Package 'AnalyzAIRR'

August 19, 2022

Version 1.0.0

Title MiAIRR compliant R package for the analysis of bulk Ig/TCR repertoire datasets

Description AnalyzeAIRR allows a general data exploration to evaluate the homogeneity within defined groups, identify outliers and filter them out. AnalyzeAIRR also proposes a set of diversity measures and statistical metrics applicable at any level of granularity. Thus, single-sample repertoire explorations or in-depth cross-comparisons of AIRR datasets can be conducted leading to ready-to-publish visualization graphics.
AnalyzeAIRR is complemented with a guided workflow to help users in their analytical strategy, and a Shiny web application making it user-friendly for biologists with little or no background in bioinformatics.

Author Vanessa Mhanna, Hang-Phuong Pham, Gabriel Pires, Nicolas Tchitchek, David Klatzmann, Adrien Six, Encarnita Mariotti-Ferrandiz

Maintainer Vanessa Mhanna <vanessajmhanna@gmail.com> and Encarnita Mariotti-Ferrandiz <encarnita.mariotti@sorbonne-universite.fr>

Depends R (>= 4.0.0), data.table

Imports pbapply, parallel, utils, stats, methods, graphics, ggplot2, lemon, DESeq2, ade4, limma, scales, rmdformats, kableExtra, vegan, naturalsort, RColorBrewer, dplyr, ggVennDiagram, gridExtra, plyr, grDevices,reshape2, ggprism, rstatix, eulerr, ggpubr, stringr, ggpmisc, tidyr, ggrepel, car, circlize, ComplexHeatmap, MESS

Suggests BiocStyle, knitr, rmarkdown

License GPL-3 + file LICENSE

URL https://github.com/tchitchek-lab/RepSeq

BugReports https://github.com/tchitchek-lab/RepSeq/issues

LazyData true

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.1

NeedsCompilation yes

biocViews SystemsBiology, Sequencing, DifferentialExpression, QualityControl

R topics documented:

AnalyzAIRR *X*

Description

AnalyzeAIRR is a MiAIRR compliant R package developed to analyze bulk Ig/TCR repertoire datasets.

countFeatures *Computing the occurrence of any repertoire level*

Description

This function calculates the occurrence of a selected repertoire level and returns the calculated values for all the samples within a RepSeqExperiment object. It takes into account the weight of the studied level, i.e. the number of sequences expressing a certain gene segment, or the count of a sequence in a sample.

Usage

```
countFeatures(
  x,
  level = c("clone", "V", "J", "VJ", "CDR3aa", "clonotype", "CDR3nt"),
  scale = c("count", "frequency"),
  group = NULL
)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the repertoire level to be analyzed. Should be one of "clone","clonotype","CDR3aa","CDR3nt","V", "J",or "VJ".
scale	a character specifying the type of occurrence to return: "count" or "frequency".
group	a vector of character indicating the group column name in mData and one experimental group within this column. Samples belonging to the chosen experimental group will be analyzed. The column must be of class factor. Default is NULL, values are calculated in all the samples within the dataset.

Value

a data.table summarizing the count or frequency of the analyzed level. In this table, rows correspond to the repertoire level, and columns correspond to the sample_ids.

Examples

```
data(RepSeqData)
level_statistics <- countFeatures(x = RepSeqData,
                                  level = "V",
                                  group=c("cell_subset", "amTreg"),
                                  scale="frequency")

level_statistics <- countFeatures(x = RepSeqData,
                                  level = "J",
                                  group=c("sex", "F"),
                                  scale="count")
```

`diffExpGroup` *Differential expression analysis*

Description

This function identifies differentially expressed repertoire levels between groups of samples.

Usage

```
diffExpGroup(
  x,
  colGrp,
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa"),
  group
)
```

Arguments

`x`	an object of class `RepSeqExperiment`
`colGrp`	a vector of character specifying the column names in the mData slot corresponding to the experimental condition to be analyzed.
`level`	a character specifying the level of the repertoire to be compared. Should be one of "clone","clonotype", "V", "J", "VJ", "CDR3nt" or "CDR3aa".
`group`	a vector of character indicating the column name in the mData slot, as well as the two groups to be compared.

Details

This function uses the DESeq2 package: https://www.rdocumentation.org/packages/DESeq2/versions/1.12.3/topics/DESeq Briefly, it estimates the size factors using the poscounts method which deals with zero counts. It then performs a default analysis by estimating the dispersion using a local regression of log dispersions over log base mean. Finally, a generalized linear model is fitted using a Negative Binomial distribution and Wald statistics.

Value

a data.frame with 6 columns: the repertoire level in rownames, and the baseMean, log2FoldChange, lfcSE, stat, pvalue and padj in columns. The table is ordered by adjusted p-values.

Examples

```
data(RepSeqData)
dds1 <- diffExpGroup(x = RepSeqData,
                     colGrp = "cell_subset" ,
                     level = "V",
                     group = c("cell_subset", "amTreg", "nTreg"))
```

`diffExpInd` *Differential expression analysis between two samples*

Description

This function compares the expression of a repertoire level between two samples. Log-ratios are calculated on the occurrence of a selected repertoire level between two compared samples, and differentially expressed genes/sequences are identified based on the user-defined log-ratio threshold.

Usage

```
diffExpInd(
  x,
  sampleNames = NULL,
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa"),
  scale = c("frequency", "count"),
  th = 1.5,
  remove.zeros = TRUE
)
```

Arguments

`x`	an object of class `RepSeqExperiment`
`sampleNames`	a vector of character with the sample_ids of the repertoires to drop from the RepSeqExperiment object.
`level`	a character specifying the repertoire level to be analyzed. Should be one of "clone","clonotype","CDR3aa","CDR3nt","V", "J",or "VJ".
`scale`	a character specifying the type of occurrence to take into account: "count" or "frequency".
`th`	the lof2FC threshold to be used
`remove.zeros`	a boolean indicating whether or not repertoire levels that are completely absent in one of the two compared samples should be to take into account in the calculation.

Value

a data frame containing the list of differentially expressed repertoire levels, their occurrence in both samples and their calculated log2FC.

Examples

```
data(RepSeqData)

diff_expression <- diffExpInd(RepSeqData,
                              level="V",
                              scale="frequency",
                              sampleNames = c("tripod-30-813","tripod-30-815"),
                              remove.zeros = FALSE)
```

`diversityIndices` *Calculation of diversity indices*

Description

This function computes a set of diversity indices of a repertoire level for each sample.

The calculated indices are the following:

- Shannon index: Calculates the proportional abundance of species in a repertoire (Shannon, 1948).

- Simpson index: Takes into account the number of species present as well as their abundance. It gives relatively little weight to the rare species and more weight to the frequent ones (Simpson, 1949).

- Inverse Simpson index: Is the effective number of species that is obtained when the weighted arithmetic mean is used to quantify average proportional abundance of species.

- Berger-Parker index: Expresses the proportional importance of the most abundant species. This metric is highly biased by sample size and richness (Berger and Parker 1970).

- Gini coefficient: Measures the degree of inequality in a distribution of abundances (Gini, 1921).

- Chao1: Estimates undetected species using the information on the rarest species (the numbers of singletons and doubletons) (Chao, 1984).

- Improved Chao1: an extension of Chao1 which uses additional information, namely, the numbers of tripletons and quadrupletons (Chiu et al., 2014).

Usage

```
diversityIndices(
  x,
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa")
)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the level of the repertoire on which the diversity should be estimated. Should be one of "clone","clonotype", "V", "J", "VJ", "CDR3nt" or "CDR3aa".

