Package 'VDJdive'

November 12, 2026

Title Analysis Tools for 10X V(D)J Data

Version 1.14.0

Description This package provides functions for handling and analyzing immune receptor repertoire data, such as produced by the CellRanger V(D)J pipeline. This includes reading the data into R, merging it with paired single-cell data, quantifying clonotype abundances, calculating diversity metrics, and producing common plots. It implements the E-M Algorithm for clonotype assignment, along with other methods, which makes use of ambiguous cells for improved quantification.

Depends R (>= 4.2)

Imports BiocParallel, cowplot, ggplot2, gridExtra, IRanges, Matrix, methods, RColorBrewer, Rcpp, S4Vectors, SingleCellExperiment, stats, SummarizedExperiment, utils

Suggests breakaway, covr, knitr, rmarkdown, testthat, BiocStyle

LinkingTo Rcpp

License Artistic-2.0

URL https://github.com/kstreet13/VDJdive

BugReports https://github.com/kstreet13/VDJdive/issues

biocViews Software, ImmunoOncology, SingleCell, Annotation, RNASeq, TargetedResequencing

Encoding UTF-8

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

StagedInstall no

VignetteBuilder knitr

Collate 'RcppExports.R' 'clonoStats_class.R' 'abundanceVDJ.R' 'barVDJ.R' 'boxVDJ.R' 'calculateDiversity.R' 'clonoStats_helpers.R' 'clonoStats.R' 'contigs.R' 'pieVDJ.R' 'runBreakaway.R' 'runVDJPCA.R' 'scatterVDJ.R' 'setup.R' 'utils.R'

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

git_branch RELEASE_3_23

git_last_commit 521b34b

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Kelly Street [aut, cre] (ORCID:
<https://orcid.org/0000-0001-6379-5013>),
Mercedeh Movassagh [aut] (ORCID:
<https://orcid.org/0000-0001-7690-0230>),
Jill Lundell [aut] (ORCID: <https://orcid.org/0000-0002-6048-4700>),
Jared Brown [ctb],
Linglin Huang [ctb],
Mingzhi Ye [ctb]

Maintainer Kelly Street <street.kelly@gmail.com>

Contents

abundanceVDJ 2
addVDJtoSCE 3
barVDJ 5
boxVDJ 6
calculateDiversity 7
clonoStats 8
clonoStats-class 10
contigs 12
pieVDJ 14
readVDJcontigs 15
runBreakaway 16
runVDJPCA 17
scatterVDJ 17
splitClonotypes 19
summarizeClonotypes 20
writeVDJcontigs 21

Index **23**

---

abundanceVDJ *Create an abundance graph for clonotype expansion*

Description

abundanceVDJ creates a dot plot using ggplot that shows the number of reads for each clonotype in each sample and labels the most abundant clonotypes.

Usage

```
abundanceVDJ(x, ...)

## S4 method for signature 'clonoStats'
abundanceVDJ(x, annotate = 5, title = NULL)
```

Arguments

x	A matrix created with clonoStats.
...	additional arguments.
annotate	An integer that specifies how many of the most abundant clonotypes should be annotated on the plot.
title	Character vector with an optional title. If FALSE, no title is generated.

Value

Returns a ggplot plot with a dot plot that shows the abundance of the clonotypes in each sample. The most abundant clonotypes are annotated on the plot and ordered from most abundant to least abundant.

Examples

```
data('contigs')
x <- clonoStats(contigs)
abundanceVDJ(x)
```

---

addVDJtoSCE *Add 10X CellRanger V(D)J data to SingleCellExperiment*

Description

Matches CellRanger V(D)J data to paired data in an existing SingleCellExperiment object.

Usage

```
addVDJtoSCE(samples, sce, ...)

## S4 method for signature 'SplitDataFrameList,SingleCellExperiment'
addVDJtoSCE(samples, sce, sample.names = NULL, barcode = "Barcode")

## S4 method for signature 'character,SingleCellExperiment'
addVDJtoSCE(samples, sce, sample.names = names(samples), barcode = "Barcode")
```

Arguments

samples	A character vector containing one or more directory names, each corresponding to a 10X sample. Each directory should contain a file named `filtered_contig_annotations.csv`. Alternatively, a `SplitDataFrameList`, the output of `readVDJcontigs`.
sce	A `SingleCellExperiment` object.
...	additional arguments.
sample.names	A character vector of length equal to `samples` containing the sample names to store in the output object. If `NULL` and `samples` is a character vector, the `basenames` of each directory will be used.
barcode	The column name from the `colData` of `sce` containing cell barcodes. These should match the barcodes in the V(D)J data (see Details). Alternatively, a vector of cell barcodes of length equal to `ncol(sce)`.

