Package 'mia'

September 14, 2026

Type Package

Version 1.20.0

Title Microbiome analysis

Description mia implements tools for microbiome analysis based on the SummarizedExperiment, SingleCellExperiment and TreeSummarizedExperiment infrastructure. Data wrangling and analysis in the context of taxonomic data is the main scope. Additional functions for common task are implemented such as community indices calculation and summarization.

biocViews Microbiome, Software, DataImport

License Artistic-2.0 | file LICENSE

Encoding UTF-8

LazyData false

Depends R (>= 4.0), MultiAssayExperiment, SingleCellExperiment, SummarizedExperiment, TreeSummarizedExperiment (>= 1.99.3)

Imports ape, BiocGenerics, BiocParallel, Biostrings, bluster, DECIPHER, decontam, DelayedArray, DelayedMatrixStats, DirichletMultinomial, dplyr, IRanges, MASS, MatrixGenerics, methods, ecodive, rlang, S4Vectors, scater, stats, stringr, tibble, tidyr, utils, vegan, Rcpp

Suggests ade4, BiocStyle, biomformat, dada2, knitr, mediation, miaTime, miaViz, microbiomeDataSets, NMF, patchwork, philr, phyloseq, reldist, rhdf5, rmarkdown, scuttle, testthat, topicdoc, topicmodels, yaml

LinkingTo Rcpp

URL https://microbiome.github.io/mia/, https://github.com/microbiome/mia

BugReports https://github.com/microbiome/mia/issues

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

VignetteBuilder knitr

git_url https://git.bioconductor.org/packages/mia

git_branch RELEASE_3_23

git_last_commit 5c52652

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-09-13

Author Tuomas Borman [aut, cre] (ORCID:
<https://orcid.org/0000-0002-8563-8884>),
Felix G.M. Ernst [aut] (ORCID: <https://orcid.org/0000-0001-5064-0928>),
Sudarshan A. Shetty [aut] (ORCID:
<https://orcid.org/0000-0001-7280-9915>),
Leo Lahti [aut] (ORCID: <https://orcid.org/0000-0001-5537-637X>),
Yang Cao [ctb],
Nathan D. Olson [ctb],
Levi Waldron [ctb],
Marcel Ramos [ctb],
Héctor Corrada Bravo [ctb],
Jayaram Kancherla [ctb],
Domenick Braccia [ctb],
Basil Courbayre [ctb],
Geraldson Muluh [ctb],
Giulio Benedetti [ctb],
Moritz Emanuel Beber [ctb] (ORCID:
<https://orcid.org/0000-0003-2406-1978>),
Chouaib Benchraka [ctb],
Akewak Jeba [ctb] (ORCID: <https://orcid.org/0009-0007-1347-7552>),
Himmi Lindgren [ctb],
Noah De Gunst [ctb],
Théotime Pralas [ctb],
Shadman Ishraq [ctb],
Eineje Ameh [ctb],
Artur Sannikov [ctb] (ORCID: <https://orcid.org/0000-0001-7765-123X>),
Hervé Pagès [ctb],
Rajesh Shigdel [ctb],
Katariina Pärnänen [ctb],
Pande Erawijantari [ctb],
Danielle Callan [ctb],
Sam Hillman [ctb],
Jesse Pasanen [ctb],
Eetu Tammi [ctb],
Aituar Bektanov [ctb],
Rasmus Hindström [ctb]

Maintainer Tuomas Borman <tuomas.v.borman@utu.fi>

Contents

mia-package . . . 4
addAlpha . . . 5
addContaminantQC . . . 12
addDissimilarity . . . 14
addDivergence . . . 18
addMediation . . . 20
agglomerateByPrevalence . . . 24
agglomerateByRank . . . 26
calculateDMN . . . 32
convertFromDADA2 . . . 36
convertToBIOM . . . 37
convertToPhyloseq . . . 39
deprecate . . . 40
dmn_se . . . 48
enterotype . . . 49
esophagus . . . 50
getAbundant . . . 51
getCCA . . . 55
getCluster . . . 59
getCrossAssociation . . . 62
getDominant . . . 68
getDPCoA . . . 70
getHierarchyTree . . . 72
getLDA . . . 73
getMDS . . . 75
getNMDS . . . 76
getNMF . . . 79
getPERMANOVA . . . 81
getPrevalence . . . 83
getReducedDimAttribute . . . 89
getRPCA . . . 90
GlobalPatterns . . . 93
HintikkaXOData . . . 94
ibdmdb . . . 95
importHUMAnN . . . 96
importMetaPhlAn . . . 97
importMothur . . . 98
importQIIME2 . . . 100
importTaxpasta . . . 103
meltSE . . . 104
mergeSEs . . . 106
mia-datasets . . . 109
peerj13075 . . . 110
rarefyAssay . . . 111
splitOn . . . 113
summarizeDominance . . . 115

taxonomyRanks 118
Tengeler2020 122
Tito2024QMP 123
transformAssay 124

Index **129**

`mia-package` `mia` *Package.*

Description

`mia` implements tools for microbiome analysis based on the `SummarizedExperiment`, `SingleCellExperiment` and `TreeSummarizedExperiment` infrastructure. Data wrangling and analysis in the context of taxonomic data is the main scope. Additional functions for common task are implemented such as community indices calculation and summarization.

Author(s)

Maintainer: Tuomas Borman <tuomas.v.borman@utu.fi> (ORCID)

Authors:

- Felix G.M. Ernst (ORCID)
- Sudarshan A. Shetty (ORCID)
- Leo Lahti <leo.lahti@iki.fi> (ORCID)

Other contributors:

- Yang Cao [contributor]
- Nathan D. Olson [contributor]
- Levi Waldron [contributor]
- Marcel Ramos [contributor]
- Héctor Corrada Bravo [contributor]
- Jayaram Kancherla [contributor]
- Domenick Braccia [contributor]
- Basil Courbayre [contributor]
- Geraldson Muluh [contributor]
- Giulio Benedetti [contributor]
- Moritz Emanuel Beber (ORCID) [contributor]
- Chouaib Benchraka [contributor]
- Akewak Jeba (ORCID) [contributor]
- Himmi Lindgren [contributor]
- Noah De Gunst [contributor]

- Théotime Pralas [contributor]
- Shadman Ishraq [contributor]
- Eineje Ameh [contributor]
- Artur Sannikov (ORCID) [contributor]
- Hervé Pagès [contributor]
- Rajesh Shigdel [contributor]
- Katariina Pärnänen [contributor]
- Pande Erawijantari [contributor]
- Danielle Callan [contributor]
- Sam Hillman [contributor]
- Jesse Pasanen [contributor]
- Eetu Tammi [contributor]
- Aituar Bektanov [contributor]
- Rasmus Hindström [contributor]

See Also

TreeSummarizedExperiment

`addAlpha` *Estimate alpha diversity indices*

Description

These functions estimates alpha diversity indices optionally using rarefaction.

Usage

```
addAlpha(x, ...)

getAlpha(x, ...)

## S4 method for signature 'SummarizedExperiment'
addAlpha(x, ...)

## S4 method for signature 'SummarizedExperiment'
getAlpha(
  x,
  assay.type = "counts",
 index = c("dbp_dominance", "faith_diversity", "observed_richness", "shannon_diversity"),
  name = index,
  niter = NULL,
  BPPARAM = SerialParam(),
  ...
)
```

Arguments

x	a SummarizedExperiment object.
...	optional arguments: • sample: Integer scalar. Specifies the rarefaction depth i.e. the number of counts drawn from each sample. (Default: min(colSums2(assay(x, assay.type)))) • tree.name: Character scalar. Specifies which rowTree will be used. (Faith's index). (Default: "phylo") • node.label: Character vector or NULL Specifies the links between rows and node labels of phylogeny tree specified by tree.name. If a certain row is not linked with the tree, missing instance should be noted as NA. When NULL, all the rownames should be found from the tree. (Faith's index). (Default: NULL) • only.tips: (Faith's index). Logical scalar. Specifies whether to remove internal nodes when Faith's index is calculated. When only.tips=TRUE, those rows that are not tips of tree are removed. (Default: FALSE) • threshold: (Coverage and all evenness indices). Numeric scalar. From 0 to 1, determines the threshold for coverage and evenness indices. When evenness indices are calculated values under or equal to this threshold are denoted as zeroes. For coverage index, see details. (Default: 0.5 for coverage, 0 for evenness indices) • quantile: (log modulo skewness index). Numeric scalar. Arithmetic abundance classes are evenly cut up to to this quantile of the data. The assumption is that abundances higher than this are not common, and they are classified in their own group. (Default: 0.5) • nclasses: (log modulo skewness index). Integer scalar. The number of arithmetic abundance classes from zero to the quantile cutoff indicated by quantile. (Default: 50) • ntaxa: (absolute and relative indices). Integer scalar. The n-th position of the dominant taxa to consider. (Default: 1) • aggregate: (absolute, dbp, dmn, and relative indices). Logical scalar. Aggregate the values for top members selected by ntaxa or not. If TRUE, then the sum of relative abundances is returned. Otherwise the relative abundance is returned for the single taxa with the indicated rank (default: aggregate = TRUE). • detection: (observed index). Numeric scalar Selects detection threshold for the abundances (Default: 0)
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
index	Character vector. Specifies the alpha diversity indices to be calculated.
name	Character vector. A name for the column of the colData where results will be stored. (Default: index)
niter	Integer scalar. Specifies the number of rarefaction rounds. Rarefaction is not applied when niter=NULL (see Details section). (Default: NULL)
BPPARAM	A BiocParallelParam object specifying whether the calculation should be parallelized.

Details

Different diversity metrics considers different aspects of microbial community. Cassol et al. (2025) categorized alpha diversity metrics into four categories: richness, dominance, information, and phylogenetic. These categories provide complementary information, and by default, `*Alpha` function return indices from each category: observed richness, Berger-Parker dominance, Shannon index for "information", and Faith phylogenetic index.

Diversity:

Alpha diversity is a joint quantity that combines elements or community richness and evenness. Diversity increases, in general, when species richness or evenness increase.

The following diversity indices are available:

- 'coverage': Number of species needed to cover a given fraction of the ecosystem (50 percent by default). Tune this with the threshold argument.
- 'faith': Faith's phylogenetic alpha diversity index measures how long the taxonomic distance is between taxa that are present in the sample (Faith 1992). Larger values represent higher diversity. The current implementation is based on the Stacked Faith's Phylogenetic Diversity (SFPhD) algorithm (Armstrong et al. 2021), which produces values equivalent to `picante::pd` with the parameter `include.root=TRUE`. Using this index requires a rowTree. If the data includes features that are not in tree's tips but in internal nodes, there are two options. First, you can keep those features, and prune the tree to match features so that each tip can be found from the features. Other option is to remove all features that are not tips. (See `only.tips` parameter)
- 'fisher': Fisher's alpha; as implemented in `vegan::fisher.alpha`. (Fisher et al. 1943)
- 'gini_simpson': Gini-Simpson diversity i.e. $1 - lambda$, where $lambda$ is the Simpson index, calculated as the sum of squared relative abundances. This corresponds to the diversity index 'simpson' in `vegan::diversity`. This is also called Gibbs–Martin, or Blau index in sociology, psychology and management studies. The Gini-Simpson index (1-lambda) should not be confused with Simpson's dominance (lambda), Gini index, or inverse Simpson index (1/lambda).
- 'inverse_simpson': Inverse Simpson diversity: $1/lambda$ where $lambda = sum(p^2)$ and p refers to relative abundances. This corresponds to the diversity index 'invsimpson' in vegan::diversity. Don't confuse this with the closely related Gini-Simpson index
- 'log_modulo_skewness': The rarity index characterizes the concentration of species at low abundance. Here, we use the skewness of the frequency distribution of arithmetic abundance classes (see Magurran & McGill 2011). These are typically right-skewed; to avoid taking log of occasional negative skews, we follow Locey & Lennon (2016) and use the log-modulo transformation that adds a value of one to each measure of skewness to allow logarithmization.
- 'shannon': Shannon diversity (entropy).

Dominance:

A dominance index quantifies the dominance of one or few species in a community. Greater values indicate higher dominance.

Dominance indices are in general negatively correlated with alpha diversity indices (species richness, evenness, diversity, rarity). More dominant communities are less diverse.

The following community dominance indices are available:

- 'absolute': Absolute index equals to the absolute abundance of the most dominant n species of the sample (specify the number with the argument `ntaxa`). Index gives positive integer values.
- 'dbp': Berger-Parker index (See Berger & Parker 1970) calculation is a special case of the 'relative' index. dbp is the relative abundance of the most abundant species of the sample. Index gives values in interval 0 to 1, where bigger value represent greater dominance.

$$dbp = \frac{N_1}{N_{tot}}$$

  where N_1 is the absolute abundance of the most dominant species and N_{tot} is the sum of absolute abundances of all species.
- 'core_abundance': Core abundance index is related to core species. Core species are species that are most abundant in all samples, i.e., in whole data set. Core species are defined as those species that have prevalence over 50\ species must be prevalent in 50\ calculate the core abundance index. Core abundance index is sum of relative abundances of core species in the sample. Index gives values in interval 0 to 1, where bigger value represent greater dominance.

$$core_abundance = \frac{N_{core}}{N_{tot}}$$

  where N_{core} is the sum of absolute abundance of the core species and N_{tot} is the sum of absolute abundances of all species.
- 'gini': Gini index is probably best-known from socio-economic contexts (Gini 1921). In economics, it is used to measure, for example, how unevenly income is distributed among population. Here, Gini index is used similarly, but income is replaced with abundance.
  If there is small group of species that represent large portion of total abundance of microbes, the inequality is large and Gini index closer to 1. If all species has equally large abundances, the equality is perfect and Gini index equals 0. This index should not be confused with Gini-Simpson index, which quantifies diversity.
- 'dmn': McNaughton's index is the sum of relative abundances of the two most abundant species of the sample (McNaughton & Wolf, 1970). Index gives values in the unit interval:

$$dmn = (N_1 + N_2)/N_tot$$

  where N_1 and N_2 are the absolute abundances of the two most dominant species and N_{tot} is the sum of absolute abundances of all species.
- 'relative': Relative index equals to the relative abundance of the most dominant n species of the sample (specify the number with the argument `ntaxa`). This index gives values in interval 0 to 1.

$$relative = N_1/N_tot$$

  where N_1 is the absolute abundance of the most dominant species and N_{tot} is the sum of absolute abundances of all species.
- 'simpson_lambda': Simpson's (dominance) index or Simpson's lambda is the sum of squared relative abundances. This index gives values in the unit interval. This value equals the probability that two randomly chosen individuals belongs to the same species. The higher the probability, the greater the dominance (See e.g. Simpson 1949).

$$lambda = \sum(p^2)$$

where p refers to relative abundances.
There is also a more advanced Simpson dominance index (Simpson 1949). However, this is not provided and the simpler squared sum of relative abundances is used instead as the alternative index is not in the unit interval and it is highly correlated with the simpler variant implemented here.

Evenness:

Evenness is a standard index in community ecology, and it quantifies how evenly the abundances of different species are distributed. The following evenness indices are provided:

By default, four indices are returned, each taking into account different aspects: richness (the number of observed unique features), dominance (Berger-Parker), information (Shannon), and phylogenetics (Faith) (Cassol et al., 2025).

The available evenness indices include the following (all in lowercase):

- 'camargo': Camargo's evenness (Camargo 1992)
- 'simpson_evenness': Simpson's evenness is calculated as inverse Simpson diversity (1/lambda) divided by observed species richness S: (1/lambda)/S.
- 'pielou': Pielou's evenness (Pielou, 1966), also known as Shannon or Shannon-Weaver/Wiener/Weiner evenness; H/ln(S). The Shannon-Weaver is the preferred term; see Spellerberg and Fedor (2003).
- 'evar': Smith and Wilson's Evar index (Smith & Wilson 1996).
- 'bulla': Bulla's index (O) (Bulla 1994).

Desirable statistical evenness metrics avoid strong bias towards very large or very small abundances; are independent of richness; and range within the unit interval with increasing evenness (Smith & Wilson 1996). Evenness metrics that fulfill these criteria include at least camargo, simpson, smith-wilson, and bulla. Also see Magurran & McGill (2011) and Beisel et al. (2003) for further details.

Richness:

The richness is calculated per sample. This is a standard index in community ecology, and it provides an estimate of the number of unique species in the community. This is often not directly observed for the whole community but only for a limited sample from the community. This has led to alternative richness indices that provide different ways to estimate the species richness.

Richness index differs from the concept of species diversity or evenness in that it ignores species abundance, and focuses on the binary presence/absence values that indicate simply whether the species was detected.

The function takes all index names in full lowercase. The user can provide the desired spelling through the argument `name` (see examples).

The following richness indices are provided.

- 'ace': Abundance-based coverage estimator (ACE) is another nonparametric richness index that uses sample coverage, defined based on the sum of the probabilities of the observed species. This method divides the species into abundant (more than 10 reads or observations) and rare groups in a sample and tends to underestimate the real number of species. The ACE index ignores the abundance information for the abundant species, based on the assumption that the abundant species are observed regardless of their exact abundance. We use here the

bias-corrected version (O'Hara 2005, Chiu et al. 2014) implemented in `estimateR`. For an exact formulation, see `estimateR`. Note that this index comes with an additional column with standard error information.

- 'chao1': This is a nonparametric estimator of species richness. It assumes that rare species carry information about the (unknown) number of unobserved species. We use here the bias-corrected version (O'Hara 2005, Chiu et al. 2014) implemented in `estimateR`. This index implicitly assumes that every taxa has equal probability of being observed. Note that it gives a lower bound to species richness. The bias-corrected for an exact formulation, see `estimateR`. This estimator uses only the singleton and doubleton counts, and hence it gives more weight to the low abundance species. Note that this index comes with an additional column with standard error information.
- 'hill': Effective species richness aka Hill index (see e.g. Chao et al. 2016). Currently only the case 1D is implemented. This corresponds to the exponent of Shannon diversity. Intuitively, the effective richness indicates the number of species whose even distribution would lead to the same diversity than the observed community, where the species abundances are unevenly distributed.
- 'observed': The *observed richness* gives the number of species that is detected above a given `detection` threshold in the observed sample (default 0). This is conceptually the simplest richness index. The corresponding index in the **vegan** package is "richness".

Rarefaction:

Rarefaction can be used to control uneven sequencing depths. Although, it is highly debated method. Some think that it is the only option that successfully controls the variation caused by uneven sampling depths. The biggest argument against rarefaction is the fact that it omits data.

Rarefaction works by sampling the counts randomly. This random sampling is done `niter` times. In each sampling iteration, `sample` number of random samples are drawn, and alpha diversity is calculated for this subset. After the iterative process, there are `niter` number of result that are then averaged to get the final result.

Refer to Schloss (2024) for more details on rarefaction.

Value

`getAlpha` returns a `DataFrame`. `addAlpha` returns a `x` with additional `colData` column(s) named `name`.

References

Armstrong G. et al. (2021) Efficient computation of Faith's phylogenetic diversity with applications in characterizing microbiomes. *Genome Res.* 31(11):2131-2137. doi: 10.1101/gr.275777.121

Beisel J-N. et al. (2003) A Comparative Analysis of Diversity Index Sensitivity. *Internal Rev. Hydrobiol.* 88(1):3-15. https://portais.ufg.br/up/202/o/2003-comparative_evennes_index.pdf

Berger WH & Parker FL (1970) Diversity of Planktonic Foraminifera in Deep-Sea Sediments. *Science* 168(3937):1345-1347. doi: 10.1126/science.168.3937.1345

Bulla L. (1994) An index of diversity and its associated diversity measure. *Oikos* 70:167–171

Camargo, JA. (1992) New diversity index for assessing structural alterations in aquatic communities. *Bull. Environ. Contam. Toxicol.* 48:428–434.

Cassol, I. (2025) Key features and guidelines for the application of microbial alpha diversity metrics. *Sci. Rep.* 15:622. doi: 10.1038/s41598-024-77864-y

Chao A. (1984) Non-parametric estimation of the number of classes in a population. *Scand J Stat.* 11:265–270.

Chao A, Chun-Huo C, Jost L (2016). Phylogenetic Diversity Measures and Their Decomposition: A Framework Based on Hill Numbers. Biodiversity Conservation and Phylogenetic Systematics, Springer International Publishing, pp. 141–172, doi:10.1007/978-3-319-22461-9_8.

Chiu, C.H., Wang, Y.T., Walther, B.A. & Chao, A. (2014). Improved nonparametric lower bound of species richness via a modified Good-Turing frequency formula. *Biometrics* 70, 671-682.

Faith D.P. (1992) Conservation evaluation and phylogenetic diversity. *Biological Conservation* 61(1):1-10.

Fisher R.A., Corbet, A.S. & Williams, C.B. (1943) The relation between the number of species and the number of individuals in a random sample of animal population. *Journal of Animal Ecology 12*, 42-58.

Gini C (1921) Measurement of Inequality of Incomes. *The Economic Journal* 31(121): 124-126. doi: 10.2307/2223319

Locey KJ and Lennon JT. (2016) Scaling laws predict global microbial diversity. *PNAS* 113(21):5970-5975; doi:10.1073/pnas.1521291113.

Magurran AE, McGill BJ, eds (2011) Biological Diversity: Frontiers in Measurement and Assessment (Oxford Univ Press, Oxford), Vol 12.

McNaughton, SJ and Wolf LL. (1970). Dominance and the niche in ecological systems. *Science* 167:13, 1–139

O'Hara, R.B. (2005). Species richness estimators: how many species can dance on the head of a pin? *J. Anim. Ecol.* 74, 375-386.

Pielou, EC. (1966) The measurement of diversity in different types of biological collections. *J Theoretical Biology* 13:131–144.

Schloss PD (2024) Rarefaction is currently the best approach to control for uneven sequencing effort in amplicon sequence analyses. *mSphere* 28;9(2):e0035423. doi: 10.1128/msphere.00354-23

Simpson EH (1949) Measurement of Diversity. *Nature* 163(688). doi: 10.1038/163688a0

Smith B and Wilson JB. (1996) A Consumer's Guide to Evenness Indices. *Oikos* 76(1):70-82.

Spellerberg and Fedor (2003). A tribute to Claude Shannon (1916 –2001) and a plea for more rigorous use of species richness, species diversity and the 'Shannon–Wiener' Index. *Alpha Ecology & Biogeography* 12, 177–197.

Cassol, I., Ibañez, M. & Bustamante, J.P. (2025) Key features and guidelines for the application of microbial alpha diversity metrics. *Sci Rep* 15, 622. doi:10.1038/s41598-024-77864-y

See Also

- `plotColData`
- `estimateR`
- `diversity`

Examples

```
data("GlobalPatterns")
tse <- GlobalPatterns

# Calculate the default Shannon index with no rarefaction
tse <- addAlpha(tse, index = "shannon")

# Shows the estimated Shannon index
tse$shannon

# Calculate observed richness with 10 rarefaction rounds
tse <- addAlpha(tse,
   assay.type = "counts",
   index = "observed_richness",
   sample = min(colSums(assay(tse, "counts")), na.rm = TRUE),
   niter=10)

# Shows the estimated observed richness
tse$observed_richness

# One can also calculate the indices and get the results without adding
# them to colData
res <- getAlpha(tse, index = "shannon")
res |> head()
```

---

addContaminantQC *decontam functions*

Description

The decontam functions `isContaminant` and `isNotContaminant` are made available for SummarizedExperiment objects.

Usage

```
addContaminantQC(x, name = "isContaminant", ...)

addNotContaminantQC(x, name = "isNotContaminant", ...)

## S4 method for signature 'SummarizedExperiment'
isContaminant(
  seqtab,
  assay.type = assay_name,
  assay_name = "counts",
  concentration = NULL,
  control = NULL,
  batch = NULL,
```

```
  threshold = 0.1,
  normalize = TRUE,
  detailed = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
isNotContaminant(
  seqtab,
  assay.type = assay_name,
  assay_name = "counts",
  control = NULL,
  threshold = 0.5,
  normalize = TRUE,
  detailed = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
addContaminantQC(x, name = "contaminant", ...)

## S4 method for signature 'SummarizedExperiment'
addNotContaminantQC(x, name = "not_contaminant", ...)
```

Arguments

name	Character scalar. A name for the column of the colData where results will be stored. (Default: "contaminant" or "not_contaminant")
...	• for isContaminant/ isNotContaminant: arguments passed on to decontam:isContaminant or decontam:isNotContaminant • for addContaminantQC/addNotContaminantQC: arguments passed on to isContaminant/ isNotContaminant
seqtab, x	a SummarizedExperiment
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
concentration	Character scalar or NULL. Defining a column with numeric values from the colData to use as concentration information. (Default: NULL)
control	Character scalar or NULL. Defining a column with logical values from the colData to define control and non-control samples. (Default: NULL)
batch	Character scalar or NULL. Defining a column with values interpretable as a factor from the colData to use as batch information. (Default: NULL)
threshold	Numeric scalar.. See decontam:isContaminant or decontam:isNotContaminant
normalize	Logical scalar. See decontam:isContaminant or decontam:isNotContaminant
detailed	Logical scalar. If TRUE, the return value is a data.frame containing diagnostic information on the contaminant decision. If FALSE, the return value is a logical vector containing the binary contaminant classifications. (Default: TRUE)

Value

for `isContaminant`/ `isNotContaminant` a `DataFrame` or for `addContaminantQC`/`addNotContaminantQC` a modified object of `class(x)`

See Also

`decontam:isContaminant`, `decontam:isNotContaminant`

Examples

```
data(esophagus)
# setup of some mock data just for example
colData(esophagus)$concentration <- c(1, 2, 3)
colData(esophagus)$control <- c(FALSE, FALSE, TRUE)

isContaminant(
    esophagus,
    method = "frequency",
    concentration = "concentration"
    )
esophagus <- addContaminantQC(
    esophagus,
    method = "frequency",
    concentration = "concentration"
    )
rowData(esophagus)

isNotContaminant(esophagus, control = "control")
esophagus <- addNotContaminantQC(esophagus, control = "control")
rowData(esophagus)
```

---

addDissimilarity *Calculate dissimilarities*

Description

These functions are designed to calculate dissimilarities on data stored within a `TreeSummarizedExperiment` object. For overlap, Unifrac, and Jensen-Shannon Divergence (JSD) dissimilarities, the functions use mia internal functions, while for other types of dissimilarities, they rely on `vegdist` by default.

Usage

```
addDissimilarity(x, method, ...)

getDissimilarity(x, method, ...)

## S4 method for signature 'SummarizedExperiment'
addDissimilarity(x, method = "bray", name = method, ...)
```

```
## S4 method for signature 'SummarizedExperiment'
getDissimilarity(
  x,
  method = "bray",
  assay.type = "counts",
  niter = NULL,
  transposed = FALSE,
  ...
)

## S4 method for signature 'TreeSummarizedExperiment'
getDissimilarity(
  x,
  method = "bray",
  assay.type = "counts",
  niter = NULL,
  transposed = FALSE,
  ...
)

## S4 method for signature 'ANY'
getDissimilarity(x, method = "bray", niter = NULL, ...)
```

Arguments

x	TreeSummarizedExperiment or matrix.
method	Character scalar. Specifies which dissimilarity to calculate. (Default: "bray")
...	other arguments passed into avgdist, vegdist, or into mia internal functions: • sample: The sampling depth in rarefaction. (Default: min(rowSums2(x))) • dis.fun: Character scalar. Specifies the dissimilarity function to be used. • transf: Function. Specifies the optional transformation applied before calculating the dissimilarity matrix. • tree.name: (Unifrac) Character scalar. Specifies the name of the tree from rowTree(x) that is used in calculation. Disabled when tree is specified. (Default: "phylo") • tree: (Unifrac) phylo. A phylogenetic tree used in calculation. (Default: NULL) • weighted: (Unifrac) Logical scalar. Should use weighted-Unifrac calculation? Weighted-Unifrac takes into account the relative abundance of species/taxa shared between samples, whereas unweighted-Unifrac only considers presence/absence. Default is FALSE, meaning the unweighted-Unifrac dissimilarity is calculated for all pairs of samples. (Default: FALSE) • node.label (Unifrac) character vector. Used only if x is a matrix. Specifies links between rows/columns and tips of tree. All the node labs must be present in tree. For the links, you can provide a vector with whose

length equals to the number of rows/columns in x. Alternatively, you can provide a named vector where names represent names in abundance table and values their corresponding node in tree.