Value

a data table with the diversity indices calculated for each sample.

Examples

```
data(RepSeqData)
diversityIndices(RepSeqData, level="V")
```

---

dropSamples *Exclude repertoires from a RepSeqExperiment object.*

Description

This function allows the dropping of one or several samples from a RepSeqExperiment object by specifying their corresponding sample ids.

Usage

```
dropSamples(x, sampleNames)
```

Arguments

x	an object of class RepSeqExperiment
sampleNames	a vector of character with the sample_ids of the repertoires to drop from the RepSeqExperiment object.

Value

a RepSeqExperiment object.

Examples

```
data(RepSeqData)

dropRepSeqData<- dropSamples(x = RepSeqData,
            sampleNames=c("tripod-30-813", "tripod-30-815"))
```

`filterCount` *Filtering based on sequence count*

Description

This function filters out, in each sample, sequences having a count equal to or below a specified threshold. It can be applied on all the samples within a RepSeqExperiment object, or a group of samples belonging to a specific experimental group.

Usage

```
filterCount(
  x,
  level = c("clone", "clonotype", "CDR3aa", "CDR3nt"),
  n = 1,
  group = NULL
)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the type of sequences on which the filtering will be applied. Should be one of "clone","clonotype","CDR3aa" or "CDR3nt". For instance, for level="CDR3aa", counts will first be recalculated based on this column. Then, CDR3aa sequences with counts equal or below the n threshold will be excluded.
n	an integer specifying the count threshold below which sequences will be filtered out. For instance, for n=2, sequences with a count of 1 and 2 will be filtered out.
group	a vector of character indicating the group column name in mData and one experimental group within this column. Samples belonging to the chosen experimental group will be analyzed. The column must be of class factor. Default is NULL, values are calculated in all the samples within the dataset.

Value

an object of class RepSeqExperiment

Examples

```
data(RepSeqData)

filterdata <- filterCount(x=RepSeqData, n=1, level = "clone", group=c("cell_subset", "amTreg"))

filterdata <- filterCount(x=RepSeqData, n=1, level = "clonotype", group=NULL)
```

getPrivate *Extract private sequences*

Description

This function extract private sequences within the whole dataset, i.e. sequences found exclusively in one sample.

Usage

```
getPrivate(
  x,
  level = c("clone", "clonotype", "CDR3aa", "CDR3nt"),
  singletons = FALSE
)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the level of the repertoire to be taken into account. Should be one of clone","clonotype", "CDR3nt" or "CDR3aa".
singletons	a boolean indicating whether or not private sequences with a count of 1 should be extracted. Default is FALSE.

Value

an object of class RepSeqExperiment composed exclusively of private sequences.

Examples

```
data(RepSeqData)

privateclonotypes <- getPrivate(RepSeqData, level = "clonotype", singletons = FALSE)
```

getProductive *Extract productive sequences*

Description

Extract productive sequences from a RepSeqExperiment object by filtering out unproductive ones. Filtered sequences include:

- out-of-frame sequences: sequences with frame shifts based on the number of nucleotides in the CDR3nt column.

- sequences containing stop codons: CDR3aa sequences with a "*" or "~" symbols.

Usage

```
getProductive(x)
```

Arguments

x	an object of class RepSeqExperiment.

Value

a filtered RepSeqExperiment object exclusively containing productive sequences.

Examples

```
data(RepSeqData)
productiveData <- getProductive(x = RepSeqData)
```

---

getPublic *Extract public sequences*

Description

This function allows to subset a RepSeqExperiment object in order to keep sequences that are shared by at least two samples:

- belonging to a specified group

- within the whole dataset

Usage

```
getPublic(x, level = c("clone", "clonotype", "CDR3aa", "CDR3nt"), group = NULL)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the level of the repertoire to be taken into account. Should be one of "clone","clonotype", "CDR3nt" or "CDR3aa".
group	a vector of character indicating the group column name in the mData slot and one experimental group within this column. Samples belonging to the chosen experimental group will be analyzed. The column must be of class factor. Default is NULL, the analysis is performed on the whole dataset.

Value

an object of class RepSeqExperiment composed exclusively of shared sequences between the specified samples.

Examples

```
data(RepSeqData)

publicSeq <- getPublic(x = RepSeqData,
                              level = "clone",
                              group = c("cell_subset", "amTreg"))
```

`getTopSequences` *Extract the most frequent sequences*

Description

This function extracts the top n sequences within all samples or a group of samples.

Usage

```
getTopSequences(
  x,
  level = c("clone", "clonotype", "CDR3aa", "CDR3nt"),
  group = NULL,
  prop = 0
)
```

Arguments

`x`	an object of class `RepSeqExperiment`
`level`	a character specifying the level of the repertoire to be taken into account. Should be one of clone","clonotype", "CDR3nt" or "CDR3aa".
`group`	character, column name in sData. Must be of class factor. Default is NULL, the threshold is calculated across all the samples within the dataset.
`prop`	a numeric between 0 and 1 indicating the proportion of top sequences to extract.

Value

an object of class `RepSeqExperiment`

Examples

```
data(RepSeqData)
topClones <- getTopSequences(x = RepSeqData,
                          level = "clone",
                          group = c("cell_subset", "amTreg"), prop = 0.1)
```

getUnproductive *Extract unproductive sequences*

Description

Extract unproductive sequences from a RepSeqExperiment object, which include:

- out-of-frame sequences: sequences with frame shifts based on the number of nucleotides in the CDR3nt column

- sequences containing stop codons: CDR3aa sequences with a "*" or "~" symbols.

Usage

```
getUnproductive(x)
```

Arguments

x	an object of class RepSeqExperiment.

Value

a filtered RepSeqExperiment object exclusively containing unproductive sequences.

Examples

```
## Not run:
data(RepSeqData)

unproductiveData <- getUnproductive(x = RepSeqData)


## End(Not run)
```

mergeRepSeq *Merge RepSeqExperiment objects*

Description

This function allows the merging of two RepSeqExperiement objects.

The two objects must however contain completely different sample_ids, as duplicates are not supported.

This function can be useful in case users wish to analyze alignment files from different formats, or biological/technical replicates.

Usage

```
mergeRepSeq(a, b)
```

Arguments

a	the first RepSeqExperiment object.
b	the second RepSeqExperiment object.

Value

a RepSeqExperiment object containing all information from the two merged objects.

Examples