Details

Matching cell barcodes between data objects can cause problems, because different methods have different ways of ensuring barcodes are unique across all samples. This function and `readVDJcontigs` follow the naming conventions of `read10xCounts`, where the sample index (in the `samples` vector) is appended to each cell barcode, to ensure that each barcode is unique, across all samples. If `sce` was created by a different method, such as conversion from a `Seurat` object, you may need to check the barcode naming convention.

Value

A `SingleCellExperiment` object with an element named `"contigs"` added to the `colData`, representing the V(D)J data.

Examples

```
# load example V(D)J data
data('contigs')
# make SCE object with matching barcodes and sample IDs
ncells <- 24
u <- matrix(rpois(1000 * ncells, 5), ncol = ncells)
barcodes <- vapply(contigs[,'barcode'], function(x){ x[1] }, 'A')
samples <- vapply(contigs[,'sample'], function(x){ x[1] }, 'A')
sce <- SingleCellExperiment::SingleCellExperiment(assays = list(counts = u),
                            colData = data.frame(Barcode = barcodes,
                                                 sample = samples))
sce <- addVDJtoSCE(contigs, sce)
sce$contigs
```

barVDJ *Create a bar graph for clonotype expansion*

Description

barVDJ creates a barplot using ggplot that shows the number of reads in the sample and colors the sample in accordance to the amount of diversity.

Usage

```
barVDJ(x, ...)

## S4 method for signature 'Matrix'
barVDJ(x, title = NULL, legend = FALSE)

## S4 method for signature 'matrix'
barVDJ(x, ...)

## S4 method for signature 'clonoStats'
barVDJ(x, ...)
```

Arguments

x	A matrix created with clonoStats.
...	additional arguments.
title	Character vector with an optional title. If FALSE, no title is generated.
legend	If TRUE, a legend will be included with the plot. If FALSE, no legend is included in the plot.

Value

Returns a ggplot plot with a barplot that shows the abundance of the clonotypes. The coloring indicates the number of cells for each clonotype with darker colors being clonotypes with a single cell (singletons) and lighter colors having more cells with that clonotype (expanded clonotype).

Examples

```
data('contigs')
x <- clonoStats(contigs)
barVDJ(x)
```

boxVDJ *Create a box plot for diversity measures*

Description

boxVDJ creates a box plot of the specified diversity.

Usage

```
boxVDJ(d, ...)

## S4 method for signature 'matrix'
boxVDJ(
  d,
  sampleGroups = NULL,
  method = c("shannon", "simpson", "invsimpson", "chao1", "chaobunge"),
  title = NULL,
  legend = FALSE
)
```

Arguments

d	A matrix created with calculateDiversity.
...	additional arguments.
sampleGroups	A matrix or data.frame that identifies the groups that each sample belongs to. The matrix must contain two columns. The first column lists the individual samples and should be called "Sample". The second column should list the group that each sample belongs to (e.g. Normal and Tumor) and be called "Group". If no sampleGroups dataset is provided, all of the samples will be plotted in one group.
method	Identifies the type of diversity that is to be plotted.
title	Character vector with an optional title.
legend	If TRUE, a legend will be included with the plot. If FALSE, no legend is included in the plot.

Value

Returns a ggplot plot with a box plot that shows the diversity for each sample. A box plot is created for each of the grouping variables. The individual diversity measures are plotted on the box plots.

Examples

```
data('contigs')
x <- clonoStats(contigs)
d <- calculateDiversity(x)
sampleGroups <- data.frame(Sample = c("sample1", "sample2"),
```

```
                            Group = c("Cancer", "Normal"))
boxVDJ(d, sampleGroups = sampleGroups, method = "shannon",
       title = "Shannon diversity", legend = FALSE)
```

`calculateDiversity` *Sample diversity estimation*

Description

This function uses various methods to estimate the clonotypic diversity of samples based on a matrix of clonotype abundances (samples are columns).