- chunkSize: (JSD) Integer scalar. Defines the size of data send to the individual worker. Only has an effect, if BPPARAM defines more than one worker. (Default: nrow(x))
- BPPARAM: (JSD) BiocParallelParam. Specifies whether the calculation should be parallelized.
- detection: (Overlap) Numeric scalar. Defines detection threshold for absence/presence of features. Feature that has abundance under threshold in either of samples, will be discarded when evaluating overlap between samples. (Default: 0)
- binary: Logical scalar. Whether to perform presence/absence transformation before dissimilarity calculation. For Jaccard index the default is TRUE. For other dissimilarity metrics, please see vegdist.

name — Character scalar. The name to be used to store the result in metadata of the output. (Default: method)

assay.type — Character scalar. Specifies the name of assay used in calculation. (Default: "counts")

niter — Integer scalar. Specifies the number of rarefaction rounds. Rarefaction is not applied when niter=NULL (see Details section). (Default: NULL)

transposed — Logical scalar. Specifies if x is transposed with cells in rows. (Default: FALSE)

Details

Overlap reflects similarity between sample-pairs. When overlap is calculated using relative abundances, the higher the value the higher the similarity is. When using relative abundances, overlap value 1 means that all the abundances of features are equal between two samples, and 0 means that samples have completely different relative abundances.

Unifrac is calculated with ecodive:unweighted_unifrac() or ecodive:weighted_unifrac().

If rarefaction is enabled, vegan:avgdist() is utilized.

Rarefaction can be used to control uneven sequencing depths. Although, it is highly debated method. Some think that it is the only option that successfully controls the variation caused by uneven sampling depths. The biggest argument against rarefaction is the fact that it omits data.

Rarefaction works by sampling the counts randomly. This random sampling is done niter times. In each sampling iteration, sample number of random samples are drawn, and dissimilarity is calculated for this subset. After the iterative process, there are niter number of result that are then averaged to get the final result.

Refer to Schloss (2024) for more details on rarefaction.

Value

getDissimilarity returns a sample-by-sample dissimilarity matrix.

addDissimilarity returns x that includes dissimilarity matrix in its metadata.

References

For unifrac dissimilarity: http://bmf.colorado.edu/unifrac/

See also additional descriptions of Unifrac in the following articles:

Lozupone, Hamady and Knight, "Unifrac - An Online Tool for Comparing Microbial Community Diversity in a Phylogenetic Context.", BMC Bioinformatics 2006, 7:371

Lozupone, Hamady, Kelley and Knight, "Quantitative and qualitative (beta) diversity measures lead to different insights into factors that structure microbial communities." Appl Environ Microbiol. 2007

Lozupone C, Knight R. "Unifrac: a new phylogenetic method for comparing microbial communities." Appl Environ Microbiol. 2005 71 (12):8228-35.

For JSD dissimilarity: Jensen-Shannon Divergence and Hilbert space embedding. Bent Fuglede and Flemming Topsoe University of Copenhagen, Department of Mathematics http://www.math.ku.dk/~topsoe/ISIT2004JSD.pdf

For rarefaction: Schloss PD (2024) Rarefaction is currently the best approach to control for uneven sequencing effort in amplicon sequence analyses. *mSphere* 28;9(2):e0035423. doi: 10.1128/msphere.00354-23

See Also

http://en.wikipedia.org/wiki/Jensen-Shannon_divergence

Examples

```
library(mia)
library(scater)

# load dataset
data(GlobalPatterns)
tse <- GlobalPatterns

### Overlap dissimilarity

tse <- addDissimilarity(tse, method = "overlap", detection = 0.25)
metadata(tse)[["overlap"]][1:6, 1:6]

### JSD dissimilarity

tse <- addDissimilarity(tse, method = "jsd")
metadata(tse)[["jsd"]][1:6, 1:6]

# Multi Dimensional Scaling applied to JSD dissimilarity matrix
tse <- addMDS(tse, method = "overlap", assay.type = "counts")
reducedDim(tse, "MDS") |> head()

### Unifrac dissimilarity

res <- getDissimilarity(tse, method = "unifrac", weighted = FALSE)
dim(as.matrix(res))
```

```
tse <- addDissimilarity(tse, method = "unifrac", weighted = TRUE)
metadata(tse)[["unifrac"]][1:6, 1:6]

### Bray dissimilarity

# Bray is usually applied to relative abundances so we have to apply
# transformation first
tse <- transformAssay(tse, method = "relabundance")
res <- getDissimilarity(tse, method = "bray", assay.type = "relabundance")
as.matrix(res)[1:6, 1:6]

# If applying rarefaction, the input must be count matrix and transformation
# method specified in function call (Note: increase niter)
rclr <- function(x){
    vegan::decostand(x, method="rclr")
}
res <- getDissimilarity(
    tse, method = "euclidean", transf = rclr, niter = 2L)
as.matrix(res)[1:6, 1:6]
```

---

addDivergence *Estimate divergence*

Description

Estimate divergence against a given reference sample.

Usage

```
addDivergence(x, name = "divergence", ...)

getDivergence(
  x,
  assay.type = assay_name,
  assay_name = "counts",
  reference = "median",
  method = "bray",
  ...
)

## S4 method for signature 'SummarizedExperiment'
addDivergence(x, name = "divergence", ...)

## S4 method for signature 'SummarizedExperiment'
getDivergence(
  x,
  assay.type = assay_name,
  assay_name = "counts",
```

```
  reference = "median",
  method = "bray",
  ...
)
```

Arguments

x	a SummarizedExperiment object.
name	Character scalar. The name to be used to store the result in metadata of the output. (Default: method)
...	optional arguments passed to addDissimilarity. Additionally: • dimred: Character scalar. Specifies the name of dimension reduction result from reducedDim(x). If used, these values are used to calculate divergence instead of the assay. Can be disabled with NULL. (Default: NULL)
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
reference	Character scalar. A column name from colData(x) or either "mean" or "median". If column name is specified, the column must include reference samples for each sample. If "mean" or "median" is specified, the mean or median of the entire dataset is calculated and used as the reference value. (Default: "median")
method	Character scalar. Specifies which dissimilarity to calculate. (Default: "bray")

Details

Microbiota divergence (heterogeneity / spread) within a given sample set can be quantified by the average sample dissimilarity or beta diversity with respect to a given reference sample.

The calculation makes use of the function getDissimilarity(). The divergence measure is sensitive to sample size. Subsampling or bootstrapping can be applied to equalize sample sizes between comparisons.

Value

x with additional colData named name

See Also

- addAlpha
- addDissimilarity
- plotColData

Examples

```
data(GlobalPatterns)
tse <- GlobalPatterns

# By default, reference is median of all samples. The name of column where
# results is "divergence" by default, but it can be specified.
tse <- addDivergence(tse)

# The method that are used to calculate distance in divergence and
# reference can be specified. Here, euclidean distance is used. Reference is
# the first sample. It is recommended # to add reference to colData.
tse[["reference"]] <- rep(colnames(tse)[[1]], ncol(tse))
tse <- addDivergence(
    tse, name = "divergence_first_sample",
    reference = "reference",
    method = "euclidean")

# Here we compare samples to global mean
tse <- addDivergence(tse, name = "divergence_average", reference = "mean")

# All three divergence results are stored in colData.
colData(tse)
```

---

addMediation *Perform mediation analysis*

Description

getMediation and addMediation provide a wrapper of mediate for SummarizedExperiment.

Usage

```
addMediation(x, ...)

getMediation(x, ...)

## S4 method for signature 'SummarizedExperiment'
addMediation(
  x,
  outcome,
  treatment,
  name = "mediation",
  mediator = NULL,
  assay.type = NULL,
  dimred = NULL,
  family = gaussian(),
  covariates = NULL,
```

```
  p.adj.method = "holm",
  add.metadata = TRUE,
  verbose = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
getMediation(
  x,
  outcome,
  treatment,
  mediator = NULL,
  assay.type = NULL,
  dimred = NULL,
  family = gaussian(),
  covariates = NULL,
  p.adj.method = "holm",
  add.metadata = TRUE,
  sort = FALSE,
  verbose = TRUE,
  ...
)
```

Arguments

x	a SummarizedExperiment.
...	additional parameters that can be passed to mediate.
outcome	Character scalar. Indicates the colData variable used as outcome in the model.
treatment	Character scalar. Indicates the colData variable used as treatment in the model.
name	Character scalar. A name for the column of the colData where results will be stored. (Default: "mediation")
mediator	Character scalar. Indicates the colData variable used as mediator in the model. (Default: NULL)
assay.type	Character scalar. Specifies the assay used for feature-wise mediation analysis. (Default: NULL)
dimred	Character scalar. Indicates the reduced dimension result in reducedDims(object) for component-wise mediation analysis. (Default: NULL)
family	Character scalar. A specification for the outcome model link function. (Default: gaussian("identity"))
covariates	Character scalar or character vector. Indicates the colData variables used as covariates in the model. (Default: NULL)
p.adj.method	Character scalar. Selects adjustment method of p-values. Passed to p.adjust function. (Default: "holm")
add.metadata	Logical scalar. Should the model metadata be returned. (Default: TRUE)

verbose	Logical scalar. Should execution messages be printed. (Default: TRUE)
sort	Logical scalar. Should the results be sorted by decreasing significance in terms of ACME_pval. (Default: FALSE)

Details

This wrapper of mediate for SummarizedExperiment provides a simple method to analyse the effect of a treatment variable on an outcome variable found in colData(se) through the mediation of either another variable in colData (argument mediator) or an assay (argument assay.type) or a reducedDim (argument dimred). Importantly, those three arguments are mutually exclusive.

Value

getMediation returns a data.frame of adjusted p-values and effect sizes for the ACMEs and ADEs of every mediator given as input, whereas addMediation returns an updated SummarizedExperiment instance with the same data.frame stored in the metadata as "mediation" or as specified in the name argument. Its columns include:

mediator the mediator variable

acme the Average Causal Mediation Effect (ACME) estimate

acme_pval the original p-value for the ACME estimate

acme_lower the lower bound of the CI for the ACME estimate

acme_upper the upper bound of the CI for the ACME estimate

ade the Average Direct Effect (ADE) estimate

ade_pval the original p-value for the ADE estimate

ade_lower the lower bound of the CI for the ADE estimate

ade_upper the upper bound of the CI for the ADE estimate

total the Total Effect estimate

total_lower the lower bound of the CI for the Total Effect estimate

total_upper the upper bound of the CI for the Total Effect estimate

total_pval the original p-value for the Total Effect estimate

acme_padj the adjusted p-value for the ACME estimate

ade_padj the adjusted p-value for the ADE estimate

total_padj the adjusted p-value for the Total Effect estimate

The original output of mediate for each analysed mediator is stored as the "model_metadata" attribute of the resulting data.frame and can be accessed via the attr function.

Examples

```
## Not run:
# Import libraries
library(mia)
library(miaViz)
library(scater)
```

```
# Load dataset
data(hitchip1006, package = "miaTime")
tse <- hitchip1006

# Agglomerate features by family (merely to speed up execution)
tse <- agglomerateByRank(tse, rank = "Phylum")
# Convert BMI variable to numeric
tse$bmi_group <- as.numeric(tse$bmi_group)

# Analyse mediated effect of nationality on BMI via alpha diversity
# 100 permutations were done to speed up execution, but ~1000 are recommended
med_df <- getMediation(
    tse,
    outcome = "bmi_group",
    treatment = "nationality",
    mediator = "diversity",
    covariates = c("sex", "age"),
    treat.value = "Scandinavia",
    control.value = "CentralEurope",
    boot = TRUE, sims = 100)

# Visualise model statistics
plotMediation(med_df)

# Apply clr transformation to counts assay
tse <- transformAssay(tse, method = "clr", pseudocount = 1)

# Analyse mediated effect of nationality on BMI via clr-transformed features
# 100 permutations were done to speed up execution, but ~1000 are recommended
tse <- addMediation(
    tse, name = "assay_mediation",
    outcome = "bmi_group",
    treatment = "nationality",
    assay.type = "clr",
    covariates = c("sex", "age"),
    treat.value = "Scandinavia",
    control.value = "CentralEurope",
    boot = TRUE, sims = 100,
    p.adj.method = "fdr")

# Show results for first 5 mediators
head(metadata(tse)$assay_mediation, 5)

# Perform ordination
tse <- addMDS(
    tse, name = "MDS", method = "euclidean", assay.type = "clr",
    ncomponents = 3)

# Analyse mediated effect of nationality on BMI via NMDS components
# 100 permutations were done to speed up execution, but ~1000 are recommended
tse <- addMediation(
    tse, name = "reddim_mediation",
```

```
    outcome = "bmi_group",
    treatment = "nationality",
    dimred = "MDS",
    covariates = c("sex", "age"),
    treat.value = "Scandinavia",
    control.value = "CentralEurope",
    boot = TRUE, sims = 100,
    p.adj.method = "fdr")

# Show results for first 5 mediators
head(metadata(tse)$reddim_mediation, 5)

# Access model metadata
attr(metadata(tse)$reddim_mediation, "model_metadata")

## End(Not run)
```

---

agglomerateByPrevalence *Agglomerate data based on population prevalence*

Description

Agglomerate data based on population prevalence

Usage

```
agglomerateByPrevalence(x, ...)

## S4 method for signature 'SummarizedExperiment'
agglomerateByPrevalence(
  x,
  rank = NULL,
  other.name = other_label,
  other_label = "Other",
  ...
)

## S4 method for signature 'TreeSummarizedExperiment'
agglomerateByPrevalence(
  x,
  rank = NULL,
  other.name = other_label,
  other_label = "Other",
  update.tree = TRUE,
  ...
)
```

Arguments

x	TreeSummarizedExperiment.
...	arguments passed to agglomerateByRank function for SummarizedExperiment objects and other functions. See agglomerateByRank for more details.
rank	Character scalar. Defines a taxonomic rank. Must be a value of taxonomyRanks() function.
other.name	Character scalar. Used as the label for the summary of non-prevalent taxa. (default: "Other")
other_label	Deprecated. use other.name instead.
update.tree	Logical scalar. Should rowTree() also be merged? (Default: TRUE)

Details

agglomerateByPrevalence sums up the values of assays at the taxonomic level specified by rank (by default the highest taxonomic level available) and selects the summed results that exceed the given population prevalence at the given detection level. The other summed values (below the threshold) are agglomerated in an additional row taking the name indicated by other.name (by default "Other").

Value

agglomerateByPrevalence returns a taxonomically-agglomerated object of the same class as x and based on prevalent taxonomic results.

Examples

```
## Data can be aggregated based on prevalent taxonomic results
data(GlobalPatterns)
tse <- GlobalPatterns
tse <- transformAssay(tse, method = "relabundance")
tse <- agglomerateByPrevalence(
    tse,
    rank = "Phylum",
    assay.type = "relabundance",
    detection = 1/100,
    prevalence = 50/100)

tse

# Here data is aggregated at the taxonomic level "Phylum". The five phyla
# that exceed the population prevalence threshold of 50/100 represent the
# five first rows of the assay in the aggregated data. The sixth and last row
# named by default "Other" takes the summed up values of all the other phyla
# that are below the prevalence threshold.

assay(tse)[,1:5]
```

agglomerateByRank *Agglomerate data using taxonomic information or other grouping*

Description

Agglomeration functions can be used to sum-up data based on specific criteria such as taxonomic ranks, variables or prevalence.

`agglomerateByRank` can be used to sum up data based on associations with certain taxonomic ranks, as defined in `rowData`. Only available `taxonomyRanks` can be used.

`agglomerateByVariable` and `agglomerateByModule` merge data on rows or columns of a `SummarizedExperiment` as defined by a `factor` alongside the chosen dimension. This function allows agglomeration of data based on other variables than taxonomy ranks. Metadata from the `rowData` or `colData` are retained as defined by `archetype`. `assay` are agglomerated, i.e. summed up. If the assay contains values other than counts or absolute values, this can lead to meaningless values being produced.

`agglomerateByRanks` takes a `SummarizedExperiment`, splits it along the taxonomic ranks, aggregates the data per rank, converts the input to a `SingleCellExperiment` objects and stores the aggregated data as alternative experiments. `unsplitByRanks` takes these alternative experiments and flattens them again into a single `SummarizedExperiment`.

Usage

```
agglomerateByRank(x, ...)

agglomerateByVariable(x, ...)

agglomerateByModule(x, ...)

agglomerateByRanks(x, ...)

unsplitByRanks(x, ...)

## S4 method for signature 'TreeSummarizedExperiment'
agglomerateByRank(
  x,
  rank = taxonomyRanks(x)[1],
  update.tree = agglomerateTree,
  agglomerate.tree = agglomerateTree,
  agglomerateTree = TRUE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
agglomerateByRank(
  x,
  rank = taxonomyRanks(x)[1],
  altexp = NULL,
```

```
  altexp.rm = strip_altexp,
  strip_altexp = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
agglomerateByRank(
  x,
  rank = taxonomyRanks(x)[1],
  empty.rm = TRUE,
  empty.fields = c(NA, "", " ", "\t", "-", "_"),
  ...
)

## S4 method for signature 'TreeSummarizedExperiment'
agglomerateByVariable(
  x,
  by,
  group = f,
  f,
  update.tree = mergeTree,
  mergeTree = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
agglomerateByVariable(x, by, group = f, f, ...)

## S4 method for signature 'SummarizedExperiment'
agglomerateByModule(x, by, group, na.rm = FALSE)

## S4 method for signature 'SummarizedExperiment'
agglomerateByRanks(
  x,
  ranks = taxonomyRanks(x),
  na.rm = TRUE,
  as.list = FALSE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
agglomerateByRanks(
  x,
  ranks = taxonomyRanks(x),
  na.rm = TRUE,
  as.list = FALSE,
  ...
)
```

```
## S4 method for signature 'TreeSummarizedExperiment'
agglomerateByRanks(
  x,
  ranks = taxonomyRanks(x),
  na.rm = TRUE,
  as.list = FALSE,
  ...
)

splitByRanks(x, ...)

## S4 method for signature 'SingleCellExperiment'
unsplitByRanks(
  x,
  ranks = taxonomyRanks(x),
  keep.dimred = keep_reducedDims,
  keep_reducedDims = FALSE,
  ...
)

## S4 method for signature 'TreeSummarizedExperiment'
unsplitByRanks(
  x,
  ranks = taxonomyRanks(x),
  keep.dimred = keep_reducedDims,
  keep_reducedDims = FALSE,
  ...
)
```

Arguments

x	TreeSummarizedExperiment.
...	arguments passed to agglomerateByRank function for SummarizedExperiment objects and other functions. See agglomerateByRank for more details.
rank	Character scalar. Defines a taxonomic rank. Must be a value of taxonomyRanks() function.
update.tree	Logical scalar. Should rowTree() also be merged? (Default: TRUE)
agglomerate.tree	Deprecated. Use update.tree instead.
agglomerateTree	Deprecated. Use update.tree instead.
altexp	Character scalar or integer scalar. Specifies an alternative experiment containing the input data.
altexp.rm	Logical scalar. Should alternative experiments be removed prior to agglomeration? This prevents too many nested alternative experiments by default. (Default: TRUE)

strip_altexp — Deprecated. Use altexp.rm instead.

empty.rm — Logical scalar. Defines whether rows including empty.fields in specified rank will be excluded. (Default: TRUE)

empty.fields — Character vector. Defines which values should be regarded as empty. (Default: c(NA, "", " ", "\t")). They will be removed if na.rm = TRUE before agglomeration.

by — Character scalar. Determines if data is merged row-wise / for features ('rows') or column-wise / for samples ('cols'). Must be 'rows' or 'cols'.

group — Character scalar, character vector or factor vector. A column name from rowData(x) or colData(x) or alternatively a vector specifying how the merging is performed. If vector, the value must be the same length as nrow(x)/ncol(x). Rows or columns corresponding to the same level will be merged. If length(levels(group)) == nrow(x)/ncol(x), x will be returned unchanged. For agglomerateByModule, group should specify one or several names of logical or numeric binary variables from the rowData(x)/colData(x) by which to agglomerate rows or columns.

f — Deprecated. Use group instead.

mergeTree — Deprecated. Use update.tree instead.

na.rm — Logical scalar. Should NA values be omitted? (Default: TRUE)

ranks — Character vector. Defines taxonomic ranks. Must all be values of taxonomyRanks() function.

as.list — Logical scalar. Should the list of SummarizedExperiment objects be returned by the function agglomerateByRanks as a SimpleList or stored in altExps? (Default: FALSE)

keep.dimred — Logical scalar. Should the reducedDims(x) be transferred to the result? Please note, that this breaks the link between the data used to calculate the reduced dims. (Default: FALSE)

keep_reducedDims — Deprecated. Use keep.dimred instead.

Details

Agglomeration sums up the values of assays at the specified taxonomic level. With certain assays, e.g. those that include binary or negative values, this summing can produce meaningless values. In those cases, consider performing agglomeration first, and then applying the transformation afterwards.

agglomerateByVariable works similarly to aggregateAcrossGenes. However, additional support for TreeSummarizedExperiment was added and science field agnostic names were used. In addition the archetype argument lets the user select how to preserve row or column data. For merge data of assays the function from scuttle are used.

agglomerateByModule allows to agglomerate features or samples based on one or multiple variables of logical or numeric binary (0/1) type. It is particularly useful for agglomerating by taxonomic or functional modules, each defined by a logical or binary variable in the rowData, as features can belong to several modules.

agglomerateByRanks will use by default all available taxonomic ranks, but this can be controlled by setting ranks manually. NA values are removed by default, since they would not make sense, if

the result should be used for unsplitByRanks at some point. The input data remains unchanged in the returned SingleCellExperiment objects.

unsplitByRanks will remove any NA value on each taxonomic rank so that no ambiguous data is created. In additional, a column taxonomicLevel is created or overwritten in the rowData to specify from which alternative experiment this originates from. This can also be used for splitAltExps to split the result along the same factor again. The input data from the base objects is not returned, only the data from the altExp(). Be aware that changes to rowData of the base object are not returned, whereas only the colData of the base object is kept.

Value

agglomerateByRank returns a taxonomically-agglomerated, optionally-pruned object of the same class as x. agglomerateByVariable and agglomerateByModule return an object of the same class as x with the specified entries merged into one entry in all relevant components.

For agglomerateByRanks: If as.list = TRUE : SummarizedExperiment objects in a SimpleList If as.list = FALSE : The SummarizedExperiment passed as a parameter and now containing the SummarizedExperiment objects in its altExps

For unsplitByRanks: x, with rowData and assay data replaced by the unsplit data. colData of x is kept as well and any existing rowTree is dropped as well, since existing rowLinks are not valid anymore.

See Also

splitOn unsplitOn agglomerateByVariable, agglomerateByRank, altExps, splitAltExps

Examples

```
### Agglomerate data based on taxonomic information

data(GlobalPatterns)
# print the available taxonomic ranks
colnames(rowData(GlobalPatterns))
taxonomyRanks(GlobalPatterns)

# agglomerate at the Family taxonomic rank
x1 <- agglomerateByRank(GlobalPatterns, rank="Family")
## How many taxa before/after agglomeration?
nrow(GlobalPatterns)
nrow(x1)

# Do not agglomerate the tree
x2 <- agglomerateByRank(
    GlobalPatterns, rank="Family", update.tree = FALSE)
nrow(x2) # same number of rows, but
rowTree(x1) # ... different
rowTree(x2) # ... tree

# If assay contains binary or negative values, summing might lead to
# meaningless values, and you will get a warning. In these cases, you might
# want to do agglomeration again at chosen taxonomic level.
```

```
tse <- transformAssay(GlobalPatterns, method = "pa")
tse <- agglomerateByRank(tse, rank = "Genus")
tse <- transformAssay(tse, method = "pa")

# Removing empty labels by setting empty.rm = TRUE
sum(is.na(rowData(GlobalPatterns)$Family))
x3 <- agglomerateByRank(GlobalPatterns, rank="Family", empty.rm = TRUE)
nrow(x3) # different from x2

# Because all the rownames are from the same rank, rownames do not include
# prefixes, in this case "Family:".
print(rownames(x3[1:3,]))

# To add them, use getTaxonomyLabels function.
rownames(x3) <- getTaxonomyLabels(x3, with.rank = TRUE)
print(rownames(x3[1:3,]))

# use 'empty.ranks.rm' to remove columns that include only NAs
x4 <- agglomerateByRank(
    GlobalPatterns, rank="Phylum", empty.ranks.rm = TRUE)
head(rowData(x4))

# If the assay contains NAs, you might want to specify na.rm=TRUE,
# since summing-up NAs lead to NA
x5 <- GlobalPatterns
# Replace first value with NA
assay(x5)[1,1] <- NA
x6 <- agglomerateByRank(x5, "Kingdom")
head( assay(x6) )
# Use na.rm=TRUE
x6 <- agglomerateByRank(x5, "Kingdom", na.rm = TRUE)
head( assay(x6) )

## Look at enterotype dataset...
data(enterotype)
## Print the available taxonomic ranks. Shows only 1 available rank,
## not useful for agglomerateByRank
taxonomyRanks(enterotype)

### Merge TreeSummarizedExperiments on rows and columns

data(esophagus)
esophagus
plot(rowTree(esophagus))
# Get a factor for merging
f <- factor(regmatches(rownames(esophagus),
    regexpr("^[0-9]*_[0-9]*",rownames(esophagus))))
merged <- agglomerateByVariable(
    esophagus, by = "rows", f, update.tree = TRUE)
plot(rowTree(merged))
#
data(GlobalPatterns)
GlobalPatterns
```

```
merged <- agglomerateByVariable(
    GlobalPatterns, by = "cols", colData(GlobalPatterns)$SampleType)
merged

## Agglomerate by multiple modules

# Generate 30 random modules
N_module <- 30L
modules <- sample(
    c(TRUE, FALSE),
    size = nrow(tse) * N_module,
    prob = c(0.2, 0.8),
    replace = TRUE
)

# Convert modules to matrix
modules <- matrix(modules, nrow = nrow(tse))

# Add module names as colnames
colnames(modules) <- paste0("module_", seq_len(ncol(modules)))

# Add modules to rowData
rowData(tse) <- cbind(rowData(tse), modules)

# Extract module columns
module_columns <- grep("module_", colnames(rowData(tse)), value = TRUE)

# Agglomerate based on modules
tse_module <- agglomerateByModule(tse, by = 1, group = module_columns)

# Optionally, store results into altExp slot
altExp(tse, "modules") <- tse_module

data(GlobalPatterns)
# print the available taxonomic ranks
taxonomyRanks(GlobalPatterns)

# agglomerateByRanks
#
tse <- agglomerateByRanks(GlobalPatterns)
altExps(tse)
altExp(tse,"Kingdom")
altExp(tse,"Species")

# unsplitByRanks
tse <- unsplitByRanks(tse)
tse
```

`calculateDMN` *Dirichlet-Multinomial Mixture Model: Machine Learning for Microbiome Data*

Description

These functions are accessors for functions implemented in the DirichletMultinomial package

Usage

```
calculateDMN(x, ...)

getDMN(x, name = "DMN", ...)

bestDMNFit(x, name = "DMN", type = c("laplace", "AIC", "BIC"), ...)

calculateDMNgroup(x, ...)

performDMNgroupCV(x, ...)

getBestDMNFit(x, name = "DMN", type = c("laplace", "AIC", "BIC"), ...)

## S4 method for signature 'ANY'
calculateDMN(
  x,
  k = 1,
  BPPARAM = SerialParam(),
  seed = runif(1, 0, .Machine$integer.max),
  ...
)

## S4 method for signature 'SummarizedExperiment'
calculateDMN(
  x,
  assay.type = assay_name,
  assay_name = exprs_values,
  exprs_values = "counts",
  transposed = FALSE,
  ...
)

runDMN(x, name = "DMN", ...)

## S4 method for signature 'SummarizedExperiment'
getDMN(x, name = "DMN")

## S4 method for signature 'SummarizedExperiment'
bestDMNFit(x, name = "DMN", type = c("laplace", "AIC", "BIC"))

## S4 method for signature 'SummarizedExperiment'
getBestDMNFit(x, name = "DMN", type = c("laplace", "AIC", "BIC"))

## S4 method for signature 'ANY'
```

```
calculateDMNgroup(
  x,
  variable,
  k = 1,
  seed = runif(1, 0, .Machine$integer.max),
  ...
)

## S4 method for signature 'SummarizedExperiment'
calculateDMNgroup(
  x,
  variable,
  assay.type = assay_name,
  assay_name = exprs_values,
  exprs_values = "counts",
  transposed = FALSE,
  ...
)

## S4 method for signature 'ANY'
performDMNgroupCV(
  x,
  variable,
  k = 1,
  seed = runif(1, 0, .Machine$integer.max),
  ...
)

## S4 method for signature 'SummarizedExperiment'
performDMNgroupCV(
  x,
  variable,
  assay.type = assay_name,
  assay_name = exprs_values,
  exprs_values = "counts",
  transposed = FALSE,
  ...
)
```

Arguments

x	a numeric matrix with samples as rows or a SummarizedExperiment object.
...	optional arguments not used.
name	Character scalar. The name to store the result in metadata
type	Character scalar. The type of measure used for the goodness of fit. One of 'laplace', 'AIC' or 'BIC'.
k	Numeric scalar. The number of Dirichlet components to fit. See dmn. (Default: 1)

BPPARAM	A BiocParallelParam object specifying whether the calculation should be parallelized.
seed	Numeric scalar. Random number seed. See dmn
assay.type	Character scalar. Specifies the name of the assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
exprs_values	Deprecated. Use assay.type instead.
transposed	Logical scalar. Is x transposed with samples in rows? (Default: FALSE)
variable	Character scalar. A variable from colData to use as a grouping variable. Must be a character of factor.