```
l <- list.files(system.file(file.path('extdata/mixcr'),
                    package = 'AnalyzAIRR'),
                    full.names = TRUE)

metaData <- read.table(system.file(file.path('extdata/sampledata.txt'),
                        package='AnalyzAIRR'),
                        sep = "\t",
                        row.names = 1, header = TRUE)

dataset1 <- readAIRRSet(fileList = l[c(1:3)],
                        cores=1L,
                        fileFormat = "MiXCR",
                        chain = "TRA",
                        sampleinfo = metaData[1:3,],
                        filter.singletons = FALSE,
                        outFiltered = FALSE,
                        raretab = FALSE)

dataset2 <- readAIRRSet(fileList = l[c(4:8)],
                        cores=1L,
                        fileFormat = "MiXCR",
                        chain = "TRA",
                        sampleinfo = metaData[4:8,],
                        filter.singletons = FALSE,
                        outFiltered = FALSE,
                        raretab = FALSE)

dataset <- mergeRepSeq(a = dataset1, b = dataset2)
```

perturbationScore *Calculation of the perturbation score*

Description

This function computes the perturbation scores of the CDR3aa length distribution within each V gene.

Scores are calculated as a distance between each repertoire and the mean of the control group using the ISEApeaks method (Colette and Six., 2002).

Usage

```
perturbationScore(
  x,
  ctrl.names,
  distance = c("manhattan", "euclidean", "canberra", "minkowski", "maximum"),
  p = 2
)
```

Arguments

x	an object of class RepSeqExperiment
ctrl.names	a vector of characters indicating the sample_ids to be used as controls.
distance	a character specifying the distance method to be used in the calculation of the perturbation scores. Should be one of the following: "manhattan", "euclidean", "canberra", "minkowski" or "maximum".
p	an integer indicating the power of the Minkowski distance. Default is 2.

Value

a data frame containing the perturbation scores for each V-gene.

Examples

```
data(RepSeqData)
pert <- perturbationScore(RepSeqData,
                          ctrl.names = c("tripod-30-813",
                                         "tripod-30-815",
                                         "tripod-31-846"),
                          distance = "manhattan")
```

plotColors *Color palette*

Description

This function allows an automatic color assigning, chosen from color-blinded friendly palettes, to every sample and experimental group in the metadata. As such, each sample/group will be assigned the same color in all the visualization functions.

Usage

```
plotColors(x)
```

Arguments

x	an object of class RepSeqExperiment

Value

a list of distinct colors assigned to each sample/group in the metadata.

Examples

```
data(RepSeqData)

colors <- plotColors(x = RepSeqData)
```

plotCountIntervals *Visualization of the clonal distribution per count interval*

Description

This function plots two histograms of the clonal distribution per a set of count intervals in all the samples within the dataset. Intervals are: 1,]1, 10],]10, 100],]100, 1000],]1000, 10000] and]10000, Inf].

The plot titled "Distribution" calculates the proportion of each interval in the whole repertoire, whereas the one titled "Cumulative frequency" shows the cumulative frequency of the sequences within each interval.

This could allow a global view of the repertoire fraction contributing the most to the repertoire. For instance, top sequences belonging to the highest interval often constitute a low fraction of the whole repertoire but contribute more significantly in terms of cumulative frequency in view of their high occurrence.

Samples can be statistically compared in each interval using the groupBy parameter.

Usage

```
plotCountIntervals(
  x,
  level = c("clone", "clonotype", "CDR3nt", "CDR3aa"),
  groupBy = NULL,
  label_colors = NULL
)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the level of the repertoire to be taken into account when calculating the clonal distribution. Should be one of clone","clonotype", "CDR3nt" or "CDR3aa".
groupBy	a character indicating a column name in mData. Colors are thus attributed to the different groups within this column, and statistical tests are performed between the chosen groups. The chosen column must be of class factor.
label_colors	a list of colors for each variable in groupBy See plotColors. If NULL, default colors are used.

Examples

```
data(RepSeqData)

plotCountIntervals(x = RepSeqData, level="CDR3aa")

plotCountIntervals(x = RepSeqData, level="CDR3nt",  groupBy="cell_subset")
```

---

plotDimReduction *Multidimensional Scaling analysis and Principal Component Analysis*

Description

This function can be used to visualize:

- repertoire dissimilarities by performing a multidimensional scaling (MDS) on a pairwise distance matrix calculated between samples on a selected repertoire level, using a specific dissimilarity method. The proposed methods are detailed in the plotDissimilarityMatrix function.

- differentially expressed repertoire levels calculated using the diffExpGroup function by performing a principal component analysis (PCA).

Usage

```
plotDimReduction(
  x,
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa"),
  method = c("manhattan", "euclidean", "canberra", "clark", "bray", "kulczynski",
   "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial",
    "chao", "cao", "mahalanobis"),
  colorBy = NULL,
  label_colors = NULL,
  dim_method = c("MDS", "PCA")
)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the level of the repertoire on which the diversity should be estimated. Should be one of "clone","clonotype", "V", "J", "VJ", "CDR3nt" or "CDR3aa".
method	a character specifying the distance method to be computed. Should be specified only if the dim_method parameter is set to "MDS", and should be one of the following: "manhattan", "euclidean", "canberra", "clark", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial", "chao", "cao", "mahalanobis."
colorBy	a character specifying the column name in mData to be used to attribute group colors. If not specified, a color is attributed per sample_id.
label_colors	a list of colors for each factor column in metaData. See plotColors. If NULL, default colors are used.
dim_method	a character indicating the dimensional reduction method to be performed. Should be one of "PCA" or "MDS".

Details

Details on the proposed dissimilarity indices can be found in the vegan package:

https://www.rdocumentation.org/packages/vegan/versions/2.4-2/topics/vegdist

Examples

```
data(RepSeqData)

plotDimReduction(x = RepSeqData,
                 level = "V",
                 method = "euclidean",
                 colorBy = "cell_subset",
                 dim_method="MDS")

plotDimReduction(x = RepSeqData,
                level = "J",
                colorBy = "cell_subset",
```

```
                dim_method="PCA")
```

`plotDissimilarityMatrix` *Visualization of repertoire dissimilarities*

Description

This function assesses the repertoire dissimilarities by plotting a heatmap of a squared distance matrix computed between samples using a specific dissimilarity method.

It calculates a list of dissimilarity indices, each taking into account different parameters. The proposed methods include:

The Jaccard similarity: a measure of similarity between sample sets defined as the size of the intersection divided by the size of the union of the sample sets.

The Morisita-Horn similarity: a measure of similarity that tends to be over-sensitive to abundant species.

The function also performs a hierarchical clustering on the calculated distance scores.

Usage

```
plotDissimilarityMatrix(
  x,
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa"),
  method = c("manhattan", "euclidean", "canberra", "clark", "bray", "kulczynski",
   "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial",
    "chao", "cao", "mahalanobis"),
 clustering = c("ward.D", "ward.D2", "single", "complete", "average", "mcquitty",
    "median", "centroid"),
  binary = FALSE,
  label_colors = NULL
)
```

Arguments

`x`	an object of class `RepSeqExperiment`
`level`	a character specifying the level of the repertoire on which the indices are computed. Should be one of "clone","clonotype", "V", "J", "VJ", "CDR3nt" or "CDR3aa".
`method`	a character specifying the distance method to be computed. Should be one of the following: "manhattan", "euclidean", "canberra", "clark", "bray", "kulczynski", "jaccard", "gower", "altGower", "morisita", "horn", "mountford", "raup", "binomial", "chao", "cao", "mahalanobis."
`clustering`	a character specifying the clustering method to be used.

binary	a boolean indicating whether or not to transform the data into a presence/absence data. Default is FALSE
label_colors	a list of colors for each factor column in metaData. See plotColors. If NULL, default colors are used.

Details

Details on the calculated indices as well as the clustering methods can be found in the vegan package: https://www.rdocumentation.org/packages/vegan/versions/2.4-2/topics/vegdist

Examples