Usage

```
calculateDiversity(x, ...)

## S4 method for signature 'clonoStats'
calculateDiversity(
  x,
 methods = c("all", "nCells", "nClonotypes", "shannon", "normentropy", "invsimpson",
    "ginisimpson", "chao1", "chaobunge"),
  ...
)

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

Arguments

`x`	A matrix of abundance values where rows are features (clonotypes) and columns are samples. This is created with `summarizeClonotypes` using a sparse matrix computed with either `EMquant` or `CRquant`.
`...`	Additional arguments passed to external calculation methods.
`methods`	A character vector specifying which diversity measures to use (default = `'all'`, see Details).

Details

Available methods are total cells with appropriate TCR data (`'nCells'`, not a diversity measure, but a useful point of comparison), total clonotypes (`'nClonotypes'`), Shannon entropy (`'shannon'`), Simpson index (`'simpson'`), inverse Simpson index (`'invsimpson'`), Chao1 richness (`'chao1'`), and Chao-Bunge richness (`'chaobunge'`). A special value of `'all'` is also allowed, which will run all methods listed above.

The `'chao1'` and `'chaobunge'` estimates assume all abundances are integers. When this is not the case for the input matrix, `k`, all values are multiplied by the `scaling_factor` and rounded to the nearest integer. The resulting estimate is then divided by `scaling_factor` to return to the original scale. The `'shannon'`, `'simpson'`, and `'invsimpson'` methods work with any input type.

Value

A matrix of diversity estimates for each sample. Note that the 'chaobunge' method also includes an estimate of the standard error.

Examples

```
data('contigs')
x <- clonoStats(contigs)
calculateDiversity(x)
```

---

clonoStats *Assign cell-level clonotypes and calculate abundances*

Description

Assign clonotype labels to cells and produce two summary tables: the clonotypes x samples table of abundances and the counts x samples table of clonotype frequencies.

Usage

```
clonoStats(x, ...)

## S4 method for signature 'SplitDataFrameList'
clonoStats(
  x,
  group = "sample",
  type = NULL,
  assignment = FALSE,
  method = "EM",
  lang = c("cpp", "r"),
  thresh = 0.01,
  iter.max = 1000,
  BPPARAM = SerialParam()
)

## S4 method for signature 'SingleCellExperiment'
clonoStats(x, contigs = "contigs", group = "sample", ...)

## S4 method for signature 'clonoStats'
clonoStats(x, group = NULL, lang = c("cpp", "r"))
```

Arguments

x	A SplitDataFrameList object containing V(D)J contig information, split by cell barcodes, as created by readVDJcontigs. Alternatively, a SingleCellExperiment object with such a SplitDataFrameList in the colData, as created by addVDJtoSCE.

...	additional arguments.
group	character. The name of the column in x (or the colData of x, for SingleCellExperiment objects) that stores each cell's group identity, typically either its sample of origin or cluster label. Alternatively, a vector of length equal to x (or ncol(x)) indicating the group identity. Providing this information can dramatically speed up computation. When running clonoStats for the first time on a dataset, we highly recommend setting the group identity to sample of origin to avoid unwanted cross-talk between samples.
type	character. The type of VDJ data (one of "TCR" or "BCR"). If NULL, this is determined by the most prevalent chain types in x.
assignment	logical. Whether or not to return the full nCells x nClonotypes sparse matrix of clonotype assignments (default = FALSE)
method	character. Which method to use for assigning cell-level clonotypes. Options are "EM" (default), "unique", or "CellRanger". Alternatively, this may be the name of a numeric column of the contig data or any chain type contained therein. See Details.
lang	character. Indicates which implementation of certain methods to use. The EM algorithm is implemented in both pure R ('r') and mixed R and C++ ('cpp', default) versions. Similarly, clonotype summarization is implemented in two ways, which can impact speed, regardless of choice of method.
thresh	Numeric threshold for convergence of the EM algorithm. Indicates the maximum allowable deviation in a count between updates. Only used if method = "EM".
iter.max	Maximum number of iterations for the EM algorithm. Only used if method = "EM".
BPPARAM	A BiocParallelParam object specifying the parallel backend for distributed clonotype assignment operations (split by group). Default is BiocParallel::SerialParam().
contigs	character. When x is a SingleCellExperiment, this is the name of the column in the colData of x that contains the VDJ contig data.

Details

Assign cells (with at least one V(D)J contig) to clonotypes and produce summary tables that can be used for downstream analysis. Clonotype assignment can be handled in multiple ways depending on the choice of "method":

- "EM": Cells are assigned probabilistically to their most likely clonotype(s) with the Expectation-Maximization (EM) algorithm. For ambiguous cells, this leads to proportional (non-integer) assignment across multiple clonotypes and a frequency table of (non-integer) expected counts.
- "unique": Cells are assigned a clonotype if (and only if) they can be uniquely assigned a single clonotype. For a T cell, this means having exactly one alpha chain and one beta chain.
- "CellRanger": Clonotype labels are taken from contig data and matched across samples.
- column name in contig data: Similar to "unique", but additionally, cells with multiples of a particular chain are assigned a "dominant" clonotype based on which contig has the higher value in this column (typical choices being "umis" or "reads").

- `type of chain in contig data`: Clonotypes are based entirely on this type of chain (eg. `"TRA"` or `"TRB"`) and cells may be assigned to multiple clonotypes, if multiples of that chain are present.

The `"EM"`, `"unique"`, and UMI/read-based quantification methods all define a clonotype as a pair of specific chains (alpha and beta for T cells, heavy and light for B cells). Unlike other methods, the EM algorithm assigns clonotypes probabilistically, which can lead to non-integer counts for cells with ambiguous information (ie. only an alpha chain, or two alphas and one beta chain).

We highly recommend providing information on each cell's sample of origin, as this can speed up computation and provide more accurate results. This is particularly important for the EM algorithm, which shares information across cells in the same group, so splitting by sample can improve accuracy by removing extraneous clonotypes from the set of possibilities for a particular cell.

Value

Returns an object of class `clonoStats`, containing group-level clonotype summaries. May optionally include a sparse matrix of cell-level assignment information, if `assignment = TRUE`. If `x` is a `SingleCellExperiment` object, this output is added to the metadata.

See Also

`clonoStats`

Examples