Value

calculateDMN and getDMN return a list of DMN objects, one element for each value of k provided.

bestDMNFit returns the index for the best fit and getBestDMNFit returns a single DMN object.

calculateDMNgroup returns a DMNGroup object

performDMNgroupCV returns a data.frame

See Also

DMN-class, DMNGroup-class, dmn, dmngroup, cvdmngroup , accessors for DMN objects

Examples

```
fl <- system.file(package="DirichletMultinomial", "extdata", "Twins.csv")
counts <- as.matrix(read.csv(fl, row.names=1))
fl <- system.file(package="DirichletMultinomial", "extdata", "TwinStudy.t")
pheno0 <- scan(fl)
lvls <- c("Lean", "Obese", "Overwt")
pheno <- factor(lvls[pheno0 + 1], levels=lvls)
colData <- DataFrame(pheno = pheno)

tse <- TreeSummarizedExperiment(
    assays = list(counts = counts),
    colData = colData)

library(bluster)

# Compute DMM algorithm and store result in metadata
tse <- addCluster(
    tse,
    assay.type = "counts",
    name = "DMM",
    DmmParam(k = 1:3, type = "laplace"),
    by = "samples",
    full = TRUE
)
```

```
# Get the list of DMN objects
metadata(tse)$DMM$dmm

# Get and display which objects fits best
bestFit <- metadata(tse)$DMM$best
bestFit

# Get the model that generated the best fit
bestModel <- metadata(tse)$DMM$dmm[[bestFit]]
bestModel

# Get the sample-cluster assignment probability matrix
head(metadata(tse)$DMM$prob)

# Get the weight of each component for the best model
bestModel@mixture$Weight
```

---

convertFromDADA2 *Create a* TreeSummarizedExperiment *object from 'DADA2' results*

Description

Create a TreeSummarizedExperiment object from 'DADA2' results

Usage

```
convertFromDADA2(...)
```

Arguments

...	Additional arguments. For convertFromDADA2, see mergePairs function for more details.

Details

convertFromDADA2 is a wrapper for the mergePairs function from the dada2 package. A count matrix is constructed via makeSequenceTable(mergePairs(...)) and rownames are dynamically created as ASV(N) with N from 1 to nrow of the count tables. The colnames and rownames from the output of makeSequenceTable are stored as colnames and in the referenceSeq slot of the TreeSummarizedExperiment, respectively.

Value

convertFromDADA2 returns an object of class TreeSummarizedExperiment

Examples

```
### Coerce DADA2 results to a TreeSE object
if(requireNamespace("dada2")) {
  fnF <- system.file("extdata", "sam1F.fastq.gz", package="dada2")
  fnR = system.file("extdata", "sam1R.fastq.gz", package="dada2")
  dadaF <- dada2::dada(fnF, selfConsist=TRUE)
  dadaR <- dada2::dada(fnR, selfConsist=TRUE)

  tse <- convertFromDADA2(dadaF, fnF, dadaR, fnR)
  tse
}
```

---

convertToBIOM *Convert a* TreeSummarizedExperiment *object to/from 'BIOM' results*

Description

For convenience, a few functions are available to convert BIOM, DADA2 and phyloseq objects to TreeSummarizedExperiment objects, and TreeSummarizedExperiment objects to phyloseq objects.

Usage

```
convertToBIOM(x, assay.type = "counts", ...)

importBIOM(file, ...)

convertFromBIOM(
  x,
  prefix.rm = removeTaxaPrefixes,
  removeTaxaPrefixes = FALSE,
  rank.from.prefix = rankFromPrefix,
  rankFromPrefix = FALSE,
  artifact.rm = remove.artifacts,
  remove.artifacts = FALSE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
convertToBIOM(x, assay.type = "counts", ...)
```

Arguments

x	TreeSummarizedExperiment
assay.type	Character scaler. The name of assay. (Default: "counts")
...	Additional arguments. Not used currently.

`file` BIOM file location

`prefix.rm` `Logical scalar`. Should taxonomic prefixes be removed? The prefixes is removed only from detected taxa columns meaning that `rank.from.prefix` should be enabled in the most cases. (Default: `FALSE`)

`removeTaxaPrefixes`
Deprecated. Use `prefix.rm` instead.

`rank.from.prefix`
`Logical scalar`. If file does not have taxonomic ranks on feature table, should they be scraped from prefixes? (Default: `FALSE`)

`rankFromPrefix` Deprecated.Use `rank.from.prefix` instead.

`artifact.rm` `Logical scalar`. If file have some taxonomic character naming artifacts, should they be removed. (default (Default: `FALSE`)

`remove.artifacts`
Deprecated. Use `artifact.rm` instead.

Details

`convertFromBIOM` coerces a `biom` object to a `TreeSummarizedExperiment` object.

`convertToBIOM` coerces a `TreeSummarizedExperiment` object to a `biom` object.

`importBIOM` loads a BIOM file and creates a `TreeSummarizedExperiment` object from the BIOM object contained in the loaded file.

Value

`convertFromBIOM` returns an object of class `TreeSummarizedExperiment`

`importBIOM` returns an object of class `TreeSummarizedExperiment`

See Also

`importQIIME2` `importMothur`

`importMetaPhlAn` `convertFromPhyloseq` `convertFromBIOM` `convertFromDADA2` `importQIIME2` `importMothur` `importHUMAnN`

Examples

```
# Convert BIOM results to a TreeSE
# Load biom file
library(biomformat)
biom_file <- system.file("extdata", "rich_dense_otu_table.biom",
                         package = "biomformat")

# Make TreeSE from BIOM object
biom_object <- biomformat::read_biom(biom_file)
tse <- convertFromBIOM(biom_object)

# Convert TreeSE object to BIOM
biom <- convertToBIOM(tse)
```

```
# Load biom file
library(biomformat)
biom_file <- system.file(
    "extdata", "rich_dense_otu_table.biom", package = "biomformat")

# Make TreeSE from biom file
tse <- importBIOM(biom_file)

# Get taxonomyRanks from prefixes and remove prefixes
tse <- importBIOM(
    biom_file, rank.from.prefix = TRUE, prefix.rm = TRUE)

# Load another biom file
biom_file <- system.file(
   "extdata", "Aggregated_humanization2.biom", package = "mia")

# Clean artifacts from taxonomic data
tse <- importBIOM(biom_file, artifact.rm = TRUE)
```

---

convertToPhyloseq *Create a* TreeSummarizedExperiment *object from a phyloseq object*

Description

Create a TreeSummarizedExperiment object from a phyloseq object

Create a phyloseq object from a TreeSummarizedExperiment object

Usage

```
convertToPhyloseq(x, ...)

convertFromPhyloseq(x)

## S4 method for signature 'SummarizedExperiment'
convertToPhyloseq(x, assay.type = "counts", assay_name = NULL, ...)

## S4 method for signature 'TreeSummarizedExperiment'
convertToPhyloseq(x, tree.name = tree_name, tree_name = "phylo", ...)
```

Arguments

x	a TreeSummarizedExperiment object
...	Additional arguments. Not used currently.
assay.type	Character scalar. Specifies the name of assay used. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
tree.name	Character scalar. Specifies the name of the tree to be included in the phyloseq object that is created, (Default: "phylo")
tree_name	Deprecated. Use tree.name instead.

Details

convertFromPhyloseq converts phyloseq objects into TreeSummarizedExperiment objects. All data stored in a phyloseq object is transferred.

convertToPhyloseq creates a phyloseq object from a TreeSummarizedExperiment object. By using assay.type, it is possible to specify which table from assay is added to the phyloseq object.

Value

convertFromPhyloseq returns an object of class TreeSummarizedExperiment

convertToPhyloseq returns an object of class phyloseq

Examples

```
### Coerce a phyloseq object to a TreeSE object
if (requireNamespace("phyloseq")) {
    data(GlobalPatterns, package="phyloseq")
    convertFromPhyloseq(GlobalPatterns)
    data(enterotype, package="phyloseq")
    convertFromPhyloseq(enterotype)
    data(esophagus, package="phyloseq")
    convertFromPhyloseq(esophagus)
}

### Coerce a TreeSE object to a phyloseq object
# Get tse object
data(GlobalPatterns)
tse <- GlobalPatterns

# Create a phyloseq object from it
phy <- convertToPhyloseq(tse)
phy

# By default the chosen table is counts, but if there are other tables,
# they can be chosen with assay.type.

# Counts relative abundances table
tse <- transformAssay(tse, method = "relabundance")
phy2 <- convertToPhyloseq(tse, assay.type = "relabundance")
phy2
```

---

deprecate *These functions will be deprecated. Please use other functions instead.*

Description

These functions will be deprecated. Please use other functions instead.

Usage

```
cluster(x, ...)

## S4 method for signature 'SummarizedExperiment'
cluster(x, ...)

addTaxonomyTree(x, ...)

## S4 method for signature 'SummarizedExperiment'
addTaxonomyTree(x, ...)

taxonomyTree(x, ...)

## S4 method for signature 'SummarizedExperiment'
taxonomyTree(x, ...)

mergeRows(x, ...)

## S4 method for signature 'SummarizedExperiment'
mergeRows(x, ...)

## S4 method for signature 'TreeSummarizedExperiment'
mergeRows(x, ...)

mergeCols(x, ...)

## S4 method for signature 'SummarizedExperiment'
mergeCols(x, ...)

## S4 method for signature 'TreeSummarizedExperiment'
mergeCols(x, ...)

mergeFeatures(x, ...)

## S4 method for signature 'SummarizedExperiment'
mergeFeatures(x, ...)

## S4 method for signature 'TreeSummarizedExperiment'
mergeFeatures(x, ...)

mergeSamples(x, ...)

## S4 method for signature 'SummarizedExperiment'
mergeSamples(x, ...)

## S4 method for signature 'TreeSummarizedExperiment'
mergeSamples(x, ...)
```

```
mergeFeaturesByRank(x, ...)

## S4 method for signature 'SummarizedExperiment'
mergeFeaturesByRank(x, ...)

## S4 method for signature 'SingleCellExperiment'
mergeFeaturesByRank(x, ...)

mergeFeaturesByPrevalence(x, ...)

## S4 method for signature 'SummarizedExperiment'
mergeFeaturesByPrevalence(x, ...)

getExperimentCrossAssociation(x, ...)

## S4 method for signature 'MultiAssayExperiment'
getExperimentCrossAssociation(x, ...)

## S4 method for signature 'SummarizedExperiment'
getExperimentCrossAssociation(x, ...)

## S4 method for signature 'TreeSummarizedExperiment'
mergeFeaturesByRank(x, ...)

testExperimentCrossAssociation(x, ...)

## S4 method for signature 'ANY'
testExperimentCrossAssociation(x, ...)

testExperimentCrossCorrelation(x, ...)

## S4 method for signature 'ANY'
testExperimentCrossCorrelation(x, ...)

getExperimentCrossCorrelation(x, ...)

## S4 method for signature 'ANY'
getExperimentCrossCorrelation(x, ...)

loadFromBiom(...)

loadFromQIIME2(...)

readQZA(...)

loadFromMothur(...)

loadFromMetaphlan(...)
```

```
loadFromHumann(...)

countDominantFeatures(x, ...)

## S4 method for signature 'SummarizedExperiment'
countDominantFeatures(x, ...)

subsetByRareTaxa(x, ...)

## S4 method for signature 'ANY'
subsetByRareTaxa(x, ...)

subsetByRareFeatures(x, ...)

## S4 method for signature 'ANY'
subsetByRareFeatures(x, ...)

subsetByPrevalentTaxa(x, ...)

## S4 method for signature 'ANY'
subsetByPrevalentTaxa(x, ...)

subsetByPrevalentFeatures(x, ...)

## S4 method for signature 'ANY'
subsetByPrevalentFeatures(x, ...)

countDominantTaxa(x, ...)

## S4 method for signature 'SummarizedExperiment'
countDominantTaxa(x, ...)

full_join(x, ...)

## S4 method for signature 'ANY'
full_join(x, ...)

inner_join(x, ...)

## S4 method for signature 'ANY'
inner_join(x, ...)

left_join(x, ...)

## S4 method for signature 'ANY'
left_join(x, ...)
```

```
right_join(x, ...)

## S4 method for signature 'ANY'
right_join(x, ...)

plotNMDS(x, ...)

estimateDivergence(x, ...)

## S4 method for signature 'SummarizedExperiment'
estimateDivergence(x, ...)

meltAssay(x, ...)

## S4 method for signature 'SummarizedExperiment'
meltAssay(x, ...)

transformSamples(x, ...)

## S4 method for signature 'SummarizedExperiment'
transformSamples(x, ...)

ZTransform(x, ...)

## S4 method for signature 'SummarizedExperiment'
ZTransform(x, ...)

relAbundanceCounts(x, ...)

## S4 method for signature 'SummarizedExperiment'
relAbundanceCounts(x, ...)

transformCounts(x, ...)

transformFeatures(x, ...)

## S4 method for signature 'SummarizedExperiment'
transformFeatures(x, ...)

getUniqueFeatures(x, ...)

## S4 method for signature 'SummarizedExperiment'
getUniqueFeatures(x, ...)

getUniqueTaxa(x, ...)

## S4 method for signature 'SummarizedExperiment'
getUniqueTaxa(x, ...)
```

```
getTopFeatures(x, ...)

## S4 method for signature 'SummarizedExperiment'
getTopFeatures(x, ...)

getTopTaxa(x, ...)

## S4 method for signature 'SummarizedExperiment'
getTopTaxa(x, ...)

getRareFeatures(x, ...)

## S4 method for signature 'ANY'
getRareFeatures(x, ...)

## S4 method for signature 'SummarizedExperiment'
getRareFeatures(x, ...)

getRareTaxa(x, ...)

## S4 method for signature 'ANY'
getRareTaxa(x, ...)

## S4 method for signature 'SummarizedExperiment'
getRareTaxa(x, ...)

getPrevalentFeatures(x, ...)

## S4 method for signature 'ANY'
getPrevalentFeatures(x, ...)

## S4 method for signature 'SummarizedExperiment'
getPrevalentFeatures(x, ...)

getPrevalentTaxa(x, ...)

## S4 method for signature 'ANY'
getPrevalentTaxa(x, ...)

## S4 method for signature 'SummarizedExperiment'
getPrevalentTaxa(x, ...)

subsampleCounts(x, ...)

## S4 method for signature 'SummarizedExperiment'
subsampleCounts(x, ...)
```

```
addPerSampleDominantFeatures(x, ...)

## S4 method for signature 'SummarizedExperiment'
addPerSampleDominantFeatures(x, ...)

addPerSampleDominantTaxa(x, ...)

## S4 method for signature 'SummarizedExperiment'
addPerSampleDominantTaxa(x, ...)

perSampleDominantFeatures(x, ...)

## S4 method for signature 'SummarizedExperiment'
perSampleDominantFeatures(x, ...)

perSampleDominantTaxa(x, ...)

## S4 method for signature 'SummarizedExperiment'
perSampleDominantTaxa(x, ...)

makePhyloseqFromTreeSE(x, ...)

## S4 method for signature 'SummarizedExperiment'
makePhyloseqFromTreeSE(x)

## S4 method for signature 'TreeSummarizedExperiment'
makePhyloseqFromTreeSE(x)

makePhyloseqFromTreeSummarizedExperiment(x)

## S4 method for signature 'ANY'
makePhyloseqFromTreeSummarizedExperiment(x)

makeTreeSummarizedExperimentFromPhyloseq(x)

## S4 method for signature 'ANY'
makeTreeSummarizedExperimentFromPhyloseq(x)

makeTreeSEFromBiom(...)

makeTreeSummarizedExperimentFromBiom(...)

makeTreeSEFromDADA2(...)

makeTreeSummarizedExperimentFromDADA2(...)

makeTreeSEFromPhyloseq(x)
```

```
estimateEvenness(x, ...)

## S4 method for signature 'ANY'
estimateEvenness(x, ...)

estimateRichness(x, ...)

## S4 method for signature 'ANY'
estimateRichness(x, ...)

estimateDiversity(x, ...)

## S4 method for signature 'ANY'
estimateDiversity(x, ...)

estimateFaith(x, ...)

## S4 method for signature 'ANY'
estimateFaith(x, ...)

estimateDominance(x, ...)

## S4 method for signature 'ANY'
estimateDominance(x, ...)

subsetSamples(x, ...)

subsetFeatures(x, ...)

subsetTaxa(x, ...)

## S4 method for signature 'SummarizedExperiment'
subsetSamples(x, ...)

## S4 method for signature 'SummarizedExperiment'
subsetFeatures(x, ...)

## S4 method for signature 'SummarizedExperiment'
subsetTaxa(x, ...)

relabundance(x, ...)

relabundance(x) <- value

## S4 method for signature 'SummarizedExperiment'
relabundance(x)

## S4 replacement method for signature 'SummarizedExperiment'
```

```
relabundance(x) <- value

runOverlap(x, ...)

## S4 method for signature 'SummarizedExperiment'
runOverlap(x, ...)

calculateOverlap(x, ...)

## S4 method for signature 'ANY'
calculateOverlap(x, ...)

calculateJSD(x, ...)

## S4 method for signature 'ANY'
calculateJSD(x, ...)

runJSD(x, ...)

## S4 method for signature 'SummarizedExperiment'
runJSD(x, ...)

calculateUnifrac(x, ...)

## S4 method for signature 'ANY'
calculateUnifrac(x, ...)

runUnifrac(mat, tree, ...)
```

Arguments

x	A SummarizedExperiment object.
...	Additional parameters. See dedicated function.
value	a matrix to store as the 'relabundance' assay
mat	An abundance matrix
tree	A phylogenetic tree

---

dmn_se *Twins' microbiome data from 278 individuals*

Description

dmn_se is a dataset on twins' microbiome where samples are stratified by their community composition through Dirichlet Multinomial Mixtures (DMM). It was derived from the **DirichletMultinomial** package.

Usage

```
data(dmn_se)
```

Format

A SummarizedExperiment with 130 features and 278 samples. The rowData contains no taxonomic information. The colData includes:

pheno participant's weight condition (Lean, Overwt and Obese)

Author(s)

Turnbaugh, PJ et al.

References

Holmes I, Harris K, Quince C (2012). Dirichlet Multinomial Mixtures: Generative Models for Microbial Metagenomics. PLoS ONE 7(2): e30126. https://doi.org/10.1371/journal.pone.0030126

Turnbaugh PJ, Hamady M, Yatsunenko T, Cantarel BL, Duncan A, et al. (2009). A core gut microbiome in obese and lean twins. Nature 457: 480–484. https://doi.org/10.1038/nature07540

See Also

mia-datasets calculateDMN

---

enterotype *Human gut microbiome dataset from 22 subjects based on shotgun DNA sequencing*

Description

The enterotype data of the human gut microbiome includes taxonomic profiling for 280 fecal samples from 22 subjects based on shotgun DNA sequencing. The authors claimed that the data naturally clumps into three community-level clusters, or "enterotypes", that are not immediately explained by sequencing technology or demographic features of the subjects. In a later addendum from 2014 the authors stated that enterotypes should not be seen as discrete clusters, but as a way of stratifying samples to reduce complexity. It was converted into a TreeSummarizedExperiment from the **phyloseq** package.

Usage

```
data(enterotype)
```

Format

A TreeSummarizedExperiment with 553 features and 280 samples. The rowData contains taxonomic information at Genus level. The colData includes:

Enterotype enterotype the sample belongs to (1, 2 and 3)

Sample_ID sample ID of samples from all studies

SeqTech sequencing technology

SampleID sample ID of complete samples

Project original project from which sample was obtained (gill06, turnbaugh09, MetaHIT, MicroObes, MicroAge and kurokawa07)

Nationality participant's nationality (american, danish, spanish, french, italian and japanese)

Gender participant's gender (F or M)

Age participant's age (0.25 – 87)

ClinicalStatus participant's clinical status (healthy, obese, CD, UC and elderly)

Author(s)

Arumugam, M., Raes, J., et al.

Source

http://www.bork.embl.de/Docu/Arumugam_et_al_2011/downloads.html

References

Arumugam, M., et al. (2011). Enterotypes of the human gut microbiome. Nature, 473(7346), 174-180. https://doi.org/10.1038/nature09944

Arumugam, M., et al. (2014). Addendum: Enterotypes of the human gut microbiome. Nature 506, 516 (2014). https://doi.org/10.1038/nature13075

See Also

mia-datasets

---

esophagus *Human esophageal community from 3 individuals*

Description

This small dataset from a human esophageal community includes 3 samples from 3 human adults based on biopsies analysed with 16S rDNA PCR. The 16S rRNA sequence processing is provided in the mothur wiki from the link below. It was converted into a TreeSummarizedExperiment from the **phyloseq** package.

Usage

```
data(esophagus)
```

Format

A TreeSummarizedExperiment with 58 features and 3 samples. The rowData contains no taxonomic information. The colData is empty.

Author(s)

Pei et al. <zhiheng.pei@med.nyu.edu>.

Source

http://www.mothur.org/wiki/Esophageal_community_analysis

References

Pei, Z., Bini, E. J., Yang, L., Zhou, M., Francois, F., & Blaser, M. J. (2004). Bacterial biota in the human distal esophagus. Proceedings of the National Academy of Sciences of the United States of America, 101(12), 4250-4255. https://doi.org/10.1073/pnas.0306398101

McMurdie, J. & Holmes, S. (2013) *phyloseq*: An R Package for reproducible interactive analysis and graphics of microbiome census data. PLoS ONE. 8(4):e61217. https://doi.org/10.1371/journal.pone.0061217

See Also

mia-datasets

---

getAbundant — *Determine abundant and rare taxa. Rare taxa can be further classified to conditionally rare and permanently rare.*

Description

These functions determine abundant and rare taxa based on the abundances of taxa. Compared to getPrevalent and getRare, these functions determine abundant and rare taxa based on abundance while the first mentioned are based on prevalence.

Usage

```
getAbundant(x, ...)

getLowAbundant(x, ...)

getConditionallyLowAbundant(x, ...)
```

```
getPermanentlyLowAbundant(x, ...)

getAbundanceClass(x, ...)

addAbundanceClass(x, ...)

## S4 method for signature 'SingleCellExperiment'
getAbundant(x, ...)

## S4 method for signature 'SummarizedExperiment'
getAbundant(x, assay.type = "relabundance", ...)

## S4 method for signature 'ANY'
getAbundant(x, abundant.th = 1/100, ...)

## S4 method for signature 'SingleCellExperiment'
getLowAbundant(x, ...)

## S4 method for signature 'SummarizedExperiment'
getLowAbundant(x, assay.type = "relabundance", abundant.th = 1/100, ...)

## S4 method for signature 'ANY'
getLowAbundant(x, abundant.th = 1/100, ...)

## S4 method for signature 'SingleCellExperiment'
getConditionallyLowAbundant(x, ...)

## S4 method for signature 'SummarizedExperiment'
getConditionallyLowAbundant(x, assay.type = "relabundance", ...)

## S4 method for signature 'ANY'
getConditionallyLowAbundant(x, abundant.th = 1/100, crt.th = 100, ...)

## S4 method for signature 'SingleCellExperiment'
getPermanentlyLowAbundant(x, ...)

## S4 method for signature 'SummarizedExperiment'
getPermanentlyLowAbundant(x, assay.type = "relabundance", ...)

## S4 method for signature 'ANY'
getPermanentlyLowAbundant(x, abundant.th = 1/100, prt.th = 5, ...)

## S4 method for signature 'SingleCellExperiment'
getAbundanceClass(x, ...)

## S4 method for signature 'SummarizedExperiment'
getAbundanceClass(x, assay.type = "relabundance", ...)
```

```
## S4 method for signature 'ANY'
getAbundanceClass(x, abundant.th = 1/100, crt.th = 100, prt.th = 5, ...)

## S4 method for signature 'SingleCellExperiment'
addAbundanceClass(x, ...)

## S4 method for signature 'SummarizedExperiment'
addAbundanceClass(x, name = "abundance_class", ...)
```

Arguments

x	a SummarizedExperiment object.
...	additional arguments.
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "relabundance")
abundant.th	Numeric scalar. Specifies threshold that is used to separate abundant features from rare. (Default: 1/100)
crt.th	Numeric scalar. Specifies threshold that is used to separate conditionally rare features from other rare features. (Default: 100)
prt.th	Numeric scalar. Specifies threshold that is used to separate permanently rare features from other rare features. (Default: 5)
name	Character scalar. Specifies name of column in rowData where the results will be stored. (Default: "abundance_class")

Details

These functions identify abundant and rare taxa in a dataset. Abundant taxa are characterized by high average abundance across the dataset, while rare taxa are characterized by consistently low abundance.

Conditionally rare taxa exhibit variable abundance, being abundant in some samples and rare in others. In contrast, permanently rare taxa consistently maintain low abundance across all samples.

- Abundant taxa: Taxa with an average abundance exceeding abundant.th.
- Low abundant / rare taxa: Taxa with an average abundance not exceeding abundant.th. Optionally, if specified, they must also satisfy the condition $crt.th >= \frac{abundance_{max}}{abundance_{min}} > prt.th$.
- Conditionally rare or low abundant taxa (CRT): Taxa with an average abundance not exceeding abundant.th and with a maximum-to-minimum abundance ratio ($\frac{abundance_{max}}{abundance_{min}}$) greater than crt.th.
- Permanently rare or low abundant taxa (PRT): Taxa with an average abundance not exceeding abundant.th and with a maximum-to-minimum abundance ratio ($\frac{abundance_{max}}{abundance_{min}}$) less than or equal to prt.th.

Value

For getAbundant, getLowAbundant, getConditionallyLowAbundant, and getPermanentlyLowAbundant a vector of taxa. For getAbudanceClass a vector of abundance classes for each feature. For addAbudanceClass, a SummarizedExperiment object.