```
data(RepSeqData)

plotDissimilarityMatrix(x = RepSeqData, level = "V", method = "euclidean")
```

---

plotDiversity *Visualization of the diversity indices*

Description

This function plots and compares a chosen diversity index calculated on a selected repertoire level between groups of samples.

The calculated indices can be one of the following:

- Shannon index: Calculates the proportional abundance of species in a repertoire.

- Simpson index: Takes into account the number of species present as well as their abundance. It gives relatively little weight to the rare species and more weight to the frequent ones

- Inverse Simpson index: Is the effective number of species that is obtained when the weighted arithmetic mean is used to quantify average proportional abundance of species.

- Berger-Parker index: Expresses the proportional importance of the most abundant species. This metric is highly biased by sample size and richness (Berger and Parker 1970).

- Gini coefficient: Measures the degree of inequality in a distribution of abundances.

- Chao1: Estimates undetected species using the information on the rarest species (the numbers of singletons and doubletons).

- Improved Chao1: an extension of Chao1 which uses additional information, namely, the numbers of tripletons and quadrupletons.

Usage

```
plotDiversity(
  x,
 index = c("chao1", "shannon", "simpson", "invsimpson", "bergerparker", "gini", "iChao"),
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa"),
  groupBy = NULL,
  label_colors = NULL
)
```

Arguments

x	an object of class RepSeqExperiment
index	a character specifying the diversity index to be estimated. Should be one of "chao1", shannon","invsimpson","simpson" or "gini".
level	a character specifying the level of the repertoire on which the diversity should be estimated. Should be one of "clone","clonotype", "V", "J", "VJ", "CDR3nt" or "CDR3aa".
groupBy	character indicating the groups to be compared. If not specified, no comparative statistical tests will be performed, and calculated values for each sample_id will be represented.
label_colors	a list of colors for each variable in ColorBy. See plotColors. If NULL, default colors are used.

Examples

```
data(RepSeqData)

plotDiversity(x = RepSeqData, level = "V", groupBy = "sex", index="shannon")

plotDiversity(x = RepSeqData, level = "clone", groupBy = "cell_subset", index="simpson")
```

---

plotEulerr *Visualization of the inter-repertoire sharing on an Euler diagram*

Description

The repertoire sharing at any level evaluates the degree of convergence between repertoires and experimental conditions.

This function plots the number of shared sequences, at any repertoire level, between samples belonging for instance, to a same experimental group. No limitations in the number of sample_ids is imposed, however for visualization purposes, a maximum of 5 sample_ids should be ideally specified.

Usage

```
plotEulerr(
  x,
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa"),
  sampleNames = NULL
)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the level of the repertoire on which the diversity should be estimated. Should be one of "clone","clonotype", "V", "J", "VJ", "CDR3nt" or "CDR3aa".
sampleNames	a vector of character indicating the sample_ids of the repertoires to be analyzed. If not specified, the first three samples in the dataset are analyzed.

Examples

```
data(RepSeqData)

snames <- rownames(mData(RepSeqData))[1:4]

plotEulerr(x = RepSeqData, level = "V", sampleNames = snames)

plotEulerr(x = RepSeqData, level = "J", sampleNames = NULL)
```

---

plotGeneUsage *Compare V or J gene distributions*

Description

This function compares the V or J gene usages between given groups.

Usage

```
plotGeneUsage(
  x,
  level = c("V", "J"),
  scale = c("count", "frequency"),
  groupBy = NULL,
  label_colors = NULL
)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the level of the repertoire to be taken into account when calculating the gene usages. Should be one of clone" or "clonotype".
scale	a character specifying whether to plot the gene usage in "count" or "frequency".
groupBy	character indicating the groups to be compared. If not specified, no comparative statistical tests will be performed, and calculated values for each sample_id will be represented.
label_colors	a list of colors for each variable in ColorBy. See plotColors. If NULL, default colors are used.

Examples

```
data(RepSeqData)

plotGeneUsage(x = RepSeqData, level = "J", scale = "count", groupBy="cell_subset")

plotGeneUsage(x = RepSeqData, level = "V",  scale = "frequency", groupBy="sex")
```

`plotIndCountIntervals` *Visualization of the clonal distribution per count interval in a single ample*

Description

This function plots two histograms of the clonal distribution per a set of fixed count intervals in a given sample. Intervals are: 1,]1, 10],]10, 100],]100, 1000],]1000, 10000] and]10000, Inf].

The upper plot calculates the proportion of each interval in the whole repertoire, whereas the lower plot shows the cumulative frequency of the sequences within each interval.

This allows a global view of the repertoire fraction contributing the most to the repertoire. For instance, top sequences belonging to the highest interval often constitute a low fraction of the whole repertoire but contribute more significantly in terms of cumulative frequency in view of their high occurrence.

Usage

```
plotIndCountIntervals(
  x,
  sampleName = NULL,
  level = c("clone", "clonotype", "CDR3nt", "CDR3aa")
)
```

Arguments

`x`	an object of class `RepSeqExperiment`
`sampleName`	a character specifying the sample_id to analyze. Default is NULL, which plots the first sample in the dataset.
`level`	a character specifying the level of the repertoire to be taken into account when calculating the clonal distribution. Should be one of clone","clonotype", "CDR3nt" or "CDR3aa".

Examples

```
data(RepSeqData)
snames <- rownames(mData(RepSeqData))

plotIndCountIntervals(x = RepSeqData, level="CDR3aa",sampleName = snames[1])

plotIndCountIntervals(x = RepSeqData, level="clone", sampleName = NULL)
```

`plotIndGeneUsage` *Visualization of the V or J gene usage*

Description

This function plots the V or J gene usages for a given sample.

Usage

```
plotIndGeneUsage(
  x,
  sampleName = NULL,
  level = c("V", "J"),
  scale = c("count", "frequency")
)
```

Arguments

`x`	an object of class `RepSeqExperiment`
`sampleName`	a character specifying the sample_id to analyze. Default is NULL, which plots the first sample in the dataset.
`level`	a character specifying the level of the repertoire to be taken into account when calculating the gene usages. Should be one of clone" or "clonotype".
`scale`	a character specifying whether to plot the gene usage in "count" or "frequency".

Examples

```
data(RepSeqData)
snames <- rownames(mData(RepSeqData))

plotIndGeneUsage(x = RepSeqData, level = "V", sampleName = snames[1], scale = "count")

plotIndGeneUsage(x = RepSeqData, level = "V", sampleName = snames[2], scale = "frequency")
```

plotPerturbationScore *Visualization of the perturbation scores*

Description

plots the calculated perturbation scores using the perturbationScore function onto a heatmap.

Usage

