Package 'samovaR'

May 23, 2025

Type Package

Title R package for generating model metagenomes with specified properties

Version 0.6

Author@R
c(person(``Smutin'', ``Daniil'', email = ``dvsmutin@gmail.com'', role = c(``aut'', ``cre''),
comment = c(ORCID = ``0009-0009-1460-4108'')),
person(``Ivanov'', ``Artem'', role = c(``aut'')),
person(``ctlab'', role = c(``fnd'')))

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

Imports tidyverse,
plotly,
httr,
jsonlite,
xml2,
tsne,
cluster,
Matrix,
shiny,
methods,
scclust

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Suggests ggnewscale,
knitr,
rmarkdown,
testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Contents

`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",
  minimal_cluster = 2,
  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
`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

```
# 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$rescale() or assigning teatree$mi
## 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

```
# 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

```
# 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

```
# 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)
```

`progress_function` *Progress bar*

Description

Progress bar

Usage

```
progress_function(iters)
```

`read.abundance` *Build samovar data object from file or environment*

Description

Build samovar data object from file or environment

Usage

```
read.abundance(data, metadata, ...)
```

Arguments

`data`	Data.frame or path to abundance file. Row names is using as species list, column names as sample list. Unique names required
`metadata`	Data.frame or path to metadata file
`...`	Parameters processed by read.table()

`read.samovar` *Read samovar data frame from file*

Description

Prints 'Hello, world!'.

Usage

```
hello()
```

Examples

```
hello()
```

`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

```
# download and prepare data
samovar <- GMrepo_type2data(number_to_process = 2000) %>%
 teatree_trim(treshhold_species = 3, treshhold_samples = 3, treshhold_amount = 10^(-3)) %>%
  tealeaves_pack(normalization_function = function(x) log10(x+1)) %>%
  teabag_brew(min_cluster_size = 4, max_cluster_size = 6) %>%
  concotion_pour()

# generate
new_data <- samovar %>%
  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_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.

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
`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

Examples

```
# 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$rescale() or assigning teatree$mi
## 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

Examples

```
# 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$rescale() or assigning teatree$mi
## 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()
```

`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

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

#Composition
viz_composition(teatree)
```

Index