References

Sizhong Y. et al. (2017) Community structure of rare methanogenic archaea: insight from a single functional group- *FEMS Microbiol. Ecol.* 93(11). https://doi.org/10.1093/femsec/fix126

See Also

getPrevalent and getRare

Examples

```
data(GlobalPatterns)
tse <- GlobalPatterns

# Agglomerate to family level
tse <- agglomerateByRank(tse, rank = "Family")
# Transform to relative abundances. Note that we add pseudocount. This is
# because otherwise we cannot calculate CRT and PRT due to zeroes and
# zero division in calculating abundance ratio.
tse <- transformAssay(tse, method = "relabundance", pseudocount = TRUE)

# Get abundant taxa
abundant <- getAbundant(tse, assay.type = "relabundance")
abundant |> head()

# Get all rare taxa that have average relative abundance below 10%
rare <- getLowAbundant(
    tse, assay.type = "relabundance", abundant.th = 10/100)
rare |> head()

# Get rare taxa that are not permanently or conditionally rare
rare <- getLowAbundant(
    tse, assay.type = "relabundance", prt.th = 5, crt.th = 100)
rare |> head()

# Get permanently rare taxa
prt <- getPermanentlyLowAbundant(
    tse, assay.type = "relabundance", prt.th = 5)
prt |> head()

# Get conditionally rare taxa
prt <- getConditionallyLowAbundant(
    tse, assay.type = "relabundance", crt.th = 100)
prt |> head()

# To classify all features, one can use *AbundantClass function
tse <- addAbundanceClass(tse)
# When one uses add* function, the results are stored to rowData
rowData(tse)
```

getCCA *Canonical Correspondence Analysis and Redundancy Analysis*

Description

These functions perform Canonical Correspondence Analysis on data stored in a SummarizedExperiment.

Usage

```
getCCA(x, ...)

addCCA(x, ...)

getRDA(x, ...)

addRDA(x, ...)

calculateCCA(x, ...)

runCCA(x, ...)

## S4 method for signature 'ANY'
getCCA(x, formula, data, ...)

## S4 method for signature 'SummarizedExperiment'
getCCA(
  x,
  formula = NULL,
  col.var = variables,
  variables = NULL,
  test.signif = TRUE,
  assay.type = assay_name,
  assay_name = exprs_values,
  exprs_values = "counts",
  ...
)

## S4 method for signature 'SingleCellExperiment'
addCCA(x, altexp = NULL, name = "CCA", ...)

calculateRDA(x, ...)

runRDA(x, ...)

## S4 method for signature 'ANY'
getRDA(x, formula, data, ...)
```

```
## S4 method for signature 'SummarizedExperiment'
getRDA(
  x,
  formula = NULL,
  col.var = variables,
  variables = NULL,
  test.signif = TRUE,
  assay.type = assay_name,
  assay_name = exprs_values,
  exprs_values = NULL,
  dis.name = NULL,
  ...
)

## S4 method for signature 'SingleCellExperiment'
addRDA(x, altexp = NULL, name = "RDA", ...)
```

Arguments

x	TreeSummarizedExperiment.
...	additional arguments passed to vegan::cca or vegan::dbrda and other internal functions. • method a dissimilarity measure to be applied in dbRDA and possible following homogeneity test. (Default: "euclidean") • scale: Logical scalar. Should the expression values be standardized? scale is disabled when using *RDA functions. Please scale before performing RDA. (Default: TRUE) • na.action: function. Action to take when missing values for any of the variables in formula are encountered. (Default: na.fail) • full Logical scalar. Should all the results from the significance calculations be returned. When FALSE, only summary tables are returned. (Default: FALSE) • homogeneity.test: Character scalar. Specifies the significance test used to analyse vegan::betadisper results. Options include 'permanova' (vegan::permutest), 'anova' (stats::anova) and 'tukeyhsd' (stats::TukeyHSD). (Default: "permanova") • permutations: Integer scalar. Specifies the number of permutations for significance testing in vegan::anova.cca. (Default: 999) • subset.result: Logical result. Specifies whether to subset x to match the result if some samples were removed during calculation. (Default: TRUE) • binary: Logical scalar. Whether to perform presence/absence transformation before dissimilarity calculation. For Jaccard index the default is TRUE. For other dissimilarity metrics, please see vegdist.
formula	formula. If x is a SummarizedExperiment a formula can be supplied. Based on the right-hand side of the given formula colData is subset to col.var.

	col.var and formula can be missing, which turns the CCA analysis into a CA analysis and dbRDA into PCoA/MDS.
data	data.frame or coarcible to one. The covariance table including covariates defined by formula.
col.var	Character scalar. When x is a SummarizedExperiment,col.var can be used to specify variables from colData.
variables	Deprecated. Use col.var instead.
test.signif	Logical scalar. Should the PERMANOVA and analysis of multivariate homogeneity of group dispersions be performed. (Default: TRUE)
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: NULL)
assay_name	Deprecated. Use assay.type instead.
exprs_values	Deprecated. Use assay.type instead.
altexp	Character scalar or integer scalar. Specifies an alternative experiment containing the input data.
name	Character scalar. A name for the reducedDim() where results will be stored. (Default: "CCA")
dis.name	Character scalar. Specifies the name of dissimilarity matrix from metadata slot used in calculation. (Default: NULL)

Details

*CCA functions utilize vegan:cca and *RDA functions vegan:dbRDA. By default, dbRDA is done with euclidean distances, which is equivalent to RDA. col.var and formula can be missing, which turns the CCA analysis into a CA analysis and dbRDA into PCoA/MDS.

Significance tests are done with vegan:anova.cca (PERMANOVA). Group dispersion, i.e., homogeneity within groups is analyzed with vegan::betadisper (multivariate homogeneity of groups dispersions (variances)) and statistical significance of homogeneity is tested with a test specified by homogeneity.test parameter.

Value

For getCCA a matrix with samples as rows and CCA dimensions as columns. Attributes include output from scores, eigenvalues, the cca/rda object and significance analysis results.

For addCCA a modified x with the results stored in reducedDim as the given name.

See Also

For more details on the actual implementation see cca and dbrda

Examples

```
library(miaViz)
data("enterotype", package = "mia")
tse <- enterotype
```

```
# Perform CCA and exclude any sample with missing ClinicalStatus
tse <- addCCA(
    tse,
    assay.type = "counts",
    formula = data ~ ClinicalStatus,
    na.action = na.exclude
    )

# Plot CCA
plotCCA(tse, "CCA", colour_by = "ClinicalStatus")

# Fetch significance results
getReducedDimAttribute(tse, dimred = "CCA", name = "significance")

tse <- transformAssay(tse, method = "relabundance")

# Specify dissimilarity measure
tse <- addRDA(
    tse,
    formula = data ~ ClinicalStatus,
    assay.type = "relabundance",
    method = "bray",
    name = "RDA_bray",
    na.action = na.exclude
    )

# To scale values when using *RDA functions, use
# transformAssay(MARGIN = "features", ...)
tse <- transformAssay(tse, method = "standardize", MARGIN = "features")

# Data might include taxa that do not vary. Remove those because after
# z-transform their value is NA
tse <- tse[rowSums(is.na(assay(tse, "standardize"))) == 0, ]

# Calculate RDA
tse <- addRDA(
   tse,
   formula = data ~ ClinicalStatus,
   assay.type = "standardize",
   name = "rda_scaled",
   na.action = na.omit
   )

# Plot RDA
plotRDA(tse, "rda_scaled", colour_by = "ClinicalStatus")

# A common choice along with PERMANOVA is ANOVA when statistical significance
# of homogeneity of groups is analysed. Moreover, full significance test
# results can be returned.
tse <- transformAssay(tse, method = "clr", pseudocount = 1)
tse <- addRDA(
    tse,
    assay.type = "clr",
```

```
    formula = data ~ ClinicalStatus,
    homogeneity.test = "anova",
    full = TRUE
    )

# Example showing how to pass extra parameters, such as 'permutations',
# to anova.cca
tse <- addRDA(
    tse,
    assay.type = "clr",
    formula = data ~ ClinicalStatus,
    permutations = 500
    )

# In dbRDA, dissimilarity matrix is calculated internally which is
# computationally heavy operation. If you have large number of samples, and
# you want to fit multiple dbRDA models, you might want to consider
# pre-calculation of dissimilarity matrix as the same matrix will be used for
# all models. You can then run dbRDA with the pre-calculated dissimilarity
# avoiding redundant dissimilarity calculations.
tse <- addDissimilarity(tse, assay.type = "relabundance", method = "bray")
tse <- addRDA(
    tse,
    formula = data ~ ClinicalStatus,
    dis.name = "bray",
    name = "RDA_precalc_bray",
    na.action = na.exclude
    )
```

`getCluster` *Clustering wrapper*

Description

This function returns a `SummarizedExperiment` with clustering information in its colData or rowData

Usage

```
getCluster(x, ...)

addCluster(x, ...)

## S4 method for signature 'SummarizedExperiment'
addCluster(
  x,
  BLUSPARAM,
  assay.type = assay_name,
```

```
  assay_name = NULL,
  by = MARGIN,
  MARGIN = "rows",
  name = "clustering",
  clust.col = "cluster",
  full = FALSE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
addCluster(
  x,
  BLUSPARAM,
  assay.type = assay_name,
  assay_name = NULL,
  dimred = NULL,
  by = MARGIN,
  MARGIN = "rows",
  name = "clustering",
  clust.col = "cluster",
  full = FALSE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
getCluster(
  x,
  BLUSPARAM,
  assay.type = assay_name,
  assay_name = NULL,
  by = MARGIN,
  MARGIN = "rows",
  ...
)

## S4 method for signature 'SingleCellExperiment'
getCluster(
  x,
  BLUSPARAM,
  assay.type = assay_name,
  assay_name = NULL,
  dimred = NULL,
  by = MARGIN,
  MARGIN = "rows",
  ...
)

## S4 method for signature 'ANY'
```

```
getCluster(x, BLUSPARAM, by = MARGIN, MARGIN = "rows", full = FALSE, ...)
```

Arguments

x	A SummarizedExperiment object.
...	Additional parameters to use altExps for example
BLUSPARAM	A BlusterParam object specifying the algorithm to use.
assay.type	Character scalar. Specifies the name of assay used in calculation.
assay_name	Deprecated. Use assay.type instead.
by	Character scalar. Determines if association is calculated row-wise / for features ('rows') or column-wise / for samples ('cols'). Must be 'rows' or 'cols'.
MARGIN	Deprecated. Use by instead.
name	Character scalar. The name to store the result in metadata
clust.col	Character scalar. Indicates the name of the rowData (or colData) where the data will be stored. (Default: "clusters")
full	Logical scalar indicating whether the full clustering statistics should be returned for each method.
dimred	Character scalar or integer scalar. Specifies dimension reduction results used in calculation. Either dimred or assay.type must be specified.

Details

This is a wrapper for the clusterRows function from the bluster package.

When setting full = TRUE, the clustering information will be stored in the metadata of the object.

By default, clustering is done on the features.

Value

addCluster returns an object of the same type as the x parameter with clustering information named clusters stored in colData or rowData.

Examples

```
library(bluster)
data(GlobalPatterns, package = "mia")
tse <- GlobalPatterns

# Cluster on rows using Kmeans
tse <- addCluster(
    tse,
    assay.type = "counts",
    BLUSPARAM = KmeansParam(centers = 3)
)

# Clustering done on the samples using Hclust
res <- getCluster(
    tse,
```

```
    assay.type = "counts",
    by = "samples",
    BLUSPARAM = HclustParam(metric = "bray", dist.fun = vegan::vegdist)
)
res |> head()

# Apply clustering to PCA results
library(scater)
tse <- transformAssay(tse, method = "rclr")
tse <- runPCA(tse, assay.type = "rclr")
tse <- addCluster(
    tse,
    dimred = "PCA",
    BLUSPARAM = KmeansParam(centers = 3),
    by = 2
)
tse$cluster |> head()
```

`getCrossAssociation` *Calculate correlations between features of two experiments.*

Description

Calculate correlations between features of two experiments.

Usage

```
getCrossAssociation(x, ...)

## S4 method for signature 'MultiAssayExperiment'
getCrossAssociation(
  x,
  experiment1 = 1,
  experiment2 = 2,
  assay.type1 = assay_name1,
  assay_name1 = NULL,
  assay.type2 = assay_name2,
  assay_name2 = NULL,
  altexp1 = NULL,
  altexp2 = NULL,
  col.var1 = colData_variable1,
  colData_variable1 = NULL,
  col.var2 = colData_variable2,
  colData_variable2 = NULL,
  by = MARGIN,
  MARGIN = 1,
  method = "kendall",
```

```
  mode = "table",
  p.adj.method = p_adj_method,
 p_adj_method = c("fdr", "BH", "bonferroni", "BY", "hochberg", "holm", "hommel", "none"),
  p.adj.threshold = p_adj_threshold,
  p_adj_threshold = NULL,
  cor.threshold = cor_threshold,
  cor_threshold = NULL,
  sort = FALSE,
  filter.self.cor = filter_self_correlations,
  filter_self_correlations = FALSE,
  verbose = TRUE,
  test.signif = test_significance,
  test_significance = FALSE,
  show.warnings = show_warnings,
  show_warnings = TRUE,
  paired = FALSE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
getCrossAssociation(x, experiment2 = x, ...)
```

Arguments

x	A MultiAssayExperiment or SummarizedExperiment object.
...	Additional arguments: • symmetric: Logical scalar. Specifies if measure is symmetric or not. When symmetric = TRUE, associations are calculated only for unique variable-pairs, and they are assigned to corresponding variable-pair. This decreases the number of calculations in 2-fold meaning faster execution. (By default: FALSE) • association.fun: A function that is used to calculate (dis-)similarity between features. Function must take matrix as an input and give numeric values as an output. Adjust method and other parameters correspondingly. Supported functions are, for example, stats::dist and vegan::vegdist. • dimred1 Character scalar or numeric scalar. Specifies reduced dimensionality from the reducedDim of experiment 1. (Default: NULL) • dimred2 Character scalar or numeric scalar. Specifies reduced dimensionality from the reducedDim of experiment 2. (Default: NULL)
experiment1	Character scalar or numeric scalar. Selects the experiment 1 from experiments(x) of MultiassayExperiment object. (Default: 1)
experiment2	Character scalar or numeric scalar. Selects the experiment 2 fromexperiments(x) of MultiAssayExperiment object or altExp(x) of TreeSummarizedExperiment object. Alternatively, experiment2 can also be TreeSE object when x is TreeSE object. (Default: 2 when x is MAE and x when x is TreeSE)

assay.type1 Character scalar. Specifies the name of the assay in experiment 1 to be transformed. (Default: NULL)

assay_name1 Deprecated. Use assay.type1 instead.

assay.type2 Character scalar. Specifies the name of the assay in experiment 2 to be transformed. (Default: NULL)

assay_name2 Deprecated. Use assay.type2 instead.

altexp1 Character scalar or numeric scalar. Specifies alternative experiment from the altExp of experiment 1. If NULL, then the experiment is itself and altExp option is disabled. (Default: NULL)

altexp2 Character scalar or numeric scalar. Specifies alternative experiment from the altExp of experiment 2. If NULL, then the experiment is itself and altExp option is disabled. (Default: NULL)

col.var1 Character scalar. Specifies column(s) from colData of experiment 1. If col.var1 is used, assay.type1 is disabled. (Default: NULL)

colData_variable1
Deprecated. Use col.var1 instead.

col.var2 Character scalar. Specifies column(s) from colData of experiment 2. If col.var2 is used, assay.type2 is disabled. (Default: NULL)

colData_variable2
Deprecated. Use col.var2 instead.

by ACharacter scalar. Determines if association are calculated row-wise / for features ('rows') or column-wise / for samples ('cols'). Must be 'rows' or 'cols'.

MARGIN Deprecated. Use by instead.

method Character scalar. Defines the association method ('kendall', 'pearson', or 'spearman' for continuous/numeric; 'categorical' for discrete) (Default: "kendall")

mode Character scalar. Specifies the output format Available formats are 'table' and 'matrix'. (Default: "table")

p.adj.method Character scalar. Specifies adjustment method of p-values. Passed to p.adjust function. (Default: "fdr")

p_adj_method Deprecated. Use p.adj.method instead.

p.adj.threshold
Numeric scalar. From 0 to 1, specifies adjusted p-value threshold for filtering. (Default: NULL)

p_adj_threshold
Deprecated. Use p.dj.threshold instead.

cor.threshold Numeric scalar. From 0 to 1, specifies correlation threshold for filtering. (Default: NULL)

cor_threshold Deprecated. Use cor.threshold instead.

sort Logical scalar. Specifies whether to sort features or not in result matrices. Used method is hierarchical clustering. (Default: FALSE)

filter.self.cor
: Logical scalar. Specifies whether to filter out correlations between identical items. Applies only when correlation between experiment itself is tested, i.e., when assays are identical. (Default: FALSE)

filter_self_correlations
: Deprecated. Use filter.self.cor instead.

verbose
: Logical scalar. Specifies whether to get messages about progress of calculation. (Default: FALSE)

test.signif
: Logical scalar. Specifies whether to test statistical significance of associations. (Default: FALSE)

test_significance
: Deprecated. Use test.signif instead.

show.warnings
: Logical scalar. specifies whether to show warnings that might occur when correlations and p-values are calculated. (Default: FALSE)

show_warnings
: Deprecated. use show.warnings instead.

paired
: Logical scalar. Specifies if samples are paired or not. colnames must match between twp experiments. paired is disabled when by = 1. (Default: FALSE)

Details

The function getCrossAssociation calculates associations between features of two experiments. By default, it not only computes associations but also tests their significance. If desired, setting test.signif to FALSE disables significance calculation.

We recommend the non-parametric Kendall's tau as the default method for association analysis. Kendall's tau has desirable statistical properties and robustness at lower sample sizes. Spearman rank correlation can provide faster solutions when running times are critical.

Value

This function returns associations in table or matrix format. In table format, returned value is a data frame that includes features and associations (and p-values) in columns. In matrix format, returned value is a one matrix when only associations are calculated. If also significances are tested, then returned value is a list of matrices.

Examples

```
data(HintikkaXOData)
mae <- HintikkaXOData

# Subset so that less observations / quicker to run, just for example
mae[[1]] <- mae[[1]][1:20, 1:10]
mae[[2]] <- mae[[2]][1:20, 1:10]
# Several rows in the counts assay have a standard deviation of zero
# Remove them, since they do not add useful information about
# cross-association
mae[[1]] <- mae[[1]][rowSds(assay(mae[[1]])) > 0, ]
# Transform data
mae[[1]] <- transformAssay(mae[[1]], method = "rclr")
```

```
# Calculate cross-correlations
result <- getCrossAssociation(
    mae, method = "pearson", assay.type1 = "counts", assay.type2 = "nmr",
    show.warnings = FALSE, verbose = FALSE)
# Show first 5 entries
head(result, 5)

# Use altExp option to specify alternative experiment from the experiment
altExp(mae[[1]], "Phylum") <- agglomerateByRank(mae[[1]], rank = "Phylum")
# Transform data
altExp(mae[[1]], "Phylum") <- transformAssay(
    altExp(mae[[1]], "Phylum"), method = "relabundance")
# When mode = "matrix", the return value is a matrix
result <- getCrossAssociation(
    mae, experiment2 = 2, assay.type1 = "relabundance", assay.type2 = "nmr",
    altexp1 = "Phylum", method = "pearson", mode = "matrix",
    show.warnings = FALSE, verbose = FALSE)
# Show first 5 entries
head(result, 5)

# If test.signif = TRUE, then getCrossAssociation additionally returns
# significances
# filter.self.cor = TRUE filters self correlations
# p.adj.threshold can be used to filter those features that do not
# have any correlations whose p-value is lower than the threshold
result <- getCrossAssociation(
    mae[[1]], experiment2 = mae[[1]], method = "pearson",
    assay.type1 = "counts", assay.type2 = "counts",
    filter.self.cor = TRUE, p.adj.threshold = 0.05, test.signif = TRUE,
    show.warnings = FALSE, verbose = FALSE)
# Show first 5 entries
head(result, 5)

# Returned value is a list of matrices
names(result)

# Calculate Bray-Curtis dissimilarity between samples. If dataset includes
# paired samples, you can use paired = TRUE.
result <- getCrossAssociation(
    mae[[1]], mae[[1]], by = 2, paired = FALSE,
    assay.type1 = "counts", assay.type2 = "counts",
    association.fun = getDissimilarity, method = "bray",
    show.warnings = FALSE, verbose = FALSE)

# If experiments are equal and measure is symmetric
# (e.g., taxa1 vs taxa2 == taxa2 vs taxa1),
# it is possible to speed-up calculations by calculating association only
# for unique variable-pairs. Use "symmetric" to choose whether to measure
# association for only other half of of variable-pairs.
result <- getCrossAssociation(
    mae, experiment1 = "microbiota", experiment2 = "microbiota",
    assay.type1 = "counts", assay.type2 = "counts", symmetric = TRUE,
```

```
    show.warnings = FALSE, verbose = FALSE)

# For big data sets, the calculations might take a long time.
# To speed them up, you can take a random sample from the data.
# When dealing with complex biological problems, random samples can be
# enough to describe the data. Here, our random sample is 30 % of whole data.
sample_size <- 0.3
tse <- mae[[1]]
tse_sub <- tse[ sample( seq_len( nrow(tse) ), sample_size * nrow(tse) ), ]
result <- getCrossAssociation(
    tse_sub, assay.type1 = "counts", assay.type2 = "counts",
    show.warnings = FALSE, verbose = FALSE)

# It is also possible to choose variables from colData and calculate
# association between assay and sample metadata or between variables of
# sample metadata
mae[[1]] <- addAlpha(mae[[1]])
# col.var works similarly to assay.type. Instead of fetching
# an assay named assay.type from assay slot, it fetches a column named
# col.var from colData.
result <- getCrossAssociation(
    mae[[1]], assay.type1 = "counts",
    col.var2 = c("shannon_diversity", "dbp_dominance"),
    test.signif = TRUE, show.warnings = FALSE, verbose = FALSE)

# If your data contains TreeSE with alternative experiment in altExp,
# correlations can be calculated as follows.

# Create TreeSE with altExp
tse <- mae[[1]]
altExp(tse, "metabolites") <- mae[[2]]
# Calculate
res <- getCrossAssociation(
    tse,
    altexp2 = "metabolites",
    assay.type1 = "rclr",
    assay.type2 = "nmr",
    show.warnings = FALSE, verbose = FALSE
)

# To calculate correlation of features to principal coordinates, you have to
# first calculate PCoA...
library(scater)
tse <- addMDS(tse, assay.type = "rclr", method = "euclidean")
# ...then calculate the correlation.
res <- getCrossAssociation(tse, assay.type1 = "rclr", dimred2 = "MDS",
    show.warnings = FALSE, verbose = FALSE)
head(res)

# In ecological studies the association analyses are sometimes carried out
# only for those samples that have observations. You can ignore zeroes by
# replacing them with NA.
mat <- assay(tse, "counts")
```

```
mat[ mat == 0 ] <- NA
assay(tse, "non_zero", withDimnames = FALSE) <- mat
res2 <- getCrossAssociation(
    tse, tse,
    assay.type1 = "non_zero", assay.type2 = "non_zero"
)
```

---

getDominant *Get dominant taxa*

Description

These functions return information about the most dominant taxa in a SummarizedExperiment object.

Usage

```
getDominant(
  x,
  assay.type = assay_name,
  assay_name = "counts",
  group = rank,
  rank = NULL,
  other.name = "Other",
  n = NULL,
  complete = TRUE,
  ...
)

addDominant(x, name = "dominant_taxa", other.name = "Other", n = NULL, ...)

## S4 method for signature 'SummarizedExperiment'
getDominant(
  x,
  assay.type = assay_name,
  assay_name = "counts",
  group = rank,
  rank = NULL,
  other.name = "Other",
  n = NULL,
  complete = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
addDominant(
```

```
  x,
  name = "dominant_taxa",
  other.name = "Other",
  n = NULL,
  complete = FALSE,
  ...
)
```

Arguments

x	TreeSummarizedExperiment.
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
group	Character scalar. Defines a group. Must be one of the columns from rowData(x). (Default: NULL)
rank	Deprecated. Use group instead.
other.name	Character scalar. A name for features that are not included in n the most frequent dominant features in the data. (Default: "Other")
n	Numeric scalar. The number of features that are the most frequent dominant features. Default is NULL, which defaults that each sample is assigned a dominant taxon. (Default: NULL)
complete	Logical scalar. A value to manage multiple dominant taxa for a sample. Default for getDominant is TRUE to include all equally dominant taxa for each sample. complete = FALSE samples one taxa for the samples that have multiple. Default for addDominant is FALSE to add a column with only one dominant taxon assigned for each sample into colData. complete = TRUE adds a list that includes all dominant taxa for each sample into colData.
...	Additional arguments passed on to agglomerateByRank() when rank is specified.
name	Character scalar. A name for the column of the colData where results will be stored. (Default: "dominant_taxa")

Details

addDominant extracts the most abundant taxa in a SummarizedExperiment object, and stores the information in the colData. It is a wrapper for getDominant.

With group parameter, it is possible to agglomerate rows based on groups. If the value is one of the columns in taxonomyRanks(), agglomerateByRank() is applied. Otherwise, agglomerateByVariable() is utilized. E.g. if 'Genus' rank is used, all abundances of same Genus are added together, and agglomerated features are returned. See corresponding functions for additional arguments to deal with missing values or special characters.

Value

getDominant returns a named character vector x while addDominant returns SummarizedExperiment with additional column in colData named *name*.

Examples

```
data(GlobalPatterns)
x <- GlobalPatterns

# Finds the dominant taxa.
sim.dom <- getDominant(x, group = "Genus")

# Add information to colData
x <- addDominant(x, group = "Genus", name ="dominant_genera")
colData(x)
```

---

getDPCoA *Calculation of Double Principal Correspondence analysis*

Description

Double Principal Correspondence analysis is made available via the ade4 package in typical fashion. Results are stored in the reducedDims and are available for all the expected functions.

Usage

```
getDPCoA(x, y, ...)

## S4 method for signature 'ANY,ANY'
getDPCoA(
  x,
  y,
  ncomponents = 2,
  ntop = NULL,
  subset.row = subset_row,
  subset_row = NULL,
  scale = FALSE,
  transposed = FALSE,
  ...
)

## S4 method for signature 'TreeSummarizedExperiment,missing'
getDPCoA(
  x,
  ...,
  assay.type = assay_name,
  assay_name = exprs_values,
  exprs_values = "counts",
  tree.name = tree_name,
  tree_name = "phylo"
)
```

```
calculateDPCoA(x, ...)

addDPCoA(x, ..., altexp = NULL, name = "DPCoA")

runDPCoA(x, ...)
```

Arguments

x	TreeSummarizedExperiment.
y	a dist or a symmetric matrix compatible with ade4:dpcoa
...	Currently not used.
ncomponents	Numeric scalar. Indicates the number of DPCoA dimensions to obtain. (Default: 2)
ntop	Numeric scalar. Specifies the number of features with the highest variances to use for dimensionality reduction. Alternatively NULL, if all features should be used. (Default: NULL)
subset.row	Character Vector. Specifies the subset of features to use for dimensionality reduction. This can be a character vector of row names, an integer vector of row indices or a logical vector. (Default: NULL)
subset_row	Deprecated. Use subset.row instead.
scale	Logical scalar. Should the expression values be standardized? (Default: FALSE)
transposed	Logical scalar. Specifies if x is transposed with cells in rows. (Default: FALSE)
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
exprs_values	Deprecated. Use assay.type instead.
tree.name	Character scalar. Specifies the name of the tree to be included in the phyloseq object that is created, (Default: "phylo")
tree_name	Deprecated. Use tree.name instead.
altexp	Character scalar or integer scalar. Specifies an alternative experiment containing the input data. (Default: NULL)
name	Character scalar. A name for the column of the colData where results will be stored. (Default: "DPCoA")

Details

For addDPCoA a TreeSummarizedExperiment containing the expression values as well as a rowTree to calculate y using cophenetic.phylo.

In addition to the reduced dimension on the features, the reduced dimension for samples are returned as well as sample_red attribute. eig, feature_weights and sample_weights are returned as attributes as well.