```
data('contigs')
clonoStats(contigs)
```

---

clonoStats-class `clonoStats` *object class*

Description

The `clonoStats` class is designed to hold the output of the `clonoStats` function. This always includes two group-level summaries: clonotype abundances and clonotype frequencies. "Group" most often refers to sample of origin, but may alternatively refer to any partitioning of cells, such as clusters. Clonotype names are stored efficiently in two factors. Additionally, a large, sparse matrix of each cell's clonotype assignment may be included.

Usage

```
## S4 method for signature 'clonoStats'
show(object)

clonoNames(object)

## S4 method for signature 'clonoStats'
```

```
clonoNames(object)

## S4 method for signature 'SingleCellExperiment'
clonoNames(object)

clonoAbundance(object)

## S4 method for signature 'clonoStats'
clonoAbundance(object)

## S4 method for signature 'SingleCellExperiment'
clonoAbundance(object)

clonoFrequency(object)

## S4 method for signature 'clonoStats'
clonoFrequency(object)

## S4 method for signature 'SingleCellExperiment'
clonoFrequency(object)

clonoAssignment(object)

## S4 method for signature 'clonoStats'
clonoAssignment(object)

## S4 method for signature 'SingleCellExperiment'
clonoAssignment(object)

clonoGroup(object)

## S4 method for signature 'clonoStats'
clonoGroup(object)

## S4 method for signature 'SingleCellExperiment'
clonoGroup(object)
```

Arguments

object	a clonoStats object or SingleCellExperiment object containing clonoStats results.

Value

An object of class clonoStats.

Functions

- show(clonoStats): a short summary of a clonoStats object.

- `clonoNames()`: Get the full clonotype names
- `clonoAbundance()`: Get the clonotype abundance matrix (clonotype counts per group).
- `clonoFrequency()`: Get the matrix of clonotype frequencies (singletons, doubletons, etc. per group).
- `clonoAssignment()`: Get the matrix of cell-level clonotype assignments (mapping cells to clonotypes).
- `clonoGroup()`: Get the factor variable of group labels

Slots

`abundance` Summary table of clonotype abundances within each group (sample). Provides specific information about how often each clonotype is observed in each group.

`frequency` Summary table of clonotype frequencies within each group (sample). Provides a summary of clonotype abundances (ie. number of singletons, doubletons, etc.).

`group` Factor variable giving the group of origin for each cell.

`assignment` Optional matrix of clonotype assignments in each individual cell. Rows correspond to cells and row sums are all 0 or 1. When using `clonoStats` with `method = "unique"` or `method = "CellRanger"`, all values will be 0 or 1. With `method = "EM"`, fractional values are allowed, representing the assignment confidence.

`names1` Factor listing the first component of each clonotype name (alpha chains, for TCR data).

`names2` Factor listing the second component of each clonotype name (beta chains, for TCR data).

See Also

`clonoStats`

Examples

