing samples to keep species.
treshhold_species	Minimum number of representing species to keep samples.
drop_unclassified	Drop unknown and unclassified ranks. True by default
normalization_function	Function using for rescaling
plot_log	Logical or path for log plots output
network	FALSE or graph that can be used for generation. To be implemented
min_cluster_size	FALSE or minimum number of species per cluster
max_cluster_size	FALSE or minimum number of species per cluster
inner_method	Character, glm, other to be implemented (bootstrap, bsPCA)
inter_method	Character, glm, other to be implemented (bootstrap, bsPCA)
inner_model	Character, model processed by glm(). For glm mode only. quasipoisson by default
inter_model	Character, model processed by glm(). For glm mode only. quasipoisson by default
cluster_connection	Character (mean, median), or function. The way of cluster connection. If function, way of summarize all samples of species cluster
...	Additional arguments, passed

Value

Build SAMOVAR object

Examples

```
library(samovaR)
library(tidyverse)

# download and prepare data
samovar_raw <- GMrepo_type2data(number_to_process = 2000)

samovar_data <- samovar_raw %>%
  samovar_preprocess()

# Similar to:
## samovar_raw %>%
##   teatree_trim() %>%
##   tealeaves_pack() %>%
##   teabag_brew() %>%
##   concotion_pour()

# generate
new_data <- samovar_data %>%
  samovar_boil(N = 100)
```

`samovar_run` *Samovar run data class*

Description

Samovar run data class

Slots

`metadata` metadata DataFrame

`data` data

`run` character, samle IDs

Methods

`export(Class)` Returns the result of coercing the object to Class. No effect on the object itself.

`table2samovar` *Build samovar object from the abundance matrix*

Description

Build samovar object from the abundance matrix

Usage

```
table2samovar(data, metadata = F, min_sp = 0, min_samp = 0)
```

Arguments

data	abundance matrix Row names: organisms/OTUs/ASVs Column names: samples
metadata	metadata data.frame in format: ADD, or FALSE
min_sp	data_samovar$rebuild() option, minimal number of species to filter. 0 by default
min_samp	data_samovar$rebuild() option, minimal number of species to filter. 0 by default

Value

samovar object

Examples

```
library(samovaR)

## build samovar directly from the table
data <- matrix(
  runif(25),
  dimnames = list(
    rownames = paste0("sp_", 1:5),
    colnames = paste0("sample_", 1:5)
  ),
  nrow = 5)

samovar_data <- table2samovar(data)
print(samovar_data)

# or read abundance table
tf <- tempfile()
write.csv(data, tf)

samovar_data <- read_samovar(tf)
print(samovar_data)
```

---

teabag_brew *Build samovar object*

Description

Samovar network is a 2D-oriented graph with metadata and abundances of species per sample. Oriented graph could be used for network prediction, or for better generation some network could be used as initial (to be implemented) For better understanding of building database and using it in generation, visit github source

Usage

```
teabag_brew(
  samovar_data,
  dist_function = function(x) dist(x),
  network = F,
  min_cluster_size = 10,
  max_cluster_size = 100,
  plot_log = F,
```

```
  ...
)
```

Arguments

network	FALSE or graph that can be used for generation. To be implemented
min_cluster_size	FALSE or minimum number of species per cluster
max_cluster_size	FALSE or minimum number of species per cluster
plot_log	Logical or path for log plots output
...	Additional arguments, passed
data	samovar data after preprocessing stages
distance_function	function used for measuring distances between species based on samples

---

tealeaves_pack *Scale species abundances*

Description

Scale species abundances

Usage

```
tealeaves_pack(
  samovar_data,
  normalization_function = function(x) log10(x + 1),
  plot_log = T,
  ...
)
```

Arguments

samovar_data	Samovar data object to rescale
normalization_function	Function using for rescaling
plot_log	Logical or path for log plots output
...	Additional arguments, passed

Examples

```
library(samovaR)
library(tidyverse)

# download data
teatree <- GMrepo_type2data(number_to_process = 2000)

# filter
tealeaves <- teatree %>%
```

```
  teatree_trim(treshhold_species = 3, treshhold_samples = 3, treshhold_amount = 10^(-3))

# normalizing
## if you build teatree by your own, rescaling stage when building via teatree$rescale()
## or assigning teatree$min_value and teatree$max_value is required
## good approximation to normal distribution is required for glm generating methods
teabag <- tealeaves %>%
  tealeaves_pack(normalization_function = function(x) log10(x+1))

# clustering
concotion <- teabag %>%
  teabag_brew(min_cluster_size = 4, max_cluster_size = 6)
# remember: if you want to refilter, it is better to re-do welding stage to avoid crashes in future!

# building samovar
samovar <- concotion %>%
  concotion_pour()
```