Value

For `getDPCoA` a matrix with samples as rows and CCA dimensions as columns

For `addDPCoA` a modified `x` with the results stored in `reducedDim` as the given `name`

See Also

`plotReducedDim` `reducedDims`

Examples

```
data(esophagus)
dpcoa <- getDPCoA(esophagus)
head(dpcoa)

esophagus <- addDPCoA(esophagus)
reducedDims(esophagus)

library(scater)
plotReducedDim(esophagus, "DPCoA")
```

---

getHierarchyTree *Calculate hierarchy tree*

Description

These functions generate a hierarchy tree using taxonomic information from a `SummarizedExperiment` object and add this hierarchy tree into the `rowTree`.

Usage

```
getHierarchyTree(x, ...)

addHierarchyTree(x, ...)

## S4 method for signature 'SummarizedExperiment'
getHierarchyTree(x, ...)

## S4 method for signature 'SummarizedExperiment'
addHierarchyTree(x, ...)
```

Arguments

`x`	`TreeSummarizedExperiment`.
`...`	optional arguments not used currently.

Details

addHierarchyTree calculates a hierarchy tree from the available taxonomic information and add it to rowTree.

getHierarchyTree generates a hierarchy tree from the available taxonomic information. Internally it uses toTree and resolveLoop to sanitize data if needed.

Please note that a hierarchy tree is not an actual phylogenetic tree. A phylogenetic tree represents evolutionary relationships among features. On the other hand, a hierarchy tree organizes species into a hierarchical structure based on their taxonomic ranks.

Value

- addHierarchyTree: a TreeSummarizedExperiment whose phylo tree represents the hierarchy among available taxonomy information.
- getHierarchyTree: a phylo tree representing the hierarchy among available taxonomy information.

Examples

```
# Generate a tree based on taxonomic rank hierarchy (a hierarchy tree).
data(GlobalPatterns)
tse <- GlobalPatterns
getHierarchyTree(tse)

# Add a hierarchy tree to a TreeSummarizedExperiment.
# Please note that any tree already stored in rowTree() will be overwritten.
tse <- addHierarchyTree(tse)
tse
```

---

getLDA *Latent Dirichlet Allocation*

Description

These functions perform Latent Dirichlet Allocation on data stored in a TreeSummarizedExperiment object.

Usage

```
getLDA(x, ...)

addLDA(x, ...)

## S4 method for signature 'SummarizedExperiment'
getLDA(x, k = 2, assay.type = "counts", eval.metric = "perplexity", ...)

## S4 method for signature 'SummarizedExperiment'
addLDA(x, k = 2, assay.type = "counts", name = "LDA", ...)
```

Arguments

x	a TreeSummarizedExperiment object.
...	optional arguments passed to LDA
k	Integer vector. A number of latent vectors/topics. (Default: 2)
assay.type	Character scalar. Specifies which assay to use for LDA ordination. (Default: "counts")
eval.metric	Character scalar. Specifies evaluation metric that will be used to select the model with the best fit. Must be either "perplexity" (topicmodels::perplexity) or "coherence" (topicdoc::topic_coherence, the best model is selected based on mean coherence). (Default: "perplexity")
name	Character scalar. The name to be used to store the result in the reducedDims of the output. (Default: "LDA")

Details

The functions getLDA and addLDA internally use LDA to compute the ordination matrix and feature loadings.

Value

For getLDA, the ordination matrix with feature loadings matrix as attribute "loadings".

For addLDA, a TreeSummarizedExperiment object is returned containing the ordination matrix in reducedDim(..., name) with feature loadings matrix as attribute "loadings".

Examples

```
data(GlobalPatterns)
tse <- GlobalPatterns

# Reduce the number of features
tse <- agglomerateByPrevalence(tse, rank="Phylum")

# Run LDA and add the result to reducedDim(tse, "LDA")
tse <- addLDA(tse)

# Extract feature loadings
loadings <- getReducedDimAttribute(tse, "LDA", "loadings")
head(loadings)

# Estimate models with number of topics from 2 to 10
tse <- addLDA(tse, k = c(2, 3, 4, 5, 6, 7, 8, 9, 10), name = "LDA_10")
# Get the evaluation metrics
tab <- getReducedDimAttribute(tse, "LDA_10","eval_metrics")
# Plot
plot(tab[["k"]], tab[["perplexity"]], xlab = "k", ylab = "perplexity")
```

getMDS *Perform multi-dimensional scaling (MDS)*

Description

Perform multi-dimensional scaling (MDS) also know as Principal Coordinate Analysis (PCoA). These functions are wrappers for scater::calculateMDS.

Usage

```
getMDS(x, ...)

addMDS(x, ...)

## S4 method for signature 'SingleCellExperiment'
addMDS(x, name = "MDS", ...)

## S4 method for signature 'SingleCellExperiment'
getMDS(x, assay.type = "counts", ...)

## S4 method for signature 'TreeSummarizedExperiment'
getMDS(x, assay.type = "counts", ...)
```

Arguments

x	a SummarizedExperiment object.
...	additional arguments. • FUN: Function. A function that is applied to calculate dissimilarity. (Default: getDissimilarity) \item \code{subset.result}: \code{Logical result}. Specifies whether to subset \code{x} to match the result if some samples were removed during calculation. (Default: \code{TRUE})
name	Character scalar. A name for the reducedDim() where results will be stored. (Default: "MDS")
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")

Details

These functions are wrappers for scater::calculateMDS and scater::runMDS. While getMDS returns the results, addMDS adds them to reducedDim(x). The difference is that these functions apply microbiome-specific options such as getDissimilarity function by default.

See scater::calculateMDS for details.

Value

getMDS returns a MDS results. addMDS returns a x with MDS results added to its reducedDim(x, name).

See Also

scater::calculateMDS and getDissimilarity

Examples

```
library(mia)
library(scater)
library(patchwork)

data(GlobalPatterns)
tse <- GlobalPatterns

# Calculate PCoA with Bray-Curtis dissimilarity
tse <- transformAssay(tse, method = "relabundance")
tse <- addMDS(tse, assay.type = "relabundance", method = "bray")

# Calculate PCoA with Unifrac and rarefaction. (Note: increase iterations)
tse <- addMDS(tse, method = "unifrac", name = "unifrac")

# Calculate PCoA with Unifrac and rarefaction. (Note: increase iterations)
tse <- addMDS(tse, method = "unifrac", name = "unifrac_rare", niter = 2L)

# Visualize results
p1 <- plotReducedDim(tse, "unifrac", colour_by = "SampleType") +
    labs(title = "Not rarefied")
p2 <- plotReducedDim(tse, "unifrac_rare", colour_by = "SampleType") +
    labs(title = "Rarefied")
p1 + p2
```

---

getNMDS *Perform non-metric MDS on sample-level data*

Description

Perform non-metric multi-dimensional scaling (nMDS) on samples, based on the data in a SingleCellExperiment object.

Usage

```
getNMDS(x, ...)

## S4 method for signature 'ANY'
getNMDS(
```

```
  x,
  FUN = vegdist,
  nmds.fun = nmdsFUN,
  nmdsFUN = c("isoMDS", "monoMDS"),
  ncomponents = 2,
  ntop = 500,
  subset.row = subset_row,
  subset_row = NULL,
  scale = FALSE,
  transposed = FALSE,
  keep.dist = keep_dist,
  keep_dist = FALSE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
getNMDS(
  x,
  ...,
  assay.type = assay_name,
  assay_name = exprs_values,
  exprs_values = "counts",
  FUN = vegdist
)

## S4 method for signature 'SingleCellExperiment'
getNMDS(
  x,
  ...,
  assay.type = assay_name,
  assay_name = exprs_values,
  exprs_values = "counts",
  dimred = NULL,
  ndimred = n_dimred,
  n_dimred = NULL,
  FUN = vegdist
)

calculateNMDS(x, ...)

addNMDS(x, ..., altexp = NULL, name = "NMDS")

runNMDS(x, ...)
```

Arguments

x	TreeSummarizedExperiment.
...	additional arguments to pass to FUN and nmds.fun.

FUN	Function or Character scalar. A value with a function name returning a dist object
nmds.fun	Character scalar. A value to choose the scaling implementation, either "isoMDS" for MASS::isoMDS or "monoMDS" for vegan::monoMDS
nmdsFUN	Deprecated. Use nmds.fun instead.
ncomponents	Numeric scalar. Indicates the number of DPCoA dimensions to obtain. (Default: 2)
ntop	Numeric scalar. Specifies the number of features with the highest variances to use for dimensionality reduction. Alternatively NULL, if all features should be used. (Default: NULL)
subset.row	Character Vector. Specifies the subset of features to use for dimensionality reduction. This can be a character vector of row names, an integer vector of row indices or a logical vector. (Default: NULL)
subset_row	Deprecated. Use subset.row instead.
scale	Logical scalar. Should the expression values be standardized? (Default: FALSE)
transposed	Logical scalar. Specifies if x is transposed with cells in rows. (Default: FALSE)
keep.dist	Logical scalar. Indicates whether the dist object calculated by FUN should be stored as 'dist' attribute of the matrix returned/stored by getNMDS/ addNMDS. (Default: FALSE)
keep_dist	Deprecated. Use keep.dist instead.
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
exprs_values	Deprecated. Use assay.type instead.
dimred	Character scalar or integer scalar. Specifies the existing dimensionality reduction results to use.
ndimred	integer vector. Specifies the dimensions to use if dimred is specified.
n_dimred	Deprecated. Use ndimred instead.
altexp	Character scalar or integer scalar. Specifies an alternative experiment containing the input data. (Default: NULL)
name	Character scalar. A name for the column of the colData where results will be stored. (Default: "NMDS")

Details

For addNMDS a TreeSummarizedExperiment

Either MASS::isoMDS or vegan::monoMDS are used internally to compute the NMDS components. If you supply a custom FUN, make sure that the arguments of FUN and nmds.fun do not collide.

Value

For getNMDS, a matrix is returned containing the MDS coordinates for each sample (row) and dimension (column).

See Also

MASS::isoMDS, vegan::monoMDS for NMDS component calculation.

plotMDS, to quickly visualize the results.

Examples

```
# generate some example data
mat <- matrix(1:60, nrow = 6)
df <- DataFrame(n = c(1:6))
tse <- TreeSummarizedExperiment(assays = list(counts = mat),
                                rowData = df)
#
getNMDS(tse)

#
data(esophagus)
esophagus <- addNMDS(esophagus, FUN = vegan::vegdist, name = "BC")
esophagus <- addNMDS(esophagus, FUN = vegan::vegdist, name = "euclidean",
                     method = "euclidean")
reducedDims(esophagus)
```

---

getNMF *Non-negative Matrix Factorization*

Description

These functions perform Non-negative Matrix Factorization on data stored in a TreeSummarizedExperiment object.

Usage

```
getNMF(x, ...)

addNMF(x, ...)

## S4 method for signature 'SummarizedExperiment'
getNMF(x, k = 2, assay.type = "counts", eval.metric = "evar", ...)

## S4 method for signature 'SummarizedExperiment'
addNMF(
  x,
  k = 2,
  assay.type = "counts",
  eval.metric = "evar",
  name = "NMF",
  ...
)
```

Arguments

x	a TreeSummarizedExperiment object.
...	optional arguments passed to nmf::NMF.
k	numeric vector. A number of latent vectors/topics. (Default: 2)
assay.type	Character scalar. Specifies which assay to use for NMF ordination. (Default: "counts")
eval.metric	Character scalar. Specifies the evaluation metric that will be used to select the model with the best fit. Must be one of the following options: "evar" (explained variance; maximized), "sparseness.basis" (degree of sparsity in the basis matrix; maximized), "sparseness.coef" (degree of sparsity in the coefficient matrix; maximized), "rss" (residual sum of squares; minimized), "silhouette.coef" (quality of clustering based on the coefficient matrix; maximized), "silhouette.basis" (quality of clustering based on the basis matrix; maximized), "cophenetic" (correlation between cophenetic distances and original distances; maximized), "dispersion" (spread of data points within clusters; minimized). (Default: "evar")
name	Character scalar. The name to be used to store the result in the reducedDims of the output. (Default: "NMF")

Details

The functions getNMF and addNMF internally use nmf::NMF compute the ordination matrix and feature loadings.

If k is a vector of integers, NMF output is calculated for all the rank values contained in k, and the best fit is selected based on eval.metric value.

Value

For getNMF, the ordination matrix with feature loadings matrix as attribute "loadings".

For addNMF, a TreeSummarizedExperiment object is returned containing the ordination matrix in reducedDims(x, name) with the following attributes:

- "loadings" which is a matrix containing the feature loadings
- "NMF_output" which is the output of function nmf::NMF
- "best_fit" which is the result of the best fit if k is a vector of integers

Examples

```
data(GlobalPatterns)
tse <- GlobalPatterns

# Reduce the number of features
tse <- agglomerateByPrevalence(tse, rank = "Phylum")

# Run NMF and add the result to reducedDim(tse, "NMF").
tse <- addNMF(tse, k = 2, name = "NMF")
```

```
# Extract feature loadings
loadings_NMF <- getReducedDimAttribute(tse, "NMF", "loadings")
head(loadings_NMF)

# Estimate models with number of topics from 2 to 4. Perform 2 runs.
tse <- addNMF(tse, k = c(2, 3, 4), name = "NMF_4", nrun = 2)

# Extract feature loadings
loadings_NMF_4 <- getReducedDimAttribute(tse, "NMF_4", "loadings")
head(loadings_NMF_4)
```

`getPERMANOVA` *Calculate PERMANOVA (Permutational Multivariate Analysis of Variance)*

Description

These functions perform PERMANOVA to assess the significance of group differences based on a specified dissimilarity matrix. The results can be returned directly or added to metadata in an object of class `TreeSummarizedExperiment`.

Usage

```
getPERMANOVA(x, ...)

addPERMANOVA(x, ...)

## S4 method for signature 'SingleCellExperiment'
getPERMANOVA(x, ...)

## S4 method for signature 'SummarizedExperiment'
getPERMANOVA(x, assay.type = "counts", formula = NULL, col.var = NULL, ...)

## S4 method for signature 'ANY'
getPERMANOVA(x, formula, data, method = "bray", test.homogeneity = TRUE, ...)

## S4 method for signature 'SummarizedExperiment'
addPERMANOVA(x, name = "permanova", ...)
```

Arguments

`x`	a `SummarizedExperiment` object.
`...`	additional arguments passed to `vegan::adonis2`. • `by`: `Character scalar`. Specifies how significance is calculated. (Default: `"margin"`) • `na.action`: `function`. Action to take when missing values for any of the variables in `formula` are encountered. (Default: `na.fail`)

- `full` `Logical scalar`. should all the results from the homogeneity calculations be returned. When `FALSE`, only summary tables are returned. (Default: `FALSE`)
- `homogeneity.test`: `Character scalar`. Specifies the significance test used to analyse `vegan::betadisper` results. Options include 'permanova' (`vegan::permutest`), 'anova' (`stats::anova`) and 'tukeyhsd' (`stats::TukeyHSD`). (Default: `"permanova"`)

`assay.type` `Character scalar`. Specifies the name of assay used in calculation. (Default: `"counts"`)

`formula` `formula`. If `x` is a `SummarizedExperiment` a formula can be supplied. Based on the right-hand side of the given formula `colData` is subset to `col.var`.

`col.var` and `formula` can be missing, which turns the CCA analysis into a CA analysis and dbRDA into PCoA/MDS.

`col.var` `Character scalar`. When `x` is a `SummarizedExperiment`,`col.var` can be used to specify variables from `colData`.

`data` `data.frame` or coarcible to one. The covariance table including covariates defined by `formula`.

`method` `Character scalar`. A dissimilarity metric used in PERMANOVA and group dispersion calculation. (Default: `"bray"`)

`test.homogeneity`
`Logical scalar`. Should the homogeneity of group dispersions be evaluated? (Default: `TRUE`)

`name` `Character scalar`. A name for the results that will be stored to metadata. (Default: `"permanova"`)

Details

PERMANOVA is a non-parametric method used to test whether the centroids of different groups (as defined by the formula or covariates) are significantly different in terms of multivariate space.

PERMANOVA relies on the assumption of group homogeneity, meaning the groups should be distinct and have similar variances within each group. This assumption is essential as PERMANOVA is sensitive to differences in within-group dispersion, which can otherwise confound results. This is why the functions return homogeneity test results by default.

The functions utilize `vegan::adonis2` to compute PERMANOVA. For homogeneity testing, `vegan::betadisper` along with `vegan::permutest` are utilized by default, which allow testing for equal dispersion across groups and validate the homogeneity assumption.

PERMANOVA and distance-based redundancy analysis (dbRDA) are closely related methods for analyzing multivariate data. PERMANOVA is non-parametric, making fewer assumptions about the data. In contrast, dbRDA assumes a linear relationship when constructing the ordination space, although it also employs a PERMANOVA-like approach to test the significance of predictors within this space. dbRDA offers a broader scope overall, as it provides visualization of the constrained ordination, which can reveal patterns and relationships. However, when the underlying data structure is non-linear, the results from the two methods can differ significantly due to dbRDA's reliance on linear assumptions.

Value

getPERMANOVA returns the PERMANOVA results or a list containing the PERMANOVA results and homogeneity test results if test.homogeneity = TRUE. addPERMANOVA adds these results to metadata of x.

See Also

For more details on the actual implementation see vegan::adonis2, vegan::betadisper, and vegan::permutest. See also addCCA and addRDA

Examples

```
data(GlobalPatterns)
tse <- GlobalPatterns

# Apply relative transformation
tse <- transformAssay(tse, method = "relabundance")
# Perform PERMANOVA
tse <- addPERMANOVA(
    tse,
    assay.type = "relabundance",
    method = "bray",
    formula = x ~ SampleType,
    permutations = 99
    )
# The results are stored to metadata
metadata(tse)[["permanova"]]

# Calculate dbRDA
rda_res <- getRDA(
    tse, assay.type = "relabundance", method = "bray",
    formula = x ~ SampleType, permutations = 99)
# Significance results are similar to PERMANOVA
attr(rda_res, "significance")
```

---

getPrevalence *Calculation prevalence information for features across samples*

Description

These functions calculate the population prevalence for taxonomic ranks in a SummarizedExperiment object.

Usage

```
getPrevalence(x, ...)

getPrevalent(x, ...)
```

```
getRare(x, ...)

subsetByPrevalent(x, ...)

subsetByRare(x, ...)

getPrevalentAbundance(
  x,
  assay.type = assay_name,
  assay_name = "relabundance",
  ...
)

addPrevalentAbundance(x, ...)

addPrevalence(x, ...)

## S4 method for signature 'SummarizedExperiment'
addPrevalence(x, name = "prevalence", ...)

## S4 method for signature 'ANY'
getPrevalence(
  x,
  detection = 0,
  include.lowest = include_lowest,
  include_lowest = FALSE,
  sort = FALSE,
  na.rm = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
getPrevalence(
  x,
  assay.type = assay_name,
  assay_name = "counts",
  rank = NULL,
  ...
)

## S4 method for signature 'ANY'
getPrevalent(
  x,
  prevalence = 50/100,
  include.lowest = include_lowest,
  include_lowest = FALSE,
  ...
```

```
)

## S4 method for signature 'SummarizedExperiment'
getPrevalent(
  x,
  rank = NULL,
  prevalence = 50/100,
  include.lowest = include_lowest,
  include_lowest = FALSE,
  ...
)

## S4 method for signature 'ANY'
getRare(
  x,
  prevalence = 50/100,
  include.lowest = include_lowest,
  include_lowest = FALSE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
getRare(
  x,
  rank = NULL,
  prevalence = 50/100,
  include.lowest = include_lowest,
  include_lowest = FALSE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
subsetByPrevalent(x, rank = NULL, ...)

## S4 method for signature 'TreeSummarizedExperiment'
subsetByPrevalent(x, update.tree = TRUE, ...)

## S4 method for signature 'SummarizedExperiment'
subsetByRare(x, rank = NULL, ...)

## S4 method for signature 'TreeSummarizedExperiment'
subsetByRare(x, update.tree = TRUE, ...)

## S4 method for signature 'SummarizedExperiment'
addPrevalentAbundance(x, name = "prevalent_abundance", ...)

## S4 method for signature 'ANY'
getPrevalentAbundance(
```

```
  x,
  assay.type = assay_name,
  assay_name = "relabundance",
  ...
)

## S4 method for signature 'SummarizedExperiment'
getPrevalentAbundance(x, assay.type = assay_name, assay_name = "counts", ...)
```

Arguments

x	TreeSummarizedExperiment.
...	additional arguments • If !is.null(rank) arguments are passed on to agglomerateByRank. See ?agglomerateByRank for more details. • for getPrevalent, getRare, subsetByPrevalent and subsetByRare additional parameters passed to getPrevalence • for getPrevalentAbundance additional parameters passed to getPrevalent
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
name	Character scalar. Specifies name of column in rowData where the results will be stored. (Default: "prevalence")
detection	Numeric scalar. Detection threshold for absence/presence. If as_relative = FALSE, it sets the counts threshold for a taxon to be considered present. If as_relative = TRUE, it sets the relative abundance threshold for a taxon to be considered present. (Default: 0)
include.lowest	Logical scalar. Should the lower boundary of the detection and prevalence cutoffs be included? (Default: FALSE)
include_lowest	Deprecated. Use include.lowest instead.
sort	Logical scalar. Should the result be sorted by prevalence? (Default: FALSE)
na.rm	Logical scalar. Should NA values be omitted? (Default: TRUE)
rank	Character scalar. Defines a taxonomic rank. Must be a value of taxonomyRanks() function.
prevalence	Prevalence threshold (in 0 to 1). The required prevalence is strictly greater by default. To include the limit, set include.lowest to TRUE.
update.tree	Logical scalar. Should rowTree() also be agglomerated? (Default: TRUE)

Details

getPrevalence calculates the frequency of samples that exceed the detection threshold. For SummarizedExperiment objects, the prevalence is calculated for the selected taxonomic rank, otherwise for the rows. The absolute population prevalence can be obtained by multiplying the prevalence by the number of samples (ncol(x)).

The core abundance index from getPrevalentAbundance gives the relative proportion of the core species (in between 0 and 1). The core taxa are defined as those that exceed the given population prevalence threshold at the given detection level as set for getPrevalent.

subsetPrevalent and subsetRareFeatures return a subset of x. The subset includes the most prevalent or rare taxa that are calculated with getPrevalent or getRare respectively.

getPrevalent returns taxa that are more prevalent with the given detection threshold for the selected taxonomic rank.

getRare returns complement of getPrevalent.

Value

subsetPrevalent and subsetRareFeatures return subset of x.

All other functions return a named vectors:

- getPrevalence returns a numeric vector with the names being set to either the row names of x or the names after agglomeration. addPrevalence adds these results to rowData(x).
- getPrevalentAbundance returns a numeric vector with the names corresponding to the column name of x and include the joint abundance of prevalent taxa.
- getPrevalent and getRare return a character vector with only the names exceeding the threshold set by prevalence, if the rownames of x is set. Otherwise an integer vector is returned matching the rows in x.

References

A Salonen et al. The adult intestinal core microbiota is determined by analysis depth and health status. Clinical Microbiology and Infection 18(S4):16 20, 2012. To cite the R package, see citation('mia')

See Also

agglomerateByRank, getTop

Examples

```
data(GlobalPatterns)
tse <- GlobalPatterns
# Get prevalence estimates for individual ASV/OTU
prevalence.frequency <- getPrevalence(tse,
                                      detection = 0,
                                      sort = TRUE)
head(prevalence.frequency)

# Get prevalence estimates for phyla
# - the getPrevalence function itself always returns population frequencies
prevalence.frequency <- getPrevalence(tse,
                                      rank = "Phylum",
                                      detection = 0,
                                      sort = TRUE)
head(prevalence.frequency)
```

```
# - to obtain population counts, multiply frequencies with the sample size,
# which answers the question "In how many samples is this phylum detectable"
prevalence.count <- prevalence.frequency * ncol(tse)
head(prevalence.count)

# Detection threshold 1 (strictly greater by default);
# Note that the data (GlobalPatterns) is here in absolute counts
# (and not compositional, relative abundances)
# Prevalence threshold 50 percent (strictly greater by default)
prevalent <- getPrevalent(
    tse,
    rank = "Phylum",
    detection = 10,
    prevalence = 50/100)
head(prevalent)

# Add relative aundance data
tse <- transformAssay(tse, assay.type = "counts", method = "relabundance")

# Gets a subset of object that includes prevalent taxa
altExp(tse, "prevalent") <- subsetByPrevalent(tse,
                                              rank = "Family",
                                              assay.type = "relabundance",
                                              detection = 0.001,
                                              prevalence = 0.55)
altExp(tse, "prevalent")

# getRare returns the inverse
rare <- getRare(tse,
    rank = "Phylum",
    assay.type = "relabundance",
    detection = 1/100,
    prevalence = 50/100)
head(rare)

# Gets a subset of object that includes rare taxa
altExp(tse, "rare") <- subsetByRare(
    tse,
    rank = "Class",
    assay.type = "relabundance",
    detection = 0.001,
    prevalence = 0.001)
altExp(tse, "rare")

# Names of both experiments, prevalent and rare, can be found from slot
# altExpNames
tse

data(esophagus)
getPrevalentAbundance(esophagus, assay.type = "counts")
```

getReducedDimAttribute *Utilization functions for* TreeSummarizedExperiment *objects*

Description

A set of utility functions designed to facilitate operations with TreeSummarizedExperiment objects

Usage

```
getReducedDimAttribute(x, ...)

## S4 method for signature 'SingleCellExperiment'
getReducedDimAttribute(x, dimred = 1L, name = NULL, ...)
```

Arguments

x	a SummarizedExperiment object.
...	additional arguments, not used currently.
dimred	Character scalar or integer scalar. A name or index of dimension reduction results. (Default: 1L)
name	Character vector. A name of values retrieved from attributes of reducedDim(x, dimred). If NULL, all the values are retrieved. (Default: NULL)

Details

getReducedDimAttribute is a utility function that retrieves specific elements from the attributes of reducedDim in a TreeSummarizedExperiment object. These attributes may contain loadings, statistical test results, or other metadata, depending on the methods used to generate the results.

Value

The extracted element from the reducedDim attribute.

See Also

runCCA, addNMF, and addLDA

Examples

```
data(GlobalPatterns)
tse <- GlobalPatterns

# Reduce the number of features
tse <- agglomerateByPrevalence(tse, rank = "Phylum")

# Run NMF and add the result to reducedDim(tse, "NMF").
```

```
tse <- addNMF(tse, k = 1, name = "NMF")

# Extract feature loadings
res <- getReducedDimAttribute(tse, dimred = "NMF", name = "loadings")
res |> head()
```

---

getRPCA *Run (joint) robust principal component analysis (RPCA)*

Description

These functions implement robust principal component analysis (RPCA) for single tables and joint RPCA for multiple tables. `*RPCA` functions run RPCA for single table. `*JointRPCA` runs the analysis jointly for multiple tables. `get*` functions return the results of the analysis while `add*` adds them to the input object.

Usage

```
getRPCA(x, ...)

addRPCA(x, ...)

getJointRPCA(x, ...)

addJointRPCA(x, ...)

## S4 method for signature 'SingleCellExperiment'
getRPCA(x, ...)

## S4 method for signature 'SummarizedExperiment'
getRPCA(x, assay.type = "counts", ...)

## S4 method for signature 'SummarizedExperiment'
addRPCA(x, name = "RPCA", ...)

## S4 method for signature 'MultiAssayExperiment'
getJointRPCA(x, experiments, assay.types, ...)

## S4 method for signature 'SingleCellExperiment'
getJointRPCA(x, experiments, assay.types, ...)

## S4 method for signature 'SingleCellExperiment'
addJointRPCA(x, name = "JointRPCA", ...)

## S4 method for signature 'MultiAssayExperiment'
addJointRPCA(x, name = "JointRPCA", ...)
```

Arguments

x	a SummarizedExperiment object.
...	additional arguments: • ncomponents:Integer scalar. The number of components to estimate. (Default: 3) • max.iterations:Integer scalar. The number of iterations run in OptSpace algorithm. (Default: 3) • tolerance:Numeric scalar. Accepted error between lower rank representation obtained by OptSpace algorithm and original table. (Default: 3)
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
name	Character scalar. Name of results stored in x. (Default: "RPCA" or "JointRPCA" depending on the method)
experiments	Character vector or integer scalar. Names or indices of experiments selected from x.
assay.types	Character vector. Names assays selected from experiments.