```
plotPerturbationScore(
  x,
  ctrl.names = NULL,
  distance = c("manhattan", "euclidean", "canberra", "minkowski", "maximum"),
  p = 2,
  order = "cell_subset",
  label_colors = NULL
)
```

Arguments

x	an object of class RepSeqExperiment
ctrl.names	a vector of characters indicating the sample_ids to be used as controls.
distance	a character specifying the distance method to be used in the calculation of the perturbation scores. Should be one of the following: "manhattan", "euclidean", "canberra", "minkowski" or "maximum".
p	an integer indicating the power of the Minkowski distance. Default is 2.
order	a character specifying the column name in mData to be used as reference to order the sample_ids in the heatmap, as no hierarchical clustering is performed in this analysis.
label_colors	a list of colors for each variable in ColorBy. See plotColors. If NULL, default colors are used.

Examples

```
data(RepSeqData)
plotPerturbationScore(x = RepSeqData,
                      ctrl.names = c("tripod-30-813" ,"tripod-31-846"  ,"tripod-35-970"),
                       distance = "manhattan",
                       order="cell_subset")
```

plotRankDistrib *Visualization of the clonal distribution as a function of the rank*

Description

This function plots the clonal distribution, at any repertoire level, as a function of the occurrence rank within each sample.

For instance, the most frequent sequence is attributed a rank of 1, and its relative abundance is plotted on the y-axis.

Usage

```
plotRankDistrib(
  x,
  level = c("clone", "clonotype", "CDR3nt", "CDR3aa"),
  scale = c("count", "frequency"),
  colorBy = NULL,
  grouped = FALSE,
  label_colors = NULL
)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the level of the repertoire to be taken into account to plot the clonal distribution. Should be one of "clone", "clonotype", "CDR3nt" or "CDR3aa".
scale	a character specifying whether to plot the clonal abundance in "count" or "frequency".
colorBy	a character indicating a column name in mData. Colors are thus attributed to the different groups within this column. The chosen column must be of class factor.
grouped	a boolean indicating whether or not the mean and se of samples belonging to the same experimental group specified in the ColorBy parameter should be computed. Default is FALSE. The area under the curve (AUC) is calculated for each sample and compared between chosen groups using a Wilcoxon test with Holm's correction.
label_colors	a list of colors for each variable in ColorBy. See plotColors. If NULL, default colors are used.

Examples

```
data("RepSeqData")

plotRankDistrib(x = RepSeqData, level="clone",colorBy = "sex", scale = "count")

plotRankDistrib(x = RepSeqData,level="clonotype", colorBy = "cell_subset", scale = "frequency")
```

plotRarefaction *Visualization of the rarefaction curves*

Description

This function plots the rarefaction curve for each sample within the dataset.

Rarefaction is a measure of species richness. The curves plot the number of clonotypes against the number of sequences in a sample, each being obtained by randomly re-sampling a number of sequences multiple times and representing the mean number of found clonotypes.

Calculation can be obtained using the rarefactionTab function which computes rarefaction values for each sample.

Usage

```
plotRarefaction(x, colorBy = NULL, label_colors = NULL)
```

Arguments

x	an object of class RepSeqExperiment
colorBy	a character indicating a column name in mData. Colors are thus attributed to the different groups within this column. The chosen column must be of class factor.
label_colors	a list of colors for each variable in ColorBy. See plotColors. If NULL, default colors are used.

Value

none

Examples

```
data(RepSeqData)

plotRarefaction(x = RepSeqData, colorBy = "sex")

plotRarefaction(x = RepSeqData, colorBy = "cell_subset")
```

plotRenyiIndex *Visualization of the Renyi index*

Description

This function plots the Renyi values at any repertoire level for all the samples in the dataset. The alpha values for which the Renyi is to be estimated can be personalized, thus allowing to focus on certain indices such as the Shannon index for alpha=1 or the Simpson index for alpha=2. For instance, the most frequent sequence is attributed a rank of 1, and its relative abundance is plotted on the y-axis.

Usage

```
plotRenyiIndex(
  x,
  alpha = c(0, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, Inf),
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa"),
  colorBy = NULL,
  grouped = FALSE,
  label_colors = NULL
)
```

Arguments

x	an object of class RepSeqExperiment
alpha	a numerical vector specifying the alpha values to compute. If not specified, the following values are estimated: c(0, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, Inf).
level	a character specifying the level of the repertoire to be taken into account when calculating VJ usages. Should be one of clone","clonotype", "V", "J", "VJ", "CDR3nt" or "CDR3aa".
colorBy	a character indicating a column name in mData. Colors are thus attributed to the different groups within this column. The chosen column must be of class factor.
grouped	a boolean indicating whether or not the mean and se of samples belonging to the same experimental group specified in the ColorBy parameter should be calculated. Default is FALSE. The area under the curve (AUC) is calculated for each sample and compared between chosen groups using a Wilcoxon test with Holm's correction.
label_colors	a list of colors for each variable in ColorBy. See plotColors. If NULL, default colors are used.

Details

The Renyi index is a generalization of the Shannon index. It represents the distribution of clonal expansions as a function of the parameter alpha. At alpha=0, it equally considers all species including the rare ones, whereas it up-weighs the abundant species with an increasing value of alpha. Alpha

=1 is an approximation of the Shannon index; alpha = 2 corresponds to the Simpson index and alpha=Inf corresponds to the Berger-Parker index. The latter highlights the highest clonal expansion in a repertoire.

Examples

```
data(RepSeqData)
plotRenyiIndex(x = RepSeqData, level = "V", colorBy = "sex")

plotRenyiIndex(x = RepSeqData, level = "J", colorBy = "sex", grouped=TRUE)
```

`plotScatter` *Visualization of the inter-repertoire sharing on a scatter plot*

Description

This function computes a correlation between a pair of repertoires at any repertoire level. It allows a comparative analysis of the occurrence of a given repertoire level between a pair of samples, which goes beyond the simple calculation of the sharing degree.

Usage

```
plotScatter(
  x,
  sampleNames = NULL,
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa"),
  scale = c("frequency", "log")
)
```

Arguments

`x`	an object of class `RepSeqExperiment`
`sampleNames`	a vector of character indicating the two sample_ids of the repertoires to be analyzed. If not specified, the first two samples in the dataset are analyzed.
`level`	a character specifying the level of the repertoire to be analyzes. Should be one of "clone", "clonotype", "V", "J", "VJ", "CDR3nt" or "CDR3aa".
`scale`	an integer indicating whether to plot a regular or a logarithmic scale.

Examples

```
data(RepSeqData)
plotScatter(x = RepSeqData,
            level = "V",
            sampleNames = c("tripod-30-813", "tripod-30-815"),
            scale = "log")
```

plotSpectratyping *Visualization of CDR3 spectratyping*

Description

CDR3 spectratyping allows large-scale analysis of the repertoire diversity based on the CDR3 amino acid length usage.

With this technique, a polyclonal repertoire is represented by an eight-peak bell-shaped profile, that is disturbed following an immune response.

This function plots a histogram of the CDR3 length distribution in a single sample. It is also possible to add the information on the V gene usage per CDR3 length. To do so, the proportion of the most used V genes within the repertoire can be specified with the prop parameter, thus showing their relative proportions within each CDR3 length.

Moreover, this function integrates a statistical method that can be applied on the spectratypes in order to compare peak distributions across different groups of samples called ISEApeaks (Colette and Six, 2002). The strategy can be applied using the `perturbationScore` function.