```
data('contigs')
cs <- clonoStats(contigs)
cs
```

---

contigs *SplitDataFrameList containing AIRR-seq (TCR) data for six cells*

Description

Data are a small subset of the TCR-seq data from the paper, "Progressive immune dysfunction with advancing disease stage in renal cell carcinoma" (Braun et al. 2021). The full dataset can be obtained from dbGap phs002252.v1.p1.

Usage

```
data(contigs)
```

Format

A SplitDataFrameList with six elements. Each list element contains the TCR-seq data for a single cell in a `DFrame`. Each `DFrame` has 19 variables and as many rows as there are contigs for the cell.

The variables in the dataset are the same as those in the contig_annotations.csv file created by 10X. The meaning of each variable label is specified at https://support.10xgenomics.com/single-cell-vdj/software/pipelines/latest/output/annotation, but they are also summarized below:

barcode Cell barcode for the contig in the list element.

is_cell True or False value indicating whether the barcode was called as a cell.

contig_id Unique identifier for this contig.

high_confidence True or False value indicating whether the contig was called as high-confidence (unlikely to be a chimeric sequence or some other artifact).

length The contig sequence length in nucleotides.

chain The chain associated with this contig; for example, TRA, TRB, IGK, IGL, or IGH. A value of "Multi" indicates that segments from multiple chains were present.

v_gene The highest-scoring V segment, for example, TRAV1-1.

d_gene The highest-scoring D segment, for example, TRBD1.

j_gene The highest-scoring J segment, for example, TRAJ1-1.

c_gene The highest-scoring C segment, for example, TRAC.

full_length If the contig was declared as full-length.

productive If the contig was declared as productive.

cdr3 The predicted CDR3 amino acid sequence.

cdr3_nt The predicted CDR3 nucleotide sequence.

reads The number of reads aligned to this contig.

umis The number of distinct UMIs aligned to this contig.

raw_clonotype_id The ID of the clonotype to which this cell barcode was assigned.

raw_consensus_id The ID of the consensus sequence to which this contig was assigned.

sample Sample identifier. The data for contigs come from two different samples.

Source

Braun, David A., et al. "Progressive immune dysfunction with advancing disease stage in renal cell carcinoma." Cancer cell 39, no. 5 (2021): 632-648.

Examples

```
data('contigs')
x <- clonoStats(contigs)
```

pieVDJ *Create a pie chart for clonotype expansion*

Description

pieVDJ creates a list of pie charts created using ggplot that shows the the level of expansion in each clonotype.

Usage

```
pieVDJ(x, ...)

## S4 method for signature 'Matrix'
pieVDJ(x, legend = "bottom")

## S4 method for signature 'matrix'
pieVDJ(x, ...)

## S4 method for signature 'clonoStats'
pieVDJ(x, ...)
```

Arguments

x	A matrix created with summarizeClonotypes.
...	additional arguments.
legend	Can take on the values use in the legend.position command in ggplot ("left","top", "right", "bottom", or a numeric vector) to indicate where the legend should be placed. If left NULL, no legend will be created.

Value

If x contains more than one sample, a list of pie charts will be returned. If x contains only one sample, the pie chart will be returned. The coloring indicates the number of cells for each clonotype with darker colors being clonotypes with a single cell (singletons) and lighter colors having more cells with that clonotype (expanded clonotype).

Examples

```
data('contigs')
x <- clonoStats(contigs)
pieVDJ(x)
```

readVDJcontigs *Load 10X CellRanger V(D)J data*

Description

Creates a SplitDataFrameList (see DataFrameList) from a character vector of directory names corresponding to the output of the CellRanger V(D)J pipeline.

Usage

```
readVDJcontigs(samples, ...)