Details

These functions perform robust principal component analysis (RPCA) using a low-rank matrix approximation followed by principal component analysis. Missing values are handled using matrix completion based on the OptSpace algorithm.

Single-table RPCA

For a single assay, the workflow is:

1. Extract the selected assay matrix
2. Estimate a low-rank approximation of the matrix using the OptSpace algorithm, which reconstructs missing values and reduces noise.
3. Apply double centering (row and column centering).
4. Apply PCA (via singular value decomposition) to the centered matrix.
5. Return the sample coordinates in PCA space together with additional information such as loadings, explained variance, and pairwise distances.

Joint RPCA

When multiple assays are provided, joint RPCA estimates a shared low-rank representation across all tables. Each table may contain different features, but they must share the same samples.

The workflow is:

1. Extract the selected experiments and assays.
2. Estimate a joint low-rank representation using a joint OptSpace algorithm that learns shared sample factors while allowing table-specific feature loadings.
3. Apply double centering (row and column centering).
4. Perform PCA on the centered matrix. to obtain the final sample coordinates.

5. Return the sample coordinates in PCA space together with additional information such as loadings, explained variance, and pairwise distances.

To assess generalization, the joint RPCA procedure optionally splits the samples into training and test sets. The low-rank model is learned using the training data and the test samples are projected into the resulting PCA space. Reconstruction error for the test set is returned as a measure of model fit.

The returned object contains the PCA sample scores as the main result, with additional information (e.g., rotation matrix, explained variance, reconstructed matrix, and reconstruction error) stored in attributes.

Value

`matrix`, `TreeSummarizedExperiment` or `MultiAssayExperiment` object.

Author(s)

The RPCA method is reported in Martino et al. (2020) and the R/Bioconductor implementation utilizes the robust Aitchison distance from `vegan::decostand`.

The Joint-RPCA method was adapted from the original Python-based implementation in biocore/Gemelli by Bianca Cordazzo Vargas, Liat Shenhav, and Cameron Martino. The R/Bioconductor implementation was subsequently prepared by Aituar Bektanov, Tuomas Borman, and Leo Lahti.

References

Martino, C. and Shenhav, L. et al. (2020) Context-aware dimensionality reduction deconvolutes gut microbial community dynamics. *Nat. Biotechnol.* doi:10.1038/s41587-020-0660-7

See Also

`scater::runPCA`

Examples

```
data("ibdmdb")
mae <- ibdmdb

# Apply data transformations
mae[[1]] <- transformAssay(mae[[1]], assay.type = "mgx", method = "rclr")
mae[[2]] <- transformAssay(mae[[2]], assay.type = "mtx", method = "rclr")

# Run joint-RPCA
res <- getJointRPCA(
    mae,
    experiments = c(1, 2),
    assay.types = c("rclr", "rclr")
)

# Run RPCA for single experiment
res <- getRPCA(mae[[1]], assay.type = "rclr")
```

`GlobalPatterns` *Global patterns of 16S rRNA diversity at a depth of millions of sequences per sample.*

Description

GlobalPatterns compared the microbial communities from 25 environmental samples and three known "mock communities" at a an average depth of 3.1 million reads per sample. Authors reproduced diversity patterns seen in many other published studies, while investigating technical bias by applying the same techniques to simulated microbial communities of known composition. Special thanks are given to J. Gregory Caporaso for providing the OTU-clustered data files for inclusion in the **phyloseq** package, from which this data was converted to `TreeSummarizedExperiment`.

Usage

```
data(GlobalPatterns)
```

Format

A TreeSummarizedExperiment with 19216 features and 26 samples. The rowData contains taxonomic information at Kingdom, Phylum, Class, Order, Family, Genus and Species levels. The colData includes:

X.SampleID Sample ID taken from the corresponding study

Primer primer used for sequencing

Final_Barcode final barcode (6 nucleotides)

Barcode_truncated_plus_T truncated barcode with an added tyrosine (6 nucleotides)

Barcode_full_length complete barcode with a length of 11 nucleotides

SampleType sampling type by collection site (Soil, Feces, Skin, Tongue, Freshwater, Creek Freshwater, Ocean, Estuary Sediment and Mock)

Description additional information (sampling location, environmental factors and study type)

Author(s)

Caporaso, J. G., et al.

References

Caporaso, J. G., et al. (2011). Global patterns of 16S rRNA diversity at a depth of millions of sequences per sample. PNAS, 108, 4516-4522. https://doi.org/10.1073/pnas.1000080107

See Also

mia-datasets

HintikkaXOData *Multiomics dataset from 40 rat samples*

Description

HintikkaXO is a multiomics dataset from a rat experiment studying effect of fat and prebiotics in diet. It contains high-throughput profiling data from 40 rat samples, including 39 biomarkers, 38 metabolites (NMR), and 12706 OTUs from 318 species, measured from Cecum. This is diet comparison study with High/Low fat diet and xylo-oligosaccaride supplementation. Column metadata is common for all experiments (microbiota, metabolites, biomarkers) and is described below.

Usage

```
data(HintikkaXOData)
```

Format

A MultiAssayExperiment with 3 experiments (microbiota, metabolites and biomarkers). rowData of the microbiota experiment contains taxonomic information at Phylum, Class, Order, Family, Genus, Species and OTU levels. The metabolites and biomarkers experiments contain 38 NMR metabolites and 39 biomarkers, respectively. The colData includes:

Sample Sample ID (character)

Rat Rat ID (factor)

Site Site of measurement ("Cecum"); single value

Diet Diet group (factor; combination of the Fat and XOS fields)

Fat Fat in Diet (factor; Low/High)

XOS XOS Diet Supplement (numeric; 0/1)

Author(s)

Hintikka L et al.

References

Hintikka L et al. (2021): Xylo-oligosaccharides in prevention of hepatic steatosis and adipose tissue inflammation: associating taxonomic and metabolomic patterns in fecal microbiota with biclustering. International Journal of Environmental Research and Public Health 18(8):4049. https://doi.org/10.3390/ijerph18084049

See Also

mia-datasets

ibdmdb *IBDMDB 2-omic demo dataset (MGX + MTX)*

Description

A compact example derived from the Integrative Human Microbiome Project (iHMP) Inflammatory Bowel Disease (IBD) cohort.

Usage

```
data("ibdmdb")
```

Format

A `MultiAssayExperiment` with two experiments:

se_mgx A `SummarizedExperiment` with MGX abundance data.

se_mtx A `SummarizedExperiment` with MTX abundance data.

mae2 A `MultiAssayExperiment` containing both experiments.

Details

The dataset `ibdmdb` is a named list with:

- `se_mgx`: metagenomic taxonomic profiles (MGX)
- `se_mtx`: metatranscriptomic taxonomic profiles (MTX)
- `mae2`: a MultiAssayExperiment containing both experiments

These compact objects are intended for quick examples and vignettes. Load with: `data("ibdmdb")`.

Source

Prepared by `inst/scripts/prepare_ibdmdb_demo.R`. The script is for developers and downloads publicly available IBDMDB/HMP2 taxonomic profiles and metadata from Zenodo into a local cache (e.g. `tools/cache/`), then subsets and filters the raw data to create a small example suitable for vignettes.

References

Lloyd-Price J, Arze C, Ananthakrishnan AN, et al. *Multi-omics of the gut microbial ecosystem in inflammatory bowel diseases.* Nature 569, 655–662 (2019). doi:10.1038/s41586019123 79

importHUMAnN *Import HUMAnN results to* TreeSummarizedExperiment

Description

Import HUMAnN results to TreeSummarizedExperiment

Arguments

file	Character scalar. Defines the file path of the HUMAnN file. The file must be in merged HUMAnN format.
col.data	a DataFrame-like object that includes sample names in rownames, or a single character value defining the file path of the sample metadata file. The file must be in tsv format (Default: NULL).
colData	Deprecated. Use col.data instead.
...	additional arguments: • assay.type: Character scalar. Specifies the name of the assay used in calculation. (Default: "counts") • prefix.rm: Logical scalar. Should taxonomic prefixes be removed? (Default: FALSE) • remove.suffix: Logical scalar. Should suffixes of sample names be removed? HUMAnN pipeline adds suffixes to sample names. Suffixes are formed from file names. By selecting remove.suffix = TRUE, you can remove pattern from end of sample names that is shared by all. (Default: FALSE)

Details

Import HUMAnN (currently version 3.0 supported) results of functional predictions based on metagenome composition (e.g. pathways or gene families). The input must be in merged HUMAnN format. (See the HUMAnN documentation and humann_join_tables method.)

The function parses gene/pathway information along with taxonomy information from the input file. This information is stored to rowData. Abundances are stored to assays.

Usually the workflow includes also taxonomy data from Metaphlan. See importMetaPhlAn to load the data to TreeSE.

Value

A TreeSummarizedExperiment object

References

Beghini F, McIver LJ, Blanco-Míguez A, Dubois L, Asnicar F, Maharjan S, Mailyan A, Manghi P, Scholz M, Thomas AM, Valles-Colomer M, Weingart G, Zhang Y, Zolfo M, Huttenhower C, Franzosa EA, & Segata N (2021) Integrating taxonomic, functional, and strain-level profiling of diverse microbial communities with bioBakery 3. *eLife*. 10:e65088.

See Also

importMetaPhlAn convertFromPhyloseq convertFromBIOM convertFromDADA2 importQIIME2 importMothur

Examples

```
# File path
file_path <- system.file("extdata", "humann_output.tsv", package = "mia")
# Import data
tse <- importHUMAnN(file_path)
tse
```

---

importMetaPhlAn *Import Metaphlan results to* TreeSummarizedExperiment

---

Description

Import Metaphlan results to TreeSummarizedExperiment

Arguments

file	NULL or DataFrame-like object. Defines the file path of the Metaphlan file. The file must be in merged Metaphlan format.
col.data	a DataFrame-like object that includes sample names in rownames, or a single character value defining the file path of the sample metadata file. The file must be in tsv format (Default: NULL).
colData	Deprecated. use col.data instead.
sample_meta	Deprecated. Use col.data instead.
tree.file	Character scalar. Defines the file path of the phylogenetic tree. (Default: NULL).
phy_tree	Deprecated. Use tree.file instead.
...	additional arguments: • assay.type: Character scalar. Specifies the name of the assay used in calculation. (Default: "metaphlan") • prefix.rm: Logical scalar. Should taxonomic prefixes be removed? (Default: FALSE) • remove.suffix: Logical scalar. Should suffixes of sample names be removed? Metaphlan pipeline adds suffixes to sample names. Suffixes are formed from file names. By selecting remove.suffix = TRUE, you can remove pattern from end of sample names that is shared by all. (Default: FALSE) • set.ranks: Logical scalar. Should the columns in the rowData that are treated as taxonomy ranks be updated according to the ranks found in the imported data? (Default: FALSE)

Details

Import Metaphlan (versions 2, 3 and 4 supported) results. Input must be in merged Metaphlan format. (See the Metaphlan documentation and `merge_metaphlan_tables` method.) Data is imported so that data at the lowest rank is imported as a `TreeSummarizedExperiment` object. Data at higher rank is imported as a `SummarizedExperiment` objects which are stored to `altExp` of `TreeSummarizedExperiment` object.

Currently Metaphlan versions 2, 3, and 4 are supported.

Value

A `TreeSummarizedExperiment` object

References

Beghini F, McIver LJ, Blanco-Míguez A, Dubois L, Asnicar F, Maharjan S, Mailyan A, Manghi P, Scholz M, Thomas AM, Valles-Colomer M, Weingart G, Zhang Y, Zolfo M, Huttenhower C, Franzosa EA, & Segata N (2021) Integrating taxonomic, functional, and strain-level profiling of diverse microbial communities with bioBakery 3. *eLife*. 10:e65088. doi: 10.7554/eLife.65088

See Also

`convertFromPhyloseq` `convertFromBIOM` `convertFromDADA2` `importQIIME2` `importMothur`

Examples

```
# (Data is from tutorial
# https://github.com/biobakery/biobakery/wiki/metaphlan3#merge-outputs)

# File path
file_path <- system.file(
    "extdata", "merged_abundance_table.txt", package = "mia")
# Import data
tse <- importMetaPhlAn(file_path)
# Data at the lowest rank
tse
# Data at higher rank is stored in altExp
altExps(tse)
# Higher rank data is in SE format, for example, Phylum rank
altExp(tse, "phylum")
```

---

importMothur *Import Mothur results as a* `TreeSummarizedExperiment`

Description

This method creates a `TreeSummarizedExperiment` object from `Mothur` files provided as input.

Usage

```
importMothur(
  assay.file = sharedFile,
  sharedFile,
  taxonomyFile = NULL,
  row.file = taxonomyFile,
  designFile = NULL,
  col.file = designFile
)
```

Arguments

assay.file	Character scalar. Defines the file path of the feature table to be imported. The File has to be in shared file format as defined in Mothur documentation.
sharedFile	Deprecated. Use assay.file instead.
taxonomyFile	Deprecated. Use row.file instead.
row.file	Character scalar. Defines the file path of the taxonomy table to be imported. The File has to be in taxonomy file or constaxonomy file format as defined in Mothur documentation. (Default: NULL).
designFile	Deprecated. Use col.file instead.
col.file	Character scalar. Defines file path of the sample metadata to be imported. The File has to be in design file format as defined in Mothur documentation. (Default: NULL).

Details

Results exported from Mothur can be imported as a SummarizedExperiment using importMothur. Except for the assay.file, the other data types, row.file, and col.file, are optional, but are highly encouraged to be provided.

Value

A TreeSummarizedExperiment object

References

https://mothur.org/ https://mothur.org/wiki/shared_file/ https://mothur.org/wiki/taxonomy_file/ https://mothur.org/wiki/constaxonomy_file/ https://mothur.org/wiki/design_file/

See Also

convertFromPhyloseq convertFromBIOM convertFromDADA2 importQIIME2

Examples

```
# Abundance table
counts <- system.file("extdata", "mothur_example.shared", package = "mia")
# Taxa table (in "cons.taxonomy" or "taxonomy" format)
taxa <- system.file(
    "extdata", "mothur_example.cons.taxonomy", package = "mia")
#taxa <- system.file("extdata", "mothur_example.taxonomy", package = "mia")
# Sample meta data
meta <- system.file("extdata", "mothur_example.design", package = "mia")

# Creates se object from files
se <- importMothur(assay.file = counts, row.file = taxa, col.file = meta)
# Convert SE to TreeSE
tse <- as(se, "TreeSummarizedExperiment")
tse
```

---

importQIIME2 *Import QIIME2 results to* TreeSummarizedExperiment

---

Description

Results exported from QIMME2 can be imported as a `TreeSummarizedExperiment` using `importQIIME2`. Except for the `assay.file`, the other data types, `row.file`, `refseq.file` and `tree.file`, are optional, but are highly encouraged to be provided.

Import the QIIME2 artifacts to R.

Usage

```
importQIIME2(
  assay.file = featureTableFile,
  featureTableFile,
  row.file = taxonomyTableFile,
  taxonomyTableFile = NULL,
  col.file = sampleMetaFile,
  sampleMetaFile = NULL,
  as.refseq = featureNamesAsRefSeq,
  featureNamesAsRefSeq = TRUE,
  refseq.file = refSeqFile,
  refSeqFile = NULL,
  tree.file = phyTreeFile,
  phyTreeFile = NULL,
  ...
)

importQZA(file, temp.dir = temp, temp = tempdir(), ...)
```

Arguments

assay.file	Character scalar. Defines the file path of the feature table to be imported.
featureTableFile	Deprecated. use assay.file instead.
row.file	Character scalar or NULL. Defines the file path of the taxonomy table to be imported. (default: NULL).
taxonomyTableFile	Deprecated. use row.file instead.
col.file	Character scalar or NULL. Defines the file path of the sample metadata to be imported. The file has to be in tsv format. (Default: NULL).
sampleMetaFile	Deprecated. Use col.file instead.
as.refseq	Logical scalar or NULL. Should the feature names of the feature table be regarded as reference sequences? This setting will be disregarded, if refseq.file is not NULL. If the feature names do not contain valid DNA characters only, the reference sequences will not be set.
featureNamesAsRefSeq	Deprecated. Use as.refseq instead.
refseq.file	Character scalar or NULL. Defines the file path of the reference sequences for each feature. (Default: NULL).
refSeqFile	Deprecated. Use refseq.file instead.
tree.file	Character scalar. Defines the file path of the phylogenetic tree. (Default: NULL).
phyTreeFile	Deprecated. Use tree.file instead.
...	additional arguments: • temp.dir: the temporary directory used for decompressing the data. (default: tempdir()) • prefix.rm: TRUE or FALSE: Should taxonomic prefixes be removed? (default: prefix.rm = FALSE)
file	character, path of the input qza file. Only files in format of BIOMV210DirFmt (feature table), TSVTaxonomyDirectoryFormat (taxonomic table), NewickDirectoryFormat (phylogenetic tree) and DNASequencesDirectoryFormat (representative sequences) are supported right now.
temp.dir	character, a temporary directory in which the qza file will be decompressed to, default tempdir().
temp	Deprecated. Use temp.dir instead.

Details

Both arguments as.refseq and refseq.file can be used to define reference sequences of features. as.refseq is only taken into account, if refseq.file is NULL. No reference sequences are tried to be created, if featureNameAsRefSeq is FALSE and refseq.file is NULL.

Value

A TreeSummarizedExperiment object

matrix object for feature table, DataFrame for taxonomic table, ape::phylo object for phylogenetic tree, Biostrings::DNAStringSet for representative sequences of taxa.

Author(s)

Yang Cao

References

Bolyen E et al. 2019: Reproducible, interactive, scalable and extensible microbiome data science using QIIME 2. Nature Biotechnology 37: 852–857. https://doi.org/10.1038/s41587-019-0209-9

https://qiime2.org

See Also

convertFromPhyloseq convertFromBIOM convertFromDADA2 importMothur

Examples

```
assay.file <- system.file("extdata", "table.qza", package = "mia")
row.file <- system.file("extdata", "taxonomy.qza", package = "mia")
col.file <- system.file("extdata", "sample-metadata.tsv", package = "mia")
tree.file <- system.file("extdata", "tree.qza", package = "mia")
refseq.file <- system.file("extdata", "refseq.qza", package = "mia")
tse <- importQIIME2(
  assay.file = assay.file,
  row.file = row.file,
  col.file = col.file,
  refseq.file = refseq.file,
  tree.file = tree.file
)

tse
# Read individual files
assay.file <- system.file("extdata", "table.qza", package = "mia")
row.file <- system.file("extdata", "taxonomy.qza", package = "mia")
col.file <- system.file("extdata", "sample-metadata.tsv", package = "mia")

assay <- importQZA(assay.file)
rowdata <- importQZA(row.file, prefix.rm = TRUE)
coldata <- read.table(col.file, header = TRUE, sep = "\t", comment.char = "")

# Assign rownames
rownames(coldata) <- coldata[, 1]
coldata[, 1] <- NULL

# Order coldata based on assay
coldata <- coldata[match(colnames(assay), rownames(coldata)), ]
```

```
# Create SE from individual files
se <- SummarizedExperiment(
    assays = list(assay), rowData = rowdata, colData = coldata)
se
```

`importTaxpasta` *Import taxpasta-specific BIOM results to* `TreeSummarizedExperiment`

Description

Import taxpasta-specific BIOM results to `TreeSummarizedExperiment`

Usage

```
importTaxpasta(file, add.tree = TRUE, ...)
```

Arguments

`file`	`Character scalar`. Defines the file path to a BIOM file.
`add.tree`	`Logical scalar`. Specifies whether to calculate and add hierarchy tree using `addHierarchyTree`. (Default: `TRUE`)
`...`	additional arguments • `set.ranks`: `Logical scalar`. Should column names of taxonomy table be treated as taxonomy ranks? (Default: `FALSE`)

Details

`importTaxpasta` imports data that is returned from Taxonomic Profile Aggregation and Standardization (taxpasta) pipeline. See more information on taxpasta from taxpasta documentation.

Value

A `TreeSummarizedExperiment` object.

See Also

`importBIOM` `convertFromBIOM`

Examples

```
## Not run:
# File path to BIOM file
file_path <- system.file("extdata", "complete.biom", package = "mia")
# Import BIOM as TreeSE, and set ranks.
tse <- importTaxpasta(file_path, set.ranks = TRUE)
# Import BIOM as TreeSE without adding hierarchy tree
tse <- importTaxpasta(file_path, add.tree = FALSE)

## End(Not run)
```

---

meltSE *Converting a SummarizedExperiment object into a long data.frame*

Description

meltSE Converts a SummarizedExperiment object into a long data.frame which can be used for tidyverse-tools.

Usage

```
meltSE(x, ...)

## S4 method for signature 'SummarizedExperiment'
meltSE(
  x,
  assay.type = assay_name,
  assay_name = "counts",
  add.row = add_row_data,
  add_row_data = NULL,
  add.col = add_col_data,
  add_col_data = NULL,
  row.name = feature_name,
  feature_name = "FeatureID",
  col.name = sample_name,
  sample_name = "SampleID",
  ...
)

## S4 method for signature 'SingleCellExperiment'
meltSE(x, add.dimred = NULL, ...)
```

Arguments

x	TreeSummarizedExperiment.
...	optional arguments:

- check.names: Logical scalar. Passed to data.frame function's check.name argument. Determines if sample names are checked that they are syntactically valid variable names and are not duplicated. If they are not, sample names are modified. (Default: TRUE)

assay.type — Character scalar. Specifies the name of assay used in calculation. (Default: "counts")

assay_name — Deprecated. Use assay.type instead.

add.row — Logical scalar or Character vector. To select information from the rowData to add to the molten assay data. If add.row = NULL no data will be added, if add.row = TRUE all data will be added and if add.row is a character vector, it will be used to subset to given column names in rowData. (Default: NULL)

add_row_data — Deprecated. Use add.row instead.

add.col — Logical scalar. NULL, or character vector. Used to select information from the colData to add to the molten assay data. If add.col = NULL no data will be added, if add.col = TRUE all data will be added and if add.col is a character vector, it will be used to subset to given column names in colData. (Default: NULL)

add_col_data — Deprecated. Use add.col instead.

row.name — Character scalar. To use as the output's name for the feature identifier. (Default: "FeatureID")

feature_name — Deprecated. Use row.name instead.

col.name — Character scalar. To use as the output's name for the sample identifier. (Default: "SampleID")

sample_name — Deprecated. Use col.name instead.

add.dimred — Logical scalar or Character vector. To select information from the reducedDim to add to the molten assay data. If add.dimred = NULL no data will be added, if add.dimred = TRUE all data will be added and if add.dimred is a character vector, it will be used to subset to given names in reducedDimNames(x). (Default: NULL)

Details

If the colData contains a column "SampleID" or the rowData contains a column "FeatureID", they will be renamed to "SampleID_col" and "FeatureID_row", if row names or column names are set.

Value

A tibble with the molten data. The assay values are given in a column named like the selected assay assay.type. In addition, a column "FeatureID" will contain the rownames, if set, and analogously a column "SampleID" with the colnames, if set.

Examples

```
data(GlobalPatterns)
molten_tse <- meltSE(
    GlobalPatterns,
```

```
    assay.type = "counts",
    add.row = TRUE,
    add.col = TRUE
    )
molten_tse
```

---

mergeSEs *Merge SE objects into single SE object.*

Description

Merge SE objects into single SE object.

Usage

```
mergeSEs(x, ...)

## S4 method for signature 'SimpleList'
mergeSEs(
  x,
  assay.type = "counts",
  assay_name = NULL,
  join = "full",
  missing.values = missing_values,
  missing_values = NA,
  collapse.cols = collapse_samples,
  collapse_samples = FALSE,
  collapse.rows = collapse_features,
  collapse_features = TRUE,
  verbose = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
mergeSEs(x, y = NULL, ...)

## S4 method for signature 'list'
mergeSEs(x, ...)
```

Arguments

x	TreeSummarizedExperiment.
...	optional arguments (not used).
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.

join	Character scalar. A value for selecting the joining method. Must be 'full', 'inner', 'left', or 'right'. 'left' and 'right' are disabled when more than two objects are being merged. (Default: "full")
missing.values	NA, 0 or Character scalar. Specifies the notation of missing values. (By default: NA)
missing_values	Deprecated. Use missing.values instead.
collapse.cols	Logical scalar. Determines whether to collapse identically named samples to one. (Default: FALSE)
collapse_samples	Deprecated. Use collapse.cols instead.
collapse.rows	Logical scalar. Selects whether to collapse identically named features to one. Since all taxonomy information is taken into account, this concerns rownames-level (usually strain level) comparison. Often OTU or ASV level is just an arbitrary number series from sequencing machine meaning that the OTU information is not comparable between studies. With this option, it is possible to specify whether these strains are combined if their taxonomy information along with OTU number matches. (Default: TRUE)
collapse_features	Deprecated. Use collapse.rows instead.
verbose	Logical scalar. Choose whether to show messages. (Default: TRUE)
y	a SummarizedExperiment object when x is a SummarizedExperiment object. Disabled when x is a list.

Details

This function merges multiple SummarizedExperiment objects. It combines rowData, assays, and colData so that the output includes each unique row and column ones. The merging is done based on rownames and colnames. rowTree and colTree are preserved if linkage between rows/cols and the tree is found.

Equally named rows are interpreted as equal. Further matching based on rowData is not done. For samples, collapsing is disabled by default meaning that equally named samples that are stored in different objects are interpreted as unique. Collapsing can be enabled with collapse.cols = TRUE when equally named samples describe the same sample.

If, for example, all rows are not shared with individual objects, there are missing values in assays. The notation of missing can be specified with the missing.values argument. If input consists of TreeSummarizedExperiment objects, also rowTree, colTree, and referenceSeq are preserved if possible. The data is preserved if all the rows or columns can be found from it.

Compared to cbind and rbind mergeSEs allows more freely merging since cbind and rbind expect that rows and columns are matching, respectively.

You can choose joining methods from 'full', 'inner', 'left', and 'right'. In all the methods, all the samples are included in the result object. However, with different methods, it is possible to choose which rows are included.

- full – all unique features
- inner – all shared features

- `left` – all the features of the first object
- `right` – all the features of the second object

The output depends on the input. If the input contains `SummarizedExperiment` object, then the output will be `SummarizedExperiment`. When all the input objects belong to `TreeSummarizedExperiment`, the output will be `TreeSummarizedExperiment`.

Value

A single `SummarizedExperiment` object.

See Also

- `TreeSummarizedExperiment::cbind`
- `TreeSummarizedExperiment::rbind`
- `full_join`
- `inner_join`
- `left_join`
- `right_join`

Examples

```
data(GlobalPatterns)
data(esophagus)
data(enterotype)

# Take only subsets so that it wont take so long
tse1 <- GlobalPatterns[1:100, ]
tse2 <- esophagus
tse3 <- enterotype[1:100, ]

# Merge two TreeSEs
tse <- mergeSEs(tse1, tse2)

# Merge a list of TreeSEs
list <- SimpleList(tse1, tse2, tse3)
tse <- mergeSEs(list, assay.type = "counts", missing.values = 0)
tse

# With 'join', it is possible to specify the merging method. Subsets are used
# here just to show the functionality
tse_temp <- mergeSEs(tse[1:10, 1:10], tse[5:100, 11:20], join = "left")
tse_temp

# If your objects contain samples that describe one and same sample,
# you can collapse equally named samples to one by specifying 'collapse.cols'
tse_temp <- mergeSEs(list(tse[1:10, 1], tse[1:20, 1], tse[1:5, 1]),
                     collapse.cols = TRUE,
                     join = "inner")
tse_temp
```

```
# Merge all available assays
tse <- transformAssay(tse, method="relabundance")
ts1 <- transformAssay(tse1, method="relabundance")
tse_temp <- mergeSEs(tse, tse1, assay.type = assayNames(tse))
```

`mia-datasets` *mia datasets*

Description

mia provides various datasets derived from independent experimental studies. The datasets represent instances of the TreeSummarizedExperiment and MultiAssayExperiment containers and can serve as tools to practice the mia functionality.