Usage

```
plotSpectratyping(
  x,
  sampleName = NULL,
  scale = c("count", "frequency"),
  prop = 0
)
```

Arguments

x	an object of class `RepSeqExperiment`
sampleName	a character specifying the sample_id to analyze. Default is NULL, which plots the first sample in the dataset.
scale	a character specifying whether to plot the length usage in "count" or "frequency".
prop	a numeric indicating the proportions of top V genes to be plotted. It ranges from 0 to 1.

See Also

`perturbationScore`

Examples

```
data(RepSeqData)
snames <- rownames(mData(RepSeqData))

plotSpectratyping(x = RepSeqData, sampleName = NULL, scale = "count", prop=1)
```

```
plotSpectratyping(x = RepSeqData, sampleName = NULL, scale = "frequency", prop=0)
```

---

plotSpectratypingV *Visualization of the CDR3 spectratyping per V gene*

Description

CDR3 spectratyping allows large-scale analysis of the repertoire diversity based on the CDR3 amino acid length usage. With this technique, a polyclonal repertoire is represented by an eight-peak bell-shaped profile that is disturbed following an immune response.

This function plots a histogram of the CDR3 length distribution per V gene in a single sample.

Usage

```
plotSpectratypingV(
  x,
  sampleName = NULL,
  scale = c("count", "frequency"),
  prop = 0
)
```

Arguments

x	an object of class RepSeqExperiment
sampleName	a character specifying the sample_id to analyze. Default is NULL, which plots the first sample in the dataset.
scale	a character specifying whether to plot the length usage in "count" or "frequency".
prop	a numeric indicating the proportions of top V genes to be plotted. It ranges from 0 to 1.

Examples

```
data(RepSeqData)

snames <- rownames(mData(RepSeqData))

plotSpectratypingV(x = RepSeqData, sampleName = snames[1], scale = "count",prop=0.05)

plotSpectratypingV(x = RepSeqData, sampleName = snames[1], scale = "frequency")
```

plotStatistics *Visualization of basic statistics*

Description

This function plots the statistics in the mData slot, calculated for each sample during the building of the RepSeqExperiment object.

These statistics include:

- nSequences: the total number of sequences in a sample
- CDR3nt: the number of unique nucleotide CDR3s
- CDR3aa: the number of unique amino acid CDR3s
- V: the total number of V genes expressed in each sample
- J: the total number of J genes
- VJ: the total number of V-J gene combinations
- clone: the number of unique clones
- clonotype: the number of unique clonotypes

They can be compared between groups of samples or simply plotted for each sample.

Usage

```
plotStatistics(
  x,
 stat = c("nSequences", "clone", "clonotype", "V", "J", "VJ", "CDR3aa", "CDR3nt"),
  groupBy = NULL,
  label_colors = NULL
)
```

Arguments

x	an object of class RepSeqExperiment
stat	a character specifying the statistic to plot. Should one of the statistics in the metaData slot.
groupBy	a character indicating the groups to be compared. A Wilcoxon test is thus performed and adjusted p-values using the Holm method are shown. If not specified, no statistical tests will be performed, and calculated values for each sample_id will be represented.
label_colors	a list of colors for each variable in groupBy See plotColors. If NULL, default colors are used.

Examples

```
data(RepSeqData)

plotStatistics(x = RepSeqData, groupBy = "sex", stat = "V")
```

plotVJusage *Visualization of the VJ combination usage*

Description

This function calculates the count or frequency of each combination by taking into account the weight of the chosen repertoire level: either "clone" or "clonotype".

It offers two types of visualization of the calculated VJ usage in a given sample.

- For prop=1, a heatmap with all the VJ combinations is plotted, with the V genes as columns and the J genes as rows.

- For prop!=1, a circos plot is drawn showing the top proportions of VJ combinations specified in the prop parameter.

Usage

```
plotVJusage(
  x,
  sampleName = NULL,
  scale = c("count", "frequency"),
  level = c("clone", "clonotype"),
  prop = 1
)
```

Arguments

x	an object of class RepSeqExperiment
sampleName	a character specifying the sample_id to analyze. Default is NULL, which plots the first sample in the dataset.
scale	a character specifying whether to plot the VJ usage in "count" or "frequency".
level	a character specifying the level of the repertoire to be taken into account when calculate VJ usages. Should be one of "clone" or "clonotype".
prop	a numeric indicating the proportions of top VJ combinations to be plotted. It ranges from 0 to 1.

Examples

```
data(RepSeqData)
snames <- rownames(mData(RepSeqData))

plotVJusage(x = RepSeqData, sampleName = NULL, scale = "count", level="clone", prop=1)
```

plotVolcano *Visualization of differential expression in a volcano plot*

Description

This function plots differentially expressed repertoire levels calculated using the diffExpGroup in a volcano plot.

Usage

```
plotVolcano(
  x,
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa"),
  group = c("cell_subset", "amTreg", "nTreg"),
  FC.TH = 2,
  PV.TH = 0.05,
  top = 10
)
```

Arguments

x	an object of class RepSeqExperiment
level	a character specifying the level of the repertoire on which the diversity should be estimated. Should be one of "clone","clonotype", "V", "J", "VJ", "CDR3nt" or "CDR3aa".
group	a vector of character indicating the column name in the mData slot, as well as the two groups to be compared.
FC.TH	an integer indicating the log2FoldChange threshold. Default is 2.
PV.TH	an integer indicating the adjusted pvalue threshold. Default is 0.05.
top	an integer indicating the top n significant labels to be shown on the plot. Default is 10.

Examples

```
data(RepSeqData)
plotVolcano(x = RepSeqData,
            level = "V",
            group = c("cell_subset", "amTreg", "nTreg"),
            top = 10,
            FC.TH = 1,
            PV.TH = 0.05)
```

rarefactionTab *Calculation of rarefaction values*

Description

This function computes rarefaction values for each sample.

Rarefaction is a measure of species richness. It assesses the number of observed clonotypes for sets of N of sequences in a sample by randomly re-sampling a number of sequences multiple times and calculating the mean number of observed clonotypes.

Usage

```
rarefactionTab(x)
```

Arguments

x	an object of class RepSeqExperiment

Value

a data.table of values that can be represented graphically as rarefaction curves. The function plotRarefaction can be used for that purpose.

Examples

```
data(RepSeqData)

rarefactionTab(x = RepSeqData)
```

readAIRRSet *Building of a RepSeqExperiment object*

Description

This function builds a RepSeqExperiment object from alignment files generated by MiXCR and immunoseq, as well as files formatted following the AIRR standards guidelines.