## S4 method for signature 'character'
readVDJcontigs(samples, sample.names = names(samples))
```

Arguments

samples	A character vector containing one or more directory names, each corresponding to a 10X sample. Each directory should contain a file named filtered_contig_annotations.csv.
...	additional arguments.
sample.names	A character vector of length equal to samples containing the sample names to store in the output object. If NULL, the basenames of each directory will be used.

Details

The resulting list of DataFrames contains all the data in filtered_contig_annotations.csv, split by cell barcode. Note that the index of each sample in samples is concatenated to the cell barcodes, so that cells from different samples cannot have identical barcodes.

Value

A SplitDataFrameList object containing data on all contigs, grouped by cell barcode.

Examples

```
# write the example data to a temporary directory
example(writeVDJcontigs)
# specify sample locations and read in data
samples <- file.path(loc, c('sample1','sample2'))
contigs <- readVDJcontigs(samples)
```

runBreakaway *Clonotype richness estimation with Breakaway*

Description

This function uses the Breakaway method to estimate the clonotype richness (total number of clonotypes) present in each group of a clonoStats object.

Usage

```
runBreakaway(x, ...)

## S4 method for signature 'clonoStats'
runBreakaway(x, nof1 = FALSE, ...)
```

Arguments

x	A clonoStats object.
...	Additional arguments passed to breakaway or breakaway_nof1.
nof1	logical. Indicates whether to use the breakaway_nof1 function, for abundance data that may contain spurious singletons.

Value

A list of alpha_estimate objects, one per group, containing detailed results of running the Breakaway estimator on the vector of clonotype frequencies from that group.

References

Willis, A. and Bunge, J. (2015). Estimating diversity via frequency ratios. *Biometrics*.

Willis, A. (2015). Species richness estimation with high diversity but spurious singletons. *arXiv*.

Examples

```
data('contigs')
x <- clonoStats(contigs, method = 'unique')
runBreakaway(x)
```

runVDJPCA *Run PCA on clonotype abundance matrix*

Description

Perform Principal Components Analysis (PCA) on the matrix of sample-level clonotype abundances. In the context of clonotype analysis, this is a form of beta diversity.

Usage

```
runVDJPCA(x, ...)

## S4 method for signature 'clonoStats'
runVDJPCA(x, unit = c("samples", "clonotypes"))
```

Arguments

x	A matrix of abundance values where rows are features (clonotypes) and columns are samples.
...	additional arguments.
unit	Character value indicating whether the unit of interest is "samples" or "clonotypes".

Value

A list with class "prcomp". The component x stores the reduced-dimensional representation of the data. For a full description, see prcomp.

Examples

```
data('contigs')
x <- clonoStats(contigs)
runVDJPCA(x)
```

scatterVDJ *Create a scatterplot for diversity evenness and abundance*

Description

scatterVDJ creates a scatterplot that shows the abundance of the sample on the x-axis and the evenness on the y-axis.

Usage

```
scatterVDJ(d, ...)

## S4 method for signature 'matrix'
scatterVDJ(d, sampleGroups = NULL, title = NULL, legend = FALSE)
```

Arguments

d	A matrix created with calculateDiversity. The matrix must include nClonotypes and normentropy.
...	additional arguments.
sampleGroups	A matrix or data.frame that identifies the groups that each sample belongs to. The matrix must contain two columns. The first column lists the individual samples and should be called "Sample". The second column should list the group that each sample belongs to (e.g. Normal and Tumor) and be called "Group". If no sampleGroups dataset is provided, all of the samples will be plotted in the same color.
title	Character vector with an optional title.
legend	If TRUE, a legend will be included with the plot. If FALSE, no legend is included in the plot.

Value

Returns a ggplot plot with a scatterplot that shows the abundance for each sample on the x-axis and the evenness for each sample on the y-axis. Richness can be expressed as the total number of unique clonotypes in the sample or as the breakaway diversity measure (Willis and Bunge 2015), which estimates the total number of unique clonotypes in the population. Evenness is measured as the normalized entropy, which is a measure of how evenly cells are distributed across the different clonotypes. Evenness is a measure between 0 and 1 that is independent of the number of cells in a sample. Diversity measures such as Shannon entropy contain information about both the evenness and the abundance of a sample, but because both characteristics are combined into one number, comparison between samples or groups of samples is difficult. Other measures, such as the breakaway measure of diversity, only express the abundance of the sample and not the evenness. The scatterplot shows how evenness and abundance differs between each sample and between each group of samples.

Examples

```
data('contigs')
x <- clonoStats(contigs)
d <- calculateDiversity(x)
sampleGroups <- data.frame(Sample = c("sample1", "sample2"),
                           Group = c("Cancer", "Normal"))
scatterVDJ(d, sampleGroups = NULL,
       title = "Evenness-abundance plot", legend = TRUE)
```

splitClonotypes *Split cell-level clonotype counts by sample*

Description

Takes a matrix of cell-level clonotype counts and splits