Details

Currently, the following datasets are available:

- `dmn_se`: A SummarizedExperiment with 130 features and 278 samples
- `enterotype`: A TreeSummarizedExperiment with 553 features and 280 samples
- `esophagus`: A TreeSummarizedExperiment with 58 features and 3 samples
- `GlobalPatterns`: A TreeSummarizedExperiment with 19216 features and 26 samples
- `ibdmdb`: A compact `MultiAssayExperiment` demo dataset with 2 experiments (MGX + MTX)
- `HintikkaXOData`: A MultiAssayExperiment with 3 experiments (microbiota, metabolites and biomarkers)
- `peerj13075`: A TreeSummarizedExperiment with 674 features and 58 samples
- `Tengeler2020`: A TreeSummarizedExperiment with 151 features and 27 samples
- `Tito2024QMP`: A TreeSummarizedExperiment with 676 features and 589 samples

Examples

```
# Load dataset from mia
library(mia)
data("GlobalPatterns", package = "mia")

# In this case, the dataset is a TreeSE, so it is renamed as tse
tse <- GlobalPatterns

# Print summary
tse
```

peerj13075 *Skin microbial profiles 58 genetically unrelated individuals*

Description

peerj13075 includes skin microbial profiles of 58 volunteers with multiple factors. 16S r-RNA sequencing of V3-V4 regions was done to generate millions of read using illumina platform. A standard bioinformatic and statistical analysis done to explore skin bacterial diversity and its association with age, diet, geographical locations. The authors investigated significant association of skin microbiota with individual's geographical location.

Usage

```
data(peerj13075)
```

Format

A TreeSummarizedExperiment with 674 features and 58 samples. The rowData contains taxonomic information at kingdom, phylum, class, order, family and genus level. The colData includes:

Sample sample ID

Geographical_location city where participant lives (Ahmednagar, Pune and Nashik)

Gender participant's gender (Male or Female)

Age participant's age group (Middle_age, Adult and Elderly)

Diet participant's diet (Veg or Mixed)

Author(s)

Potbhare, R., et al.

References

Potbhare, R., RaviKumar, A., Munukka, E., Lahti, L., & Ashma, R. (2022). Skin microbiota diversity among genetically unrelated individuals of Indian origin. PeerJ, 10, e13075. https://doi.org/10.7717/peerj.13075 Supplemental information includes OTU table and taxonomy table publicly-accessible from: https://www.doi.org/10.7717/peerj.13075/supp-1 https://www.doi.org/10.7717/peerj.13075/supp-2

See Also

mia-datasets

rarefyAssay *Subsample Counts*

Description

rarefyAssay randomly subsamples counts within a SummarizedExperiment object and returns a new SummarizedExperiment containing the original assay and the new subsampled assay.

Usage

```
rarefyAssay(x, ...)

## S4 method for signature 'SummarizedExperiment'
rarefyAssay(
  x,
  assay.type = assay_name,
  assay_name = "counts",
  sample = min_size,
  min_size = min(colSums2(assay(x, assay.type))),
  replace = FALSE,
  name = "subsampled",
  ...
)
```

Arguments

x	TreeSummarizedExperiment.
...	optional arguments: • verbose: Logical scalar. Choose whether to show messages. (Default: TRUE)
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
sample	Integer scalar. Indicates the number of counts being simulated i.e. rarefying depth. This can equal to lowest number of total counts found in a sample or a user specified number.
min_size	Deprecated. Use sample instead.
replace	Logical scalar. Whether to åperform subsampling with replacement. Ths works similarly to sample(..., replace = TRUE). (Default: FALSE)
name	Character scalar. The name for the transformed assay to be stored. (Default: method)

Details

Although the subsampling approach is highly debated in microbiome research, we include the `rarefyAssay` function because there may be some instances where it can be useful. Note that the output of `rarefyAssay` is not the equivalent as the input and any result have to be verified with the original dataset.

Subsampling/Rarefying may undermine downstream analyses and have unintended consequences. Therefore, make sure this normalization is appropriate for your data.

To maintain the reproducibility, please define the seed using set.seed() before implement this function.

When `replace = FALSE`, the function uses internally `vegan::rarefy` while with replacement enabled the function utilizes own implementation, inspired by `phyloseq::rarefy_even_depth`.

Value

`rarefyAssay` return `x` with subsampled data.

References

McMurdie PJ, Holmes S. Waste not, want not: why rarefying microbiome data is inadmissible. PLoS computational biology. 2014 Apr 3;10(4):e1003531.

Gloor GB, Macklaim JM, Pawlowsky-Glahn V & Egozcue JJ (2017) Microbiome Datasets Are Compositional: And This Is Not Optional. Frontiers in Microbiology 8: 2224. doi: 10.3389/fmicb.2017.02224

Weiss S, Xu ZZ, Peddada S, Amir A, Bittinger K, Gonzalez A, Lozupone C, Zaneveld JR, Vázquez-Baeza Y, Birmingham A, Hyde ER. Normalization and microbial differential abundance strategies depend upon data characteristics. Microbiome. 2017 Dec;5(1):1-8.

See Also

- `vegan::rrarefy`
- `phyloseq::rarefy_even_depth`

Examples

```
# When samples in TreeSE are less than specified sample, they will be
# removed. If after subsampling features are not present in any of the
# samples, they will be removed.
data(GlobalPatterns)
tse <- GlobalPatterns
set.seed(123)
tse_subsampled <- rarefyAssay(tse, sample = 60000, name = "subsampled")
tse_subsampled
dim(tse)
dim(assay(tse_subsampled, "subsampled"))
```

splitOn *Split* TreeSummarizedExperiment *column-wise or row-wise based on grouping variable*

Description

Split TreeSummarizedExperiment column-wise or row-wise based on grouping variable

Usage

```
splitOn(x, ...)

unsplitOn(x, ...)

## S4 method for signature 'SummarizedExperiment'
splitOn(x, group = f, f = NULL, ...)

## S4 method for signature 'SingleCellExperiment'
splitOn(x, group = f, f = NULL, ...)

## S4 method for signature 'TreeSummarizedExperiment'
splitOn(
  x,
  group = f,
  f = NULL,
  update.tree = update_rowTree,
  update_rowTree = TRUE,
  ...
)

## S4 method for signature 'list'
unsplitOn(x, update.tree = update_rowTree, update_rowTree = TRUE, ...)

## S4 method for signature 'SimpleList'
unsplitOn(x, update.tree = update_rowTree, update_rowTree = TRUE, ...)

## S4 method for signature 'SingleCellExperiment'
unsplitOn(
  x,
  altexp = altExpNames,
  altExpNames = names(altExps(x)),
  keep.dimred = keep_reducedDims,
  keep_reducedDims = FALSE,
  ...
)
```

Arguments

x	TreeSummarizedExperiment.
...	Arguments passed to agglomerateByVariable function for SummarizedExperiment objects and other functions. See agglomerateByVariable for more details. • use.names: Logical scalar. Specifies whether to name elements of list by their group names. (Default: TRUE)
group	Character scalar, character vector or factor vector. A column name from rowData(x) or colData(x) or alternatively a vector specifying how the merging is performed. If vector, the value must be the same length as nrow(x)/ncol(x). Rows/Cols corresponding to the same level will be merged. If length(levels(group)) == nrow(x)/ncol(x), x will be returned unchanged. If group matches with both dimensions, by must be specified. (Default: NULL)
f	Deprecated. Use group instead.
update.tree	Logical scalar. Should rowTree() also be merged? (Default: TRUE)
update_rowTree	Deprecated. Use update.tree instead.
altexp	Character vector. Specify the alternative experiments to be unsplit. (Default: names(altExps(x)))
altExpNames	Deprecated. Use altexp instead.
keep.dimred	Logical scalar. Should the reducedDims(x) be transferred to the result? Please note, that this breaks the link between the data used to calculate the reduced dims. (Default: FALSE)
keep_reducedDims	Deprecated. Use keep.dimred instead.

Details

splitOn split data based on grouping variable. Splitting can be done column-wise or row-wise. The returned value is a list of SummarizedExperiment objects; each element containing members of each group.

Value

For splitOn: SummarizedExperiment objects in a SimpleList.

For unsplitOn: x, with rowData and assay data replaced by the unsplit data. colData of x is kept as well and any existing rowTree is dropped as well, since existing rowLinks are not valid anymore.

See Also

agglomerateByRanks agglomerateByVariable, aggregateAcrossFeatures, agglomerateByRank, altExps, splitAltExps

Examples

```
data(GlobalPatterns)
tse <- GlobalPatterns
# Split data based on SampleType.
se_list <- splitOn(tse, group = "SampleType")

# List of SE objects is returned.
se_list

# Create arbitrary groups
rowData(tse)$group <- sample(1:3, nrow(tse), replace = TRUE)
colData(tse)$group <- sample(1:3, ncol(tse), replace = TRUE)

# Split based on rows
# Each element is named based on their group name. If you don't want to name
# elements, use use_name = FALSE. Since "group" can be found from rowdata and
# colData you must use `by`.
se_list <- splitOn(tse, group = "group", use.names = FALSE, by = 1)

# When column names are shared between elements, you can store the list to
# altExps
altExps(tse) <- se_list

altExps(tse)

# If you want to split on columns and update rowTree, you can do
se_list <- splitOn(tse, group = colData(tse)$group, update.tree = TRUE)

# If you want to combine groups back together, you can use unsplitBy
unsplitOn(se_list)
```

---

summarizeDominance *Summarizing microbiome data*

Description

To query a SummarizedExperiment for interesting features, several functions are available.

Usage

```
summarizeDominance(x, group = NULL, name = "dominant_taxa", ...)

getUnique(x, ...)

getTop(
  x,
  top = 5L,
  method = c("mean", "sum", "median"),
```

```
  assay.type = assay_name,
  assay_name = "counts",
  na.rm = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
getTop(
  x,
  top = 5L,
  method = c("mean", "sum", "median", "prevalence"),
  assay.type = assay_name,
  assay_name = "counts",
  na.rm = TRUE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
getUnique(x, rank = NULL, ...)

## S4 method for signature 'SummarizedExperiment'
summarizeDominance(x, group = NULL, name = "dominant_taxa", ...)

## S4 method for signature 'SummarizedExperiment'
summary(object, assay.type = assay_name, assay_name = "counts")
```

Arguments

x	TreeSummarizedExperiment.
group	With group, it is possible to group the observations in an overview. Must be one of the column names of colData.
name	Character scalar. A name for the column of the colData where results will be stored. (Default: "dominant_taxa")
...	Additional arguments passed on to agglomerateByRank() when rank is specified for summarizeDominance.
top	Numeric scalar. Determines how many top taxa to return. Default is to return top five taxa. (Default: 5)
method	Character scalar. Specify the method to determine top taxa. Either sum, mean, median or prevalence. (Default: "mean")
assay.type	Character scalar. Specifies the name of the assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
na.rm	Logical scalar. Should NA values be omitted? (Default: TRUE)
rank	Character scalar. Defines a taxonomic rank. Must be a value of the output of taxonomyRanks(). (Default: NULl)
object	A SummarizedExperiment object.

Details

The getTop extracts the most top abundant "FeatureID"s in a SummarizedExperiment object.

The getUnique is a basic function to access different taxa at a particular taxonomic rank.

summarizeDominance returns information about most dominant taxa in a tibble. Information includes their absolute and relative abundances in whole data set.

The summary will return a summary of counts for all samples and features in SummarizedExperiment object.

Value

The getTop returns a vector of the most top abundant "FeatureID"s

The getUnique returns a vector of unique taxa present at a particular rank

The summarizeDominance returns an overview in a tibble. It contains dominant taxa in a column named *name* and its abundance in the data set.

The summary returns a list with two tibbles

See Also

getPrevalent

perCellQCMetrics, perFeatureQCMetrics, addPerCellQC, addPerFeatureQC, quickPerCellQC

Examples

```
data(GlobalPatterns)
top_taxa <- getTop(GlobalPatterns,
                       method = "mean",
                       top = 5,
                       assay.type = "counts")
top_taxa

# Use 'detection' to select detection threshold when using prevalence method
top_taxa <- getTop(GlobalPatterns,
                       method = "prevalence",
                       top = 5,
                       assay_name = "counts",
                       detection = 100)
top_taxa

# Top taxa os specific rank
getTop(agglomerateByRank(GlobalPatterns,
                             rank = "Genus",
                             na.rm = TRUE))

# Gets the overview of dominant taxa
dominant_taxa <- summarizeDominance(GlobalPatterns,
                                    rank = "Genus")
dominant_taxa
```

```
# With group, it is possible to group observations based on specified groups
# Gets the overview of dominant taxa
dominant_taxa <- summarizeDominance(GlobalPatterns,
                                    rank = "Genus",
                                    group = "SampleType",
                                    na.rm = TRUE)

dominant_taxa

# Get an overview of sample and taxa counts
summary(GlobalPatterns, assay.type= "counts")

# Get unique taxa at a particular taxonomic rank
# sort = TRUE means that output is sorted in alphabetical order
# With na.rm = TRUE, it is possible to remove NAs
# sort and na.rm can also be used in function getTop
getUnique(GlobalPatterns, "Phylum", sort = TRUE)
```

---

taxonomyRanks *Functions for accessing taxonomic data stored in* rowData.

Description

These function work on data present in rowData and define a way to represent taxonomic data alongside the features of a SummarizedExperiment.

Usage

```
taxonomyRanks(x)

taxonomyRankEmpty(
  x,
  rank = taxonomyRanks(x)[1L],
  empty.fields = c(NA, "", " ", "\t", "-", "_")
)

checkTaxonomy(x, ...)

getTaxonomyLabels(x, ...)

mapTaxonomy(x, ...)

## S4 method for signature 'SummarizedExperiment'
taxonomyRanks(x)

## S4 method for signature 'SummarizedExperiment'
taxonomyRankEmpty(
```

```
  x,
  rank = taxonomyRanks(x)[1],
  empty.fields = c(NA, "", " ", "\t", "-", "_")
)

## S4 method for signature 'SummarizedExperiment'
checkTaxonomy(x)

setTaxonomyRanks(ranks)

getTaxonomyRanks()

## S4 method for signature 'SummarizedExperiment'
getTaxonomyLabels(
  x,
  empty.fields = c(NA, "", " ", "\t", "-", "_"),
  with.rank = with_rank,
  with_rank = FALSE,
  make.unique = make_unique,
  make_unique = TRUE,
  resolve.loops = resolve_loops,
  resolve_loops = FALSE,
  ...
)

## S4 method for signature 'SummarizedExperiment'
mapTaxonomy(
  x,
  taxa = NULL,
  from = NULL,
  to = NULL,
  use.grepl = use_grepl,
  use_grepl = FALSE
)

IdTaxaToDataFrame(from)
```

Arguments

x	TreeSummarizedExperiment.
rank	Character scalar. Defines a taxonomic rank. Must be a value of taxonomyRanks() function.
empty.fields	Character vector. Defines which values should be regarded as empty. (Default: c(NA, "", " ", "\t")). They will be removed if na.rm = TRUE before agglomeration.
...	additional arguments • lowest.rank: A lowest taxonomy level to be considered in getTaxonomyLabels. Ranks lower than this will be collapsed into rank specified by lowest.rank.

For example, if genus level is specified, species will be collapsed into genus. If NULL, the data is not collapsed. (Default: NULL)

ranks — Character vector. A vector of ranks to be set.

with.rank — Logical scalar. Should the level be add as a suffix? For example: "Phylum:Crenarchaeota". (Default: FALSE)

with_rank — Deprecated. Use with.rank instead.

make.unique — Logical scalar. Should the labels be made unique, if there are any duplicates? (Default: TRUE)

make_unique — Deprecated. Use make.unique instead.

resolve.loops — Logical scalar. Should resolveLoops be applied to the taxonomic data? Please note that has only an effect, if the data is unique. (Default: TRUE)

resolve_loops — Deprecated. Use resolve.loops instead.

taxa — Character vector. Used for subsetting the taxonomic information. If no information is found,NULL is returned for the individual element. (Default: NULL)

from
- For mapTaxonomy: character scalar. A value which must be a valid taxonomic rank. (Default: NULL)
- otherwise a Taxa object as returned by IdTaxa

to — Character Scalar. Must be a valid taxonomic rank. (Default: NULL)

use.grepl — Logical. Should pattern matching via grepl be used? Otherwise literal matching is used. (Default: FALSE)

use_grepl — Deprecated. Use use.grepl instead.

Details

taxonomyRanks returns, which columns of rowData(x) are regarded as columns containing taxonomic information.

taxonomyRankEmpty checks, if a selected rank is empty of information.

checkTaxonomy checks, if taxonomy information is valid and whether it contains any problems. This is a soft test, which reports some diagnostic and might mature into a data validator used upon object creation.

getTaxonomyLabels generates a character vector per row consisting of the lowest taxonomic information possible. If data from different levels, is to be mixed, the taxonomic level is prepended by default.

IdTaxaToDataFrame extracts taxonomic results from results of IdTaxa.

mapTaxonomy maps the given features (taxonomic groups; taxa) to the specified taxonomic level (to argument) in rowData of the SummarizedExperiment data object (i.e. rowData(x)[,taxonomyRanks(x)]). If the argument to is not provided, then all matching taxonomy rows in rowData will be returned. This function allows handy conversions between different

Taxonomic information from the IdTaxa function of DECIPHER package are returned as a special class. With as(taxa,"DataFrame") the information can be easily converted to a DataFrame compatible with storing the taxonomic information a rowData. Please note that the assigned confidence information are returned as metatdata and can be accessed using metadata(df)$confidence.

Value

- taxonomyRanks: a character vector with all the taxonomic ranks found in colnames(rowData(x))
- taxonomyRankEmpty: a logical value
- mapTaxonomy: a list per element of taxa. Each element is either a DataFrame, a character or NULL. If all character results have the length of one, a single character vector is returned.

See Also

agglomerateByRank, toTree, resolveLoop

Examples

```
data(GlobalPatterns)
GlobalPatterns
taxonomyRanks(GlobalPatterns)

checkTaxonomy(GlobalPatterns)

table(taxonomyRankEmpty(GlobalPatterns,"Kingdom"))
table(taxonomyRankEmpty(GlobalPatterns,"Species"))

getTaxonomyLabels(GlobalPatterns[1:20,])
# Taxonomy labels represent the lowest taxonomy name that identifies each
# taxa. For instance, they can represent OTUs which does no necessarily
# tell much. In this case, you might want to get the labels with higher
# taxonomy rank
getTaxonomyLabels(GlobalPatterns[1:20,], lowest.rank = "Class")

# mapTaxonomy
## returns the unique taxonomic information
mapTaxonomy(GlobalPatterns)
# returns specific unique taxonomic information
mapTaxonomy(GlobalPatterns, taxa = "Escherichia")
# returns information on a single output
mapTaxonomy(GlobalPatterns, taxa = "Escherichia",to="Family")

# setTaxonomyRanks
tse <- GlobalPatterns
colnames(rowData(tse))[1] <- "TAXA1"

setTaxonomyRanks(colnames(rowData(tse)))
# Taxonomy ranks set to: taxa1 phylum class order family genus species

# getTaxonomyRanks is to get/check if the taxonomic ranks is set to "TAXA1"
getTaxonomyRanks()
```

Tengeler2020 *Gut microbiota profiles of 27 individuals with ADHD and healthy controls*

Description

Tengeler2020 includes gut microbiota profiles of 27 persons with ADHD. A standard bioinformatic and statistical analysis done to demonstrate that altered microbial composition could be a driver of altered brain structure and function and concomitant changes in the animals' behavior. This was investigated by colonizing young, male, germ-free C57BL/6JOlaHsd mice with microbiota from individuals with and without ADHD.

Usage

```
data(Tengeler2020)
```

Format

A TreeSummarizedExperiment with 151 features and 27 samples. The rowData contains taxonomic information at Kingdom, Phylum, Class, Order, Family and Genus level. The colData includes:

patient_status clinical status of the patient (ADHD or Control)

cohort cohort to which the patient belongs (Cohort_1, Cohort_2 and Cohort_3)

patient_status_vs_cohort combination of patient_status and cohort

sample_name unique sample ID

Author(s)

A.C. Tengeler, et al.

References

Tengeler, A.C., Dam, S.A., Wiesmann, M. et al. Gut microbiota from persons with attention-deficit/hyperactivity disorder affects the brain in mice. Microbiome 8, 44 (2020). https://doi.org/10.1186/s40168-020-00816-x

Supplemental information includes Home-cage activity, methods, results and imaging parameters and publicly-accessible from: https://static-content.springer.com/esm/art%3A10.1186%2Fs40168-020-00816-x/MediaObjects/40168_2020_816_MOESM1_ESM.docx https://static-content.springer.com/esm/art%3A10.1186%2Fs40168-020-00816-x/MediaObjects/40168_2020_816_MOESM2_ESM.docx https://static-content.springer.com/esm/art%3A10.1186%2Fs40168-020-00816-x/MediaObjects/40168_2020_816_MOESM3_ESM.docx

See Also

mia-datasets

Tito2024QMP *Fecal microbiota samples from 589 patients across different colorectal cancer stages*

Description

The study combined Quantitative Microbiome Profiling (QMP) with extensive patient phenotyping from a group of 589 colorectal cancer (CRC) patients, advanced adenoma (AA) patients, and healthy controls. By implementing confounder control and quantitative profiling methods, the study was able to reveal potential misleading associations between microbial markers and colorectal cancer development that were driven by other factors like intestinal inflammation, rather than the cancer diagnosis itself.

Usage

```
data(Tito2024QMP)
```

Format

A TreeSummarizedExperiment with 676 features and 589 samples. The rowData contains species. The colData includes:

sampleID (character) Sample ID from the corresponding study

diagnosis (factor) Diagnosis type, with possible values: "ADE" (advanced adenoma), "CRC" (colorectal cancer), "CTL" (control)

colonoscopy (factor) Colonoscopy result, with possible values: "FIT_Positive", "familial_risk_familial_CRC_FCC", "familial_risk_no", "abdomil_complaints"

Author(s)

Shadman Ishraq

References

Raúl Y. Tito, Sara Verbandt, Marta Aguirre Vazquez, Leo Lahti, Chloe Verspecht, Verónica Lloréns-Rico, Sara Vieira-Silva, Janine Arts, Gwen Falony, Evelien Dekker, Joke Reumers, Sabine Tejpar & Jeroen Raes (2024). Microbiome confounders and quantitative profiling challenge predicted microbial targets in colorectal cancer development. Nature Medicine,30, 1339-1348. https://doi.org/10.1038/s41591-024-02963-2

See Also

mia-datasets

transformAssay *Transform assay*

Description

Variety of transformations for abundance data, stored in assay. See details for options.

Usage

```
transformAssay(x, ...)

## S4 method for signature 'SummarizedExperiment'
transformAssay(
  x,
  method,
  assay.type = "counts",
  assay_name = NULL,
  MARGIN = "samples",
  name = method,
  pseudocount = FALSE,
  ...
)

## S4 method for signature 'SingleCellExperiment'
transformAssay(x, altexp = NULL, ...)
```

Arguments

x	TreeSummarizedExperiment.
...	additional arguments passed e.g. on to vegan:decostand or philr::philr.

- reference: Character scalar. Used to to fill reference sample's column in returned assay when calculating alr. (Default: NA)
- ref_vals Deprecated. Use reference instead.
- nbins: Numeric scalar. For "binning", specifies the number of bins to use. (Default: 4)
- percentile: Numeric scalar or NULL (css). Used to set the percentile value that calculates the scaling factors in the css normalization. If NULL, percentile is estimated from the data by calculating the portion of samples that exceed the threshold. (Default: NULL)
- scaling: Numeric scalar. Adjusts the normalization scale by dividing the calculated scaling factors, effectively changing the magnitude of the normalized counts. (Default: 1000).
- threshold: Numeric scalar. For "css", specifies relative difference threshold and determines the first point where the relative change in differences between consecutive quantiles exceeds this threshold. (Default: 0.1) For

"cutoff", values less than or equal to the threshold are replaced with value. (Default: 0)

- value: Numeric scalar. For "cutoff", specifies the replacement value for counts less than or equal to the threshold. (Default: NA)
- tree: phylo. Phylogeny used in PhILR transformation. If NULL, the tree is retrieved from x. (Default: NULL).
- node.labels: Character vector. Linkages between tree and x. Used in PhILR transformation. (Default: NULL).

method	Character scalar. Specifies the transformation method.
assay.type	Character scalar. Specifies the name of assay used in calculation. (Default: "counts")
assay_name	Deprecated. Use assay.type instead.
MARGIN	Character scalar. Determines whether the transformation is applied sample (column) or feature (row) wise. (Default: "samples")
name	Character scalar. The name for the transformed assay to be stored. (Default: method)
pseudocount	Logical scalar or numeric scalar. When TRUE, automatically adds half of the minimum positive value of assay.type (missing values ignored by default: na.rm = TRUE). When FALSE, does not add any pseudocount (pseudocount = 0). Alternatively, a user-specified numeric value can be added as pseudocount. (Default: FALSE).
altexp	Character vector or NULL. Specifies the names of alternative experiments to which the transformation should also be applied. If NULL, the transformation is only applied to the main experiment. (Default: NULL).

Details

transformAssay function provides a variety of options for transforming abundance data. The transformed data is calculated and stored in a new assay.

The transformAssay provides sample-wise (column-wise) or feature-wise (row-wise) transformation to the abundance table (assay) based on specified MARGIN.

The available transformation methods include:

- 'alr', 'chi.square', 'clr', 'frequency', 'hellinger', 'log', 'normalize', 'pa', 'rank', 'rclr', 'relabundance', 'rrank', 'standardize', 'total': please refer to decostand for details.
- 'philr': please refer to philr for details.
- 'css': Cumulative Sum Scaling (CSS) can be used to normalize count data by accounting for differences in library sizes. By default, the function determines the normalization percentile for summing and scaling counts. If you want to specify the percentile value, good default value might be 0.5. The method is inspired by the CSS methods in metagenomeSeq package.
- 'log10': log10 transformation can be used for reducing the skewness of the data.

$$log10 = \log_{10} x$$

where x is a single value of data.

- 'log2': log2 transformation can be used for reducing the skewness of the data.

$$log2 = \log_2 x$$

where x is a single value of data.

- 'difference': Pairwise differences between features. Calculates $x - y$ for all unique feature pairs across samples, where x and y are entries of the specified assay.type.
- 'division': Pairwise ratios between features. Calculates x/y for all unique feature pairs across samples, where x and y are entries of the specified assay.type.
- 'invnorm': Inverse rank normalisation. Ranks values per sample/feature and maps them to standard normal quantiles.

$$z = \Phi^{-1}\left(\frac{r - \text{offset}}{n + 1 - 2\,\text{offset}}\right)$$

Controlled by `offset` (default `0.5`; also `0.375`=Blom, `0`=van der Waerden) and `ties.method` (passed to `base::rank`, default `"average"`).

- 'pseudocount': Adds only pseudocount.
- 'cutoff': In some ecological studies, only strictly positive values are taken into account. This method keeps only values greater than `threshold` and replaces all other values with `value`.
- 'binning': Binning of the abundance values into a specified number of bins. The non-zero values are sorted and divided into `nbins` groups of equal size (quantiles). The group with the highest abundances is assigned to bin `nbins`, while the group with the lowest non-zero abundances is assigned to bin 1. Zero values are assigned to bin 0. This binning approach is based on the binning strategy described by Medearis et al. (2026).

Value

`transformAssay` returns the input object `x`, with a new transformed abundance table named `name` added in the `assays`.