A set of filters can be applied on the loaded files during the building process including:

- filtering out ambiguous sequences

- filtering out unproductive sequences including the ones with stop codons, namely out-of-frame sequences

- filtering out singletons, i.e clonotypes with an occurrence of 1

- filtering out short or extensively long amino acid CDR3 sequences

Usage

```
readAIRRSet(
  fileList,
  fileFormat = c("MiXCR", "immunoseq", "MiAIRR"),
  chain = c("TRA", "TRB", "TRG", "TRD", "IGH", "IGK", "IGL"),
  sampleinfo = NULL,
  keep.ambiguous = FALSE,
  keep.unproductive = FALSE,
  filter.singletons = FALSE,
  aa.th = NULL,
  outFiltered = FALSE,
  raretab = FALSE,
  cores = 1L
)
```

Arguments

`fileList`	a list of paths to the alignment files. File format can be one of the following: .tsv, txt.gz, .zip, or .tar
`fileFormat`	a character vector specifying the format of the input files, i.e. the tool that was used to generate/align the files. Should be one of "MiXCR", "immunoseq", or "MiAIRR".
`chain`	a character vector indicating a single TCR or Ig chain to analyze. The vector can be one of the following: - "TRA", "TRB","TRG" or "TRD" for the TCR repertoires - "IGH","IGK" or "IGL" for the BCR repertoires
`sampleinfo`	a data frame containing: - a column with the sample_ids. Ids should match the base names of the corresponding files as well as their order in the list. This column should be assigned as row.names when the metadata file is loaded (see the example below). - any additional columns with relevant information for the analyses. Group columns must be transformed into factors after loading (see the example below). No specific column names nor order are required
`keep.ambiguous`	a boolean indicating whether or not ambiguous sequences containing an "N" in their nucleotide CDR3 should be left in. Default is `FALSE`.
`keep.unproductive`	a boolean indicating whether or not unproductive sequences should be left in. Default is `FALSE`. Unproductive sequences include: - out-of-frame sequences: sequences with frame shifts based on the number of nucleotides in their CDR3s - sequences containing stop codons: CDR3aa sequences with a "*" or "~" should be kept. Default is `FALSE`
`filter.singletons`	a boolean indicating whether or not clonotypes (CDR3nt+V+CDR3aa+J) with an occurrence of 1 should be filtered out. Default is `FALSE`

aa.th	an integer determining the CDR3 amino acid sequence length limits, i.e. the maximum number of amino acids deviating from the mean length that is accepted. The default value is 8, which keeps amino acid CDR3s with a length that falls inside the following range: mean length-8 =< aa.th <= mean length+ 8.
outFiltered	a boolean indicating whether or not to write in the oData slot of the RepSeqexperiment object a data frame containing the filtered out reads.
raretab	a boolean indicating whether or not a rarefaction table should be generated. It uses the rarefactionTab function, which computes a rarefaction table for each sample. Default is TRUE.
cores	an integer indicating the number of CPU cores to be used by the function. The process is paralleled and can thus be used on multiple cores. Default is 1.

Value

an object of class RepSeqExperiment that is used in all the analytical metrics proposed by the AnalyzAIRR package. See RepSeqExperiment-class for more details.

Examples

```
l <- list.files(system.file(file.path('extdata/mixcr'),
                    package = 'AnalyzAIRR'),
                    full.names = TRUE)

metaData <- read.table(system.file(file.path('extdata/sampledata.txt'),
                        package='AnalyzAIRR'),
                        sep = "\t",
                        row.names = 1, header = TRUE)

metaData$cell_subset <- factor(metaData$cell_subset)
metaData$sex <- factor(metaData$sex)

dataset <- readAIRRSet(fileList = l,
                       fileFormat = "MiXCR",
                       chain = "TRA",
                       sampleinfo = metaData,
                       filter.singletons = FALSE,
                       aa.th=9,
                       outFiltered = FALSE,
                       raretab = FALSE,
                       cores=1L)
```

---

readFormatSet *An alternative method for the building of a RepSeqExperiment object*

Description

This function allows the loading of alignment files in other formats than the ones supported by the readAIRRSet function. These files should contain the following column names: - sample_id: Sample names

- CDR3nt: CDR3 nucleotide sequence

- CDR3aa: CDR3 amino acid sequence

- V: Variable gene name (IMGT nomenclature)

- J: Joining gene name (IMGT nomenclature)

- count: the occurrence of each sequence

The order of the columns must be respected. Input files can contain additional columns that won't however be taken into account in the generated RepSeqExperiment object.

Similarly to the readAIRRSet function, readFormatSet applies a series of filters during the building process. The possible filters that can be applied include:

- filtering out ambiguous sequences

- filtering out unproductive sequences including the ones with stop codons, namely out-of-frame sequences

- filtering out singletons, i.e clonotypes with an occurrence of 1

- filtering out short or extensively long amino acid CDR3 sequences

Usage

```
readFormatSet(
  fileList,
  chain = c("TRA", "TRB", "TRG", "TRD", "IGH", "IGK", "IGL"),
  sampleinfo = NULL,
  keep.ambiguous = FALSE,
  keep.unproductive = FALSE,
  filter.singletons = FALSE,
  aa.th = 8,
  outFiltered = TRUE,
  raretab = TRUE,
  cores = 1L
)
```

Arguments

fileList	a list of paths to the alignment files. File format can be one of the following: .tsv, txt.gz, .zip, or .tar
chain	a character vector indicating a single TCR or Ig chain to analyze. The vector can be one of the following: - "TRA", "TRB","TRG" or "TRD" for the TCR repertoires - "IGH","IGK" or "IGL" for the BCR repertoires

`sampleinfo` a data frame containing:

- a column with the sample_ids. Ids should match the base names of the corresponding files as well as their order in the list. This column should be assigned as row.names when the metadata file is loaded (see the example below).

- any additional columns with relevant information for the analyses. Group columns must be transformed into factors after loading (see the example below).

No specific column names nor order are required

`keep.ambiguous` a boolean indicating whether or not ambiguous sequences containing an "N" in their nucleotide CDR3 should be left in. Default is `FALSE`.

`keep.unproductive`

a boolean indicating whether or not unproductive sequences should be left in. Default is `FALSE`. Unproductive sequences include:

- out-of-frame sequences: sequences with frame shifts based on the number of nucleotides in their CDR3s

- sequences containing stop codons: CDR3aa sequences with a "*" or "~" should be kept. Default is `FALSE`

`filter.singletons`

a boolean indicating whether or not clonotypes (CDR3nt+V+CDR3aa+J) with an occurrence of 1 should be filtered out. Default is `FALSE`

`aa.th` an integer determining the CDR3 amino acid sequence length limits, i.e. the maximum number of amino acids deviating from the mean length that is accepted. The default value is 8, which keeps amino acid CDR3s with a length that falls inside the following range: mean length-8 =< aa.th <= mean length+8.

`outFiltered` a boolean indicating whether or not to write in the oData slot of the RepSeqexperiment object a data frame containing the filtered out reads.

`raretab` a boolean indicating whether or not a rarefaction table should be generated. It uses the `rarefactionTab` function which computes a rarefaction table for each sample. Default is `TRUE`.

`cores` an integer indicating the number of cores to be uses by the function. The process is paralleled and can thus be used on multiple cores. Default is 1.

Value

an object of class `RepSeqExperiment` that is used in all the analytical metrics proposed by the AnalyzAIRR package. See `RepSeqExperiment-class` for more details.