`teatree_trim` *Filter species and samples from samovar_data object*

Description

Filter species and samples from samovar_data object

Usage

```
teatree_trim(
  samovar_data,
  metadata_filter = F,
  treshhold_amount = 10^(-5),
  treshhold_samples = 1,
  treshhold_species = 1,
  drop_species = F,
  drop_unclassified = T,
  ...
)
```

Arguments

`samovar_data`	Samovar data object to filter
`metadata_filter`	False, character or data.frame with 2 columns: first contain metadata names for filtering, and second values per column
`treshhold_amount`	Minimum value to conclude as not the noise.
`treshhold_samples`	Minimum number of representing samples to keep species.
`treshhold_species`	Minimum number of representing species to keep samples.
`drop_unclassified`	Drop unknown and unclassified ranks. True by default
`...`	Additional arguments, passed

Examples

```
library(samovaR)
library(tidyverse)

# download data
teatree <- GMrepo_type2data(number_to_process = 2000)

# filter
tealeaves <- teatree %>%
  teatree_trim(treshhold_species = 3, treshhold_samples = 3, treshhold_amount = 10^(-3))

# normalizing
## if you build teatree by your own, rescaling stage when building via teatree$rescale()
## or assigning teatree$min_value and teatree$max_value is required
## good approximation to normal distribution is required for glm generating methods
teabag <- tealeaves %>%
  tealeaves_pack(normalization_function = function(x) log10(x+1))

# clustering
concotion <- teabag %>%
  teabag_brew(min_cluster_size = 4, max_cluster_size = 6)
# remember: if you want to refilter, it is better to re-do welding stage to avoid crashes in future!

# building samovar
samovar <- concotion %>%
  concotion_pour()
```

---

unpack_config *Unpack SAMOVAR config*

Description

Unpack SAMOVAR config to a list of parameters

Usage

```
unpack_config(config_samovar)
```

Arguments

config_samovar	SAMOVAR config

Value

arguments for samovar_preprocess and samovar_boil

Examples

```
library(samovar)
library(yaml)

# Example config
tf <- tempfile()
```

```
write_yaml(
  list(
    treshhold_amount = 10^(-5),
    plot_log = F,
    min_cluster_size = 5,
    N = 5,
    N_reads = 100
  ),
  tf
)

config_samovar <- unpack_config(tf)

## build samovar directly from the table
data <- matrix(
  rlnorm(625),
  dimnames = list(
    rownames = paste0("sp_", 1:25),
    colnames = paste0("sample_", 1:25)
  ),
  nrow = 25)

samovar_data <- table2samovar(data)

# Run with config
config_samovar$samovar_data <- samovar_data
samovar <- do.call(samovar_preprocess, config_samovar)
samovar_new <- samovar_boil(samovar, N = config_samovar$N)

new_data <- samovar_new$data * config_samovar$N_reads

heatmap(as.matrix(new_data))
```

`viz_annotation` *Visualize annotation results*

Description

Visualize annotation results

Usage

```
viz_annotation(
  data,
  type = c("f1", "R2", "cv", "conf"),
  show_top = 10,
  output_dir = NULL,
  plot = T
)
```

Arguments

`data`	Processed abundance table. Row names: sequence IDs, Column names: • annotators: (starting with `taxID_`);

- `true`: for true annotation
- `length`: length of sequence
- sample

`type` character vector.

- if present column true_annotation: could be one of
  - "f1",
  - "R2",
  - "confidence",
  - "cross-validation",
  - or their combination (e.g. c("f1", "R2", "cv", "conf"))

`show_top` integer. Number of top annotations to show.

`output_dir` character. Directory to save the plots. If NULL, plots are not saved.

`plot` logical. If TRUE, plots are printed.

Value

list of ggplot objects

Examples

```
library(samovaR)

data <- read_annotation_dir("data/test_annotations/")
gglist <- viz_annotation(data)

samovar_list <- annotation2samovar(data)
```

---

viz_composition *Visualize composition*

Description

Visualize composition

Usage

```
viz_composition(
  data,
  reord_samples = "fpc",
  reord_species = "amount",
  type = "column",
  top = 15,
  interactive = F,
  ggplot_add = F,
  bottom_legend = F
)
```

Arguments

data	data.frame with dimensions of species * samples, or samovar objects: samovar_data, samovar_base, samovar_run or GMrepo_run (with data)
reord_samples	character, fpc, fpc_scaled, hcl, amount, tsne, or none reorder of samples on plot
reord_species	character, same for reord_samples
type	character, column or tile for composition visualize, or donut (to implement) for mean composition visualization
top	integer, number of top-represented taxa to show, or FALSE to show all
interactive	logical. ggplot or plotly object to return
ggplot_add	functions to add to ggplot object, or FALSE.
bottom_legend	vector length of samples to show on plot as a color legend, or FALSE

Examples

```
library(samovaR)
library(tidyverse)

# Download data
teatree <- GMrepo_type2data(number_to_process = 1000)

#Composition
viz_composition(teatree)
```

Index