References

Paulson, J., Stine, O., Bravo, H. et al. (2013) Differential abundance analysis for microbial marker-gene surveys *Nature Methods* 10, 1200–1202. doi:10.1038/nmeth.2658

Medearis, N. A., Zhu, S., & Zomorrodi, A. R. (2026). BiomeGPT: A foundation model for the human gut microbiome *bioRxiv* doi:10.64898/2026.01.05.697599

See Also

- `vegan::decostand`
- `philr::philr`

Examples

```
data(GlobalPatterns)
tse <- GlobalPatterns
```

```
# By specifying 'method', it is possible to apply different transformations,
# e.g. compositional transformation.
tse <- transformAssay(tse, method = "relabundance")

# The target of transformation can be specified with "assay.type"
# Pseudocount can be added by specifying 'pseudocount'.

# Perform CLR with half of the smallest positive value as pseudocount
tse <- transformAssay(
    tse, assay.type = "counts", method = "clr",
    pseudocount = TRUE
    )

head(assay(tse, "clr"))

# Perform CSS normalization.
tse <- transformAssay(tse, method = "css")
head(assay(tse, "css"))

# With MARGIN, you can specify the if transformation is done for samples or
# for features. Here Z-transformation is done feature-wise.
tse <- transformAssay(tse, method = "standardize", MARGIN = "features")
head(assay(tse, "standardize"))

# Name of the stored table can be specified.
tse <- transformAssay(tse, method="hellinger", name="test")
head(assay(tse, "test"))

# pa returns presence absence table.
tse <- transformAssay(tse, method = "pa")
head(assay(tse, "pa"))

# rank returns ranks of taxa.
tse <- transformAssay(tse, method = "rank")
head(assay(tse, "rank"))

# In order to use other ranking variants, modify the chosen assay directly:
assay(tse, "rank_average", withDimnames = FALSE) <- colRanks(
    assay(tse, "counts"), ties.method = "average", preserveShape = TRUE)

# Using altexp parameter. First agglomerate the data and then apply
# transformation.
tse <- GlobalPatterns
tse <- agglomerateByRanks(tse)
tse <- transformAssay(
    tse, method = "relabundance", altexp = altExpNames(tse))
# The transformation is applied to all alternative experiments
altExp(tse, "Species")

## Not run:
# philr transformation can be applied if the philr package is installed.
# Subset data b taking only prevalent taxa
tse <- subsetByPrevalent(tse)
```

```
# Apply transformation
tse <- transformAssay(tse, method = "philr", pseudocount = 1, MARGIN = 1L)
# The transformed data is added to altExp
altExp(tse, "philr")

## End(Not run)
```

Index

∗ **datasets**
dmn_se, 48
enterotype, 49
esophagus, 50
GlobalPatterns, 93
HintikkaXOData, 94
ibdmdb, 95
mia-datasets, 109
peerj13075, 110
Tengeler2020, 122
Tito2024QMP, 123
?agglomerateByRank, *86*

addAbundanceClass *(*getAbundant*)*, 51
addAbundanceClass,SingleCellExperiment-method *(*getAbundant*)*, 51
addAbundanceClass,SummarizedExperiment-method *(*getAbundant*)*, 51
addAlpha, 5, *19*
addAlpha,SummarizedExperiment-method *(*addAlpha*)*, 5
addCCA, *83*
addCCA *(*getCCA*)*, 55
addCCA,SingleCellExperiment-method *(*getCCA*)*, 55
addCluster *(*getCluster*)*, 59
addCluster,SingleCellExperiment-method *(*getCluster*)*, 59
addCluster,SummarizedExperiment-method *(*getCluster*)*, 59
addContaminantQC, 12
addContaminantQC,SummarizedExperiment-method *(*addContaminantQC*)*, 12
addDissimilarity, 14, *19*
addDissimilarity,SummarizedExperiment-method *(*addDissimilarity*)*, 14
addDivergence, 18
addDivergence,SummarizedExperiment-method *(*addDivergence*)*, 18
addDominant *(*getDominant*)*, 68
addDominant,SummarizedExperiment-method *(*getDominant*)*, 68
addDPCoA *(*getDPCoA*)*, 70
addHierarchyTree, *103*
addHierarchyTree *(*getHierarchyTree*)*, 72
addHierarchyTree,SummarizedExperiment-method *(*getHierarchyTree*)*, 72
addJointRPCA *(*getRPCA*)*, 90
addJointRPCA,MultiAssayExperiment-method *(*getRPCA*)*, 90
addJointRPCA,SingleCellExperiment-method *(*getRPCA*)*, 90
addLDA, *89*
addLDA *(*getLDA*)*, 73
addLDA,SummarizedExperiment-method *(*getLDA*)*, 73
addMDS *(*getMDS*)*, 75
addMDS,SingleCellExperiment-method *(*getMDS*)*, 75
addMediation, 20
addMediation,SummarizedExperiment-method *(*addMediation*)*, 20
addNMDS *(*getNMDS*)*, 76
addNMF, *89*
addNMF *(*getNMF*)*, 79
addNMF,SummarizedExperiment-method *(*getNMF*)*, 79
addNotContaminantQC *(*addContaminantQC*)*, 12
addNotContaminantQC,SummarizedExperiment-method *(*addContaminantQC*)*, 12
addPerCellQC, *117*
addPerFeatureQC, *117*
addPERMANOVA *(*getPERMANOVA*)*, 81
addPERMANOVA,SummarizedExperiment-method *(*getPERMANOVA*)*, 81
addPerSampleDominantFeatures *(*deprecate*)*, 40
addPerSampleDominantFeatures,SummarizedExperiment-method

(deprecate), 40
addPerSampleDominantTaxa (deprecate), 40
addPerSampleDominantTaxa,SummarizedExperiment-method (deprecate), 40
addPrevalence (getPrevalence), 83
addPrevalence,SummarizedExperiment-method (getPrevalence), 83
addPrevalentAbundance (getPrevalence), 83
addPrevalentAbundance,SummarizedExperiment-method (getPrevalence), 83
addRDA, *83*
addRDA (getCCA), 55
addRDA,SingleCellExperiment-method (getCCA), 55
addRPCA (getRPCA), 90
addRPCA,SummarizedExperiment-method (getRPCA), 90
addTaxonomyTree (deprecate), 40
addTaxonomyTree,SummarizedExperiment-method (deprecate), 40
agglomerate-methods (agglomerateByRank), 26
agglomerateByModule (agglomerateByRank), 26
agglomerateByModule,SummarizedExperiment-method (agglomerateByRank), 26
agglomerateByPrevalence, 24
agglomerateByPrevalence,SummarizedExperiment-method (agglomerateByPrevalence), 24
agglomerateByPrevalence,TreeSummarizedExperiment-method (agglomerateByPrevalence), 24
agglomerateByRank, *25*, 26, *28*, *30*, *86, 87*, *114*, *121*
agglomerateByRank,SingleCellExperiment-method (agglomerateByRank), 26
agglomerateByRank,SummarizedExperiment-method (agglomerateByRank), 26
agglomerateByRank,TreeSummarizedExperiment-method (agglomerateByRank), 26
agglomerateByRanks, *114*
agglomerateByRanks (agglomerateByRank), 26
agglomerateByRanks,SingleCellExperiment-method (agglomerateByRank), 26
agglomerateByRanks,SummarizedExperiment-method (agglomerateByRank), 26
agglomerateByRanks,TreeSummarizedExperiment-method (agglomerateByRank), 26
agglomerateByVariable, *30*, *114*
agglomerateByVariable (agglomerateByRank), 26
agglomerateByVariable,SummarizedExperiment-method (agglomerateByRank), 26
agglomerateByVariable,TreeSummarizedExperiment-method (agglomerateByRank), 26
aggregateAcrossFeatures, *114*
aggregateAcrossGenes, *29*
altExps, *30*, *114*
ape::phylo, *102*
assay, *26*
assays, *126*
avgdist, *15*

bestDMNFit (calculateDMN), 33
bestDMNFit,SummarizedExperiment-method (calculateDMN), 33
BiocParallelParam, *6*, *16*, *35*
biom, *38*
Biostrings::DNAStringSet, *102*
bluster, *61*
BlusterParam, *61*

calculateCCA (getCCA), 55
calculateDMN, 32, *49*
calculateDMN,ANY-method (calculateDMN), 33
calculateDMN,SummarizedExperiment-method (calculateDMN), 33
calculateDMNgroup (calculateDMN), 33
calculateDMNgroup,ANY-method (calculateDMN), 33
calculateDMNgroup,SummarizedExperiment-method (calculateDMN), 33
calculateDPCoA (getDPCoA), 70
calculateJSD (deprecate), 40
calculateJSD,ANY-method (deprecate), 40
calculateNMDS (getNMDS), 76
calculateOverlap (deprecate), 40
calculateOverlap,ANY-method (deprecate), 40
calculateRDA (getCCA), 55
calculateUnifrac (deprecate), 40
calculateUnifrac,ANY-method (deprecate), 40
cca, *57*
checkTaxonomy (taxonomyRanks), 118

checkTaxonomy,SummarizedExperiment-method
(taxonomyRanks), 118
cluster (deprecate), 40
cluster,SummarizedExperiment-method
(deprecate), 40
colData, *19*, *69*
convertFromBIOM, *38*, *97–99*, *102*, *103*
convertFromBIOM (convertToBIOM), 37
convertFromDADA2, 36, *38*, *97–99*, *102*
convertFromPhyloseq, *38*, *97–99*, *102*
convertFromPhyloseq
(convertToPhyloseq), 39
convertToBIOM, 37
convertToBIOM,SummarizedExperiment-method
(convertToBIOM), 37
convertToPhyloseq, 39
convertToPhyloseq,SummarizedExperiment-method
(convertToPhyloseq), 39
convertToPhyloseq,TreeSummarizedExperiment-method
(convertToPhyloseq), 39
cophenetic.phylo, *71*
countDominantFeatures (deprecate), 40
countDominantFeatures,SummarizedExperiment-method
(deprecate), 40
countDominantTaxa (deprecate), 40
countDominantTaxa,SummarizedExperiment-method
(deprecate), 40

dbrda, *57*
decontam:isContaminant, *13*, *14*
decontam:isNotContaminant, *13*, *14*
decostand, *125*
deprecate, 40
DirichletMultinomial, *33*
dist, *78*
diversity, *11*
dmn, *34*, *35*
dmn_se, 48, *109*
DMNGroup, *35*
dmngroup, *35*

ecodive:unweighted_unifrac(), *16*
ecodive:weighted_unifrac(), *16*
enterotype, 49, *109*
esophagus, 50, *109*
estimateDivergence (deprecate), 40
estimateDivergence,SummarizedExperiment-method
(deprecate), 40
estimateDiversity (deprecate), 40
estimateDiversity,ANY-method
(deprecate), 40
estimateDominance (deprecate), 40
estimateDominance,ANY-method
(deprecate), 40
estimateEvenness (deprecate), 40
estimateEvenness,ANY-method
(deprecate), 40
estimateFaith (deprecate), 40
estimateFaith,ANY-method (deprecate), 40
estimateR, *10*, *11*
estimateRichness (deprecate), 40
estimateRichness,ANY-method
(deprecate), 40

full_join, *108*
full_join (deprecate), 40
full_join,ANY-method (deprecate), 40

getAbundanceClass (getAbundant), 51
getAbundanceClass,ANY-method
(getAbundant), 51
getAbundanceClass,SingleCellExperiment-method
(getAbundant), 51
getAbundanceClass,SummarizedExperiment-method
(getAbundant), 51
getAbundant, 51
getAbundant,ANY-method (getAbundant), 51
getAbundant,SingleCellExperiment-method
(getAbundant), 51
getAbundant,SummarizedExperiment-method
(getAbundant), 51
getAlpha (addAlpha), 5
getAlpha,SummarizedExperiment-method
(addAlpha), 5
getBestDMNFit (calculateDMN), 33
getBestDMNFit,SummarizedExperiment-method
(calculateDMN), 33
getCCA, 55
getCCA,ANY-method (getCCA), 55
getCCA,SummarizedExperiment-method
(getCCA), 55
getCluster, 59
getCluster,ANY-method (getCluster), 59
getCluster,SingleCellExperiment-method
(getCluster), 59
getCluster,SummarizedExperiment-method
(getCluster), 59

getConditionallyLowAbundant (getAbundant), 51
getConditionallyLowAbundant,ANY-method (getAbundant), 51
getConditionallyLowAbundant,SingleCellExperiment-method (getAbundant), 51
getConditionallyLowAbundant,SummarizedExperiment-method (getAbundant), 51
getCrossAssociation, 62
getCrossAssociation,MultiAssayExperiment-method (getCrossAssociation), 62
getCrossAssociation,SummarizedExperiment-method (getCrossAssociation), 62
getDissimilarity, 76
getDissimilarity (addDissimilarity), 14
getDissimilarity,ANY-method (addDissimilarity), 14
getDissimilarity,SummarizedExperiment-method (addDissimilarity), 14
getDissimilarity,TreeSummarizedExperiment-method (addDissimilarity), 14
getDivergence (addDivergence), 18
getDivergence,SummarizedExperiment-method (addDivergence), 18
getDMN (calculateDMN), 33
getDMN,SummarizedExperiment-method (calculateDMN), 33
getDominant, 68
getDominant,SummarizedExperiment-method (getDominant), 68
getDPCoA, 70
getDPCoA,ANY,ANY-method (getDPCoA), 70
getDPCoA,TreeSummarizedExperiment,missing-method (getDPCoA), 70
getExperimentCrossAssociation (deprecate), 40
getExperimentCrossAssociation,MultiAssayExperiment-method (deprecate), 40
getExperimentCrossAssociation,SummarizedExperiment-method (deprecate), 40
getExperimentCrossCorrelation (deprecate), 40
getExperimentCrossCorrelation,ANY-method (deprecate), 40
getHierarchyTree, 72
getHierarchyTree,SummarizedExperiment-method (getHierarchyTree), 72
getJointRPCA (getRPCA), 90
getJointRPCA,MultiAssayExperiment-method (getRPCA), 90
getJointRPCA,SingleCellExperiment-method (getRPCA), 90
getLDA, 73
getLDA,SummarizedExperiment-method (getLDA), 73
getLowAbundant (getAbundant), 51
getLowAbundant,ANY-method (getAbundant), 51
getLowAbundant,SingleCellExperiment-method (getAbundant), 51
getLowAbundant,SummarizedExperiment-method (getAbundant), 51
getMDS, 75
getMDS,SingleCellExperiment-method (getMDS), 75
getMDS,TreeSummarizedExperiment-method (getMDS), 75
getMediation (addMediation), 20
getMediation,SummarizedExperiment-method (addMediation), 20
getNMDS, 76
getNMDS,ANY-method (getNMDS), 76
getNMDS,SingleCellExperiment-method (getNMDS), 76
getNMDS,SummarizedExperiment-method (getNMDS), 76
getNMF, 79
getNMF,SummarizedExperiment-method (getNMF), 79
getPermanentlyLowAbundant (getAbundant), 51
getPermanentlyLowAbundant,ANY-method (getAbundant), 51
getPermanentlyLowAbundant,SingleCellExperiment-method (getAbundant), 51
getPermanentlyLowAbundant,SummarizedExperiment-method (getAbundant), 51
getPERMANOVA, 81
getPERMANOVA,ANY-method (getPERMANOVA), 81
getPERMANOVA,SingleCellExperiment-method (getPERMANOVA), 81
getPERMANOVA,SummarizedExperiment-method (getPERMANOVA), 81
getPrevalence, 83
getPrevalence,ANY-method

(getPrevalence), 83
getPrevalence,SummarizedExperiment-method (getPrevalence), 83
getPrevalent, *51*, *54*, *117*
getPrevalent (getPrevalence), 83
getPrevalent,ANY-method (getPrevalence), 83
getPrevalent,SummarizedExperiment-method (getPrevalence), 83
getPrevalentAbundance (getPrevalence), 83
getPrevalentAbundance,ANY-method (getPrevalence), 83
getPrevalentAbundance,SummarizedExperiment-method (getPrevalence), 83
getPrevalentFeatures (deprecate), 40
getPrevalentFeatures,ANY-method (deprecate), 40
getPrevalentFeatures,SummarizedExperiment-method (deprecate), 40
getPrevalentTaxa (deprecate), 40
getPrevalentTaxa,ANY-method (deprecate), 40
getPrevalentTaxa,SummarizedExperiment-method (deprecate), 40
getRare, *51*, *54*
getRare (getPrevalence), 83
getRare,ANY-method (getPrevalence), 83
getRare,SummarizedExperiment-method (getPrevalence), 83
getRareFeatures (deprecate), 40
getRareFeatures,ANY-method (deprecate), 40
getRareFeatures,SummarizedExperiment-method (deprecate), 40
getRareTaxa (deprecate), 40
getRareTaxa,ANY-method (deprecate), 40
getRareTaxa,SummarizedExperiment-method (deprecate), 40
getRDA (getCCA), 55
getRDA,ANY-method (getCCA), 55
getRDA,SummarizedExperiment-method (getCCA), 55
getReducedDimAttribute, 89
getReducedDimAttribute,SingleCellExperiment-method (getReducedDimAttribute), 89
getRPCA, 90
getRPCA,SingleCellExperiment-method (getRPCA), 90
getRPCA,SummarizedExperiment-method (getRPCA), 90
getTaxonomyLabels (taxonomyRanks), 118
getTaxonomyLabels,SummarizedExperiment-method (taxonomyRanks), 118
getTaxonomyRanks (taxonomyRanks), 118
getTop, *87*
getTop (summarizeDominance), 115
getTop,SummarizedExperiment-method (summarizeDominance), 115
getTopFeatures (deprecate), 40
getTopFeatures,SummarizedExperiment-method (deprecate), 40
getTopTaxa (deprecate), 40
getTopTaxa,SummarizedExperiment-method (deprecate), 40
getUnique (summarizeDominance), 115
getUnique,SummarizedExperiment-method (summarizeDominance), 115
getUniqueFeatures (deprecate), 40
getUniqueFeatures,SummarizedExperiment-method (deprecate), 40
getUniqueTaxa (deprecate), 40
getUniqueTaxa,SummarizedExperiment-method (deprecate), 40
GlobalPatterns, 93, *109*

hierarchy-tree (getHierarchyTree), 72
HintikkaXOData, 94, *109*

ibdmdb, 95, *109*
IdTaxa, *120*
IdTaxaToDataFrame (taxonomyRanks), 118
importBIOM, *103*
importBIOM (convertToBIOM), 37
importHUMAnN, *38*, 96
importMetaPhlAn, *38*, *96*, *97*, 97
importMothur, *38*, *97*, *98*, 98, *102*
importQIIME2, *38*, *97–99*, 100
importQZA (importQIIME2), 100
importTaxpasta, 103
inner_join, *108*
inner_join (deprecate), 40
inner_join,ANY-method (deprecate), 40
isContaminant (addContaminantQC), 12
isContaminant,SummarizedExperiment-method (addContaminantQC), 12

isNotContaminant,SummarizedExperiment-method
(addContaminantQC), 12

LDA, *74*
left_join, *108*
left_join (deprecate), 40
left_join,ANY-method (deprecate), 40
loadFromBiom (deprecate), 40
loadFromHumann (deprecate), 40
loadFromMetaphlan (deprecate), 40
loadFromMothur (deprecate), 40
loadFromQIIME2 (deprecate), 40

makePhyloseqFromTreeSE (deprecate), 40
makePhyloseqFromTreeSE,SummarizedExperiment-method
(deprecate), 40
makePhyloseqFromTreeSE,TreeSummarizedExperiment-method
(deprecate), 40
makePhyloseqFromTreeSummarizedExperiment
(deprecate), 40
makePhyloseqFromTreeSummarizedExperiment,ANY-method
(deprecate), 40
makeTreeSEFromBiom (deprecate), 40
makeTreeSEFromDADA2 (deprecate), 40
makeTreeSEFromPhyloseq (deprecate), 40
makeTreeSummarizedExperimentFromBiom
(deprecate), 40
makeTreeSummarizedExperimentFromDADA2
(deprecate), 40
makeTreeSummarizedExperimentFromPhyloseq
(deprecate), 40
makeTreeSummarizedExperimentFromPhyloseq,ANY-method
(deprecate), 40
mapTaxonomy (taxonomyRanks), 118
mapTaxonomy,SummarizedExperiment-method
(taxonomyRanks), 118
MASS::isoMDS, *78, 79*
mediate, *20–22*
meltAssay (deprecate), 40
meltAssay,SummarizedExperiment-method
(deprecate), 40
meltSE, 104
meltSE,SingleCellExperiment-method
(meltSE), 104
meltSE,SummarizedExperiment-method
(meltSE), 104
mergeCols (deprecate), 40
mergeCols,SummarizedExperiment-method
(deprecate), 40
mergeCols,TreeSummarizedExperiment-method
(deprecate), 40
mergeFeatures (deprecate), 40
mergeFeatures,SummarizedExperiment-method
(deprecate), 40
mergeFeatures,TreeSummarizedExperiment-method
(deprecate), 40
mergeFeaturesByPrevalence (deprecate),
40
mergeFeaturesByPrevalence,SummarizedExperiment-method
(deprecate), 40
mergeFeaturesByRank (deprecate), 40
mergeFeaturesByRank,SingleCellExperiment-method
(deprecate), 40
mergeFeaturesByRank,SummarizedExperiment-method
(deprecate), 40
mergeFeaturesByRank,TreeSummarizedExperiment-method
(deprecate), 40
mergeRows (deprecate), 40
mergeRows,SummarizedExperiment-method
(deprecate), 40
mergeRows,TreeSummarizedExperiment-method
(deprecate), 40
mergeSamples (deprecate), 40
mergeSamples,SummarizedExperiment-method
(deprecate), 40
mergeSamples,TreeSummarizedExperiment-method
(deprecate), 40
mergeSEs, 106
mergeSEs,list-method (mergeSEs), 106
mergeSEs,SimpleList-method (mergeSEs),
106
mergeSEs,SummarizedExperiment-method
(mergeSEs), 106
metadata, *34*, *61*
metagenomeSeq, *125*
mia (mia-package), 4
mia-datasets, 109
mia-package, 4
MultiAssayExperiment, *63*, *95*

name, *9*

peerj13075, *109*, 110
perCellQCMetrics, *117*
perFeatureQCMetrics, *117*
performDMNgroupCV (calculateDMN), 33
performDMNgroupCV,ANY-method
(calculateDMN), 33

performDMNgroupCV,SummarizedExperiment-method (calculateDMN), 33
perSampleDominantFeatures (deprecate), 40
perSampleDominantFeatures,SummarizedExperiment-method (deprecate), 40
perSampleDominantTaxa (deprecate), 40
perSampleDominantTaxa,SummarizedExperiment-method (deprecate), 40
philr, *125*
philr::philr, *126*
phyloseq, *40*
phyloseq::rarefy_even_depth, *112*
picante::pd, *7*
plotColData, *11*, *19*
plotMDS, *79*
plotNMDS (deprecate), 40
plotReducedDim, *72*

quickPerCellQC, *117*

rarefyAssay, 111
rarefyAssay,SummarizedExperiment-method (rarefyAssay), 111
readQZA (deprecate), 40
reducedDims, *72*
relabundance (deprecate), 40
relabundance,SummarizedExperiment-method (deprecate), 40
relabundance<- (deprecate), 40
relabundance<-,SummarizedExperiment-method (deprecate), 40
relAbundanceCounts (deprecate), 40
relAbundanceCounts,SummarizedExperiment-method (deprecate), 40
resolveLoop, *73*, *121*
right_join, *108*
right_join (deprecate), 40
right_join,ANY-method (deprecate), 40
runCCA, *89*
runCCA (getCCA), 55
runDMN (calculateDMN), 33
runDPCoA (getDPCoA), 70
runJSD (deprecate), 40
runJSD,SummarizedExperiment-method (deprecate), 40
runNMDS (getNMDS), 76
runOverlap (deprecate), 40
runOverlap,SummarizedExperiment-method (deprecate), 40
runRDA (getCCA), 55
runUnifrac (deprecate), 40

scater::calculateMDS, *75*, *76*
scater::runMDS, *75*
scater::runPCA, *92*
scores, *57*
setTaxonomyRanks (taxonomyRanks), 118
splitAltExps, *30*, *114*
splitByRanks (agglomerateByRank), 26
splitOn, *30*, 113
splitOn,SingleCellExperiment-method (splitOn), 113
splitOn,SummarizedExperiment-method (splitOn), 113
splitOn,TreeSummarizedExperiment-method (splitOn), 113
stats::anova, *56*, *82*
stats::TukeyHSD, *56*, *82*
subsampleCounts (deprecate), 40
subsampleCounts,SummarizedExperiment-method (deprecate), 40
subsetByPrevalent (getPrevalence), 83
subsetByPrevalent,SummarizedExperiment-method (getPrevalence), 83
subsetByPrevalent,TreeSummarizedExperiment-method (getPrevalence), 83
subsetByPrevalentFeatures (deprecate), 40
subsetByPrevalentFeatures,ANY-method (deprecate), 40
subsetByPrevalentTaxa (deprecate), 40
subsetByPrevalentTaxa,ANY-method (deprecate), 40
subsetByRare (getPrevalence), 83
subsetByRare,SummarizedExperiment-method (getPrevalence), 83
subsetByRare,TreeSummarizedExperiment-method (getPrevalence), 83
subsetByRareFeatures (deprecate), 40
subsetByRareFeatures,ANY-method (deprecate), 40
subsetByRareTaxa (deprecate), 40
subsetByRareTaxa,ANY-method (deprecate), 40
subsetFeatures (deprecate), 40

subsetFeatures,SummarizedExperiment-method (deprecate), 40
subsetSamples (deprecate), 40
subsetSamples,SummarizedExperiment-method (deprecate), 40
subsetTaxa (deprecate), 40
subsetTaxa,SummarizedExperiment-method (deprecate), 40
SummarizedExperiment, *6*, *12*, *13*, *19–22*, *34*, *48*, *53*, *56*, *61*, *63*, *68*, *69*, *72*, *75*, *81–83*, *89*, *91*, *104*, *107*, *116*, *117*
summarizeDominance, 115
summarizeDominance,SummarizedExperiment-method (summarizeDominance), 115
summary (summarizeDominance), 115
summary,SummarizedExperiment-method (summarizeDominance), 115

taxonomy-methods (taxonomyRanks), 118
taxonomyRankEmpty (taxonomyRanks), 118
taxonomyRankEmpty,SummarizedExperiment-method (taxonomyRanks), 118
taxonomyRanks, *26*, 118
taxonomyRanks,SummarizedExperiment-method (taxonomyRanks), 118
taxonomyTree (deprecate), 40
taxonomyTree,SummarizedExperiment-method (deprecate), 40
Tengeler2020, *109*, 122
testExperimentCrossAssociation (deprecate), 40
testExperimentCrossAssociation,ANY-method (deprecate), 40
testExperimentCrossCorrelation (deprecate), 40
testExperimentCrossCorrelation,ANY-method (deprecate), 40
Tito2024QMP, *109*, 123
toTree, *73*, *121*
transformAssay, 124
transformAssay,SingleCellExperiment-method (transformAssay), 124
transformAssay,SummarizedExperiment-method (transformAssay), 124
transformCounts (deprecate), 40
transformFeatures (deprecate), 40
transformFeatures,SummarizedExperiment-method (deprecate), 40
transformSamples (deprecate), 40
transformSamples,SummarizedExperiment-method (deprecate), 40
TreeSummarizedExperiment, *5*, *14*, *15*, *25*, *28*, *36–40*, *56*, *69*, *71–74*, *77–80*, *86*, *96*, *98*, *99*, *102–104*, *106*, *111*, *114*, *116*, *119*, *124*
twins (dmn_se), 48

unsplitByRanks (agglomerateByRank), 26
unsplitByRanks,SingleCellExperiment-method (agglomerateByRank), 26
unsplitByRanks,TreeSummarizedExperiment-method (agglomerateByRank), 26
unsplitOn, *30*
unsplitOn (splitOn), 113
unsplitOn,list-method (splitOn), 113
unsplitOn,SimpleList-method (splitOn), 113
unsplitOn,SingleCellExperiment-method (splitOn), 113
utilization_functions (getReducedDimAttribute), 89

vegan::adonis2, *82*, *83*
vegan::betadisper, *56*, *57*, *82*, *83*
vegan::decostand, *92*, *126*
vegan::diversity, *7*
vegan::fisher.alpha, *7*
vegan::monoMDS, *78*, *79*
vegan::permutest, *56*, *82*, *83*
vegan::rrarefy, *112*
vegan:avgdist(), *16*
vegdist, *14–16*, *56*

ZTransform (deprecate), 40
ZTransform,SummarizedExperiment-method (deprecate), 40