Examples

```
l <- list.files(system.file(file.path('extdata/informat'),
                    package = 'AnalyzAIRR'),
                    full.names = TRUE)
l # list of gz-compressed files

dataset <- readFormatSet(fileList = l,
                         sampleinfo = NULL,
                         cores = 10,
```

```
                    chain = "TRA",
                    keep.ambiguous = FALSE,
                    keep.unproductive = FALSE,
                    filter.singletons = FALSE,
                    aa.th = 8,
                    raretab = TRUE,
                    outFiltered = TRUE)
```

`renyiIndex` *Computation of the Renyi index*

Description

This function computes the Renyi values at any repertoire level for all the samples in the RepSeqExperiment object.

The alpha values for which the Renyi is to be estimated can be personalized, thus allowing to focus on certain indices such as the Shannon index for alpha=1 or the Simpson index for alpha=2.

Usage

```
renyiIndex(
  x,
  alpha = c(0, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, Inf),
  level = c("clone", "clonotype", "V", "J", "VJ", "CDR3nt", "CDR3aa")
)
```

Arguments

`x`	an object of class RepSeqExperiment
`alpha`	a numerical vector specifying the alpha values to compute. If not specified, the following values are estimated: c(0, 0.25, 0.5, 1, 2, 4, 8, 16, 32, 64, Inf).
`level`	a character specifying the level of the repertoire to be taken into account when calculate VJ usages. Should be one of clone","clonotype", "V", "J", "VJ", "CDR3nt" or "CDR3aa".

Details

The Renyi index is a generalization of the Shannon index. It represents the distribution of clonal expansions as a function of the parameter alpha. At alpha=0, it equally considers all species including the rare ones, whereas it up-weighs the abundant species with an increasing value of alpha. Alpha =1 is an approximation of the Shannon index; alpha = 2 corresponds to the Simpson index and alpha=Inf corresponds to the Berger-Parker index. The latter expresses the proportional importance of the most abundant species.

Value

a data.table with the Renyi values calculated for all alpha in each sample.

Examples

```
data(RepSeqData)

renyiIndex(RepSeqData, level = "V", alpha = 1)
```

`RepSeqData` *RepSeqData*

Description

A RepSeqExperiment object built using a published TCR repertoire dataset of healthy murine cell populations (Mhanna et al., 2021). Files aligned using MiXCR are available in extdata/mixrc.

Usage

```
data(RepSeqData)
```

Format

RepSeqExperiment object and gz-compressed txt/csv tab-delimited:

`system.file(file.path('extdata/mixcr'), package='AnalyzAIRR')` Aligned data sets using MiXCR

`system.file(file.path('extdata/sampledata.txt'), package='AnalyzAIRR')` Metadata

`RepSeqExperiment-class`
The class RepSeqExperiment

Description

An S4 class object enclosing the adaptive immune repertoire data, and which is used in all the analytical metrics proposed by the AnalyzAIRR package. The RepSeqExperiment object is composed of 4 slots, each containing different information.

`[]` Extract parts of the `RepSeqExperiment` object

Usage

```
assay(object)

assay(object, i, j) <- value

oData(object)

oData(object) <- value

mData(object)

mData(object) <- value

History(object)

History(object) <- value

## S4 method for signature 'RepSeqExperiment'
assay(object)

## S4 replacement method for signature 'RepSeqExperiment'
assay(object, i, j) <- value

## S4 method for signature 'RepSeqExperiment'
oData(object)

## S4 replacement method for signature 'RepSeqExperiment'
oData(object) <- value

## S4 method for signature 'RepSeqExperiment'
mData(object)

## S4 replacement method for signature 'RepSeqExperiment'
mData(object) <- value

## S4 method for signature 'RepSeqExperiment'
History(object)

## S4 replacement method for signature 'RepSeqExperiment'
History(object) <- value

## S4 method for signature 'RepSeqExperiment'
show(object)

## S4 method for signature 'RepSeqExperiment'
names(x)

## S4 replacement method for signature 'RepSeqExperiment,ANY'
```

```
names(x) <- value

## S4 method for signature 'RepSeqExperiment,ANY,ANY,ANY'
x[i, j, drop]

is.RepSeqExperiment(x)
```

Arguments

object	a RepSeqExperiment object
i	indice(s) of clonotype(s) to extract
j	indice(s) of sample(s) to extract
value	either numeric, character or data frame
x	an object
drop	If TRUE the result is coerced to the lowest possible dimension

Value

an object of class [RepSeqExperiment]

Slots

assayData a data.table binding all clonotype tables and containing the following columns:

- sample_id: sample names
- V: Variable gene name
- J: Joining gene name
- CDR3aa: CDR3 amino acid sequence
- CDR3nt: CDR3 nucleotide sequence
- clone: Full clonotype sequence including the V gene, the amino acid CDR3 sequence and the J gene
- VJ: V-J gene combination
- count: the occurrence of the clonotype, i.e the clone sequence

metaData a data frame containing sample information specified during the building of the RepSeqExperiment object. Each row represents a sample, and columns the possible information or group that can be attributed to the samples such as the cell population, the donor's age, sex, etc...

otherData a list of meta data containing any other information than the user does not wish to include in the analysis.

History a data frame registering all operations performed on RepSeqExperiment object, such as the filtering functions and the normalization.

`sampleRepSeqExp` *Down-sampling of repertoires*

Description

This function down-samples all repertoires within the dataset to the same size.

Users can choose the value to which all the samples are down-sampled. If not specified, the lowest number of sequences across all samples within the dataset will be used.

This strategy can be applied when studying different cell subsets with significant differences in their repertoire sizes.

Usage

```
sampleRepSeqExp(
  x,
  sample.size = min(mData(x)$nSequences),
  rngseed = FALSE,
  replace = TRUE,
  verbose = TRUE
)
```

Arguments

`x`	an object of class `RepSeqExperiment`
`sample.size`	an integer indicating the desired down-sampled size. The default is the smallest repertoire size among all samples of the dataset.
`rngseed`	a integer used as seed for a reproducible result.
`replace`	a boolean indicating if the resampling should be performed with or without replacement. Default is TRUE.
`verbose`	a boolean indicating whether or not to show the details of every computation step within the function. Default is TRUE.

Value

a new `RepSeqExperiment` object with the downsized data.

Examples

```
data(RepSeqData)

RepSeqData_ds<- sampleRepSeqExp(x = RepSeqData, rngseed = 1234, replace = TRUE)

RepSeqData_ds<- sampleRepSeqExp(x = RepSeqData, rngseed = FALSE, replace = FALSE)
```

ShannonNorm *Shannon normalization of repertoires*

Description

This function allows to normalize repertoires using the Shannon entropy as a threshold, thus eliminating rare sequences.

It is described in (Chaara et al., 2018) and is used to eliminate "uninformative" sequences resulting from experimental noise.

It is particularly efficient when applied on small samples as it corrects altered count distributions caused by a high-sequencing depth.

Sequences that are eliminated with this method are stored in the `oData` slot.

Usage

```
ShannonNorm(x)
```

Arguments

x	an object of class `RepSeqExperiment`

Value

a new `RepSeqExperiment` object with the normalized data.

Examples

```
data(RepSeqData)

RepSeqData_sh<- ShannonNorm(x = RepSeqData)
```

theme_RepSeq *A theme for the visualization plots*

Description

A unique plot theme used in all the visualization plots. As this function is generated using the ggplot2 package, users can easily customize all of the generated plots using ggplot2 arguments.

Usage

```
theme_RepSeq()
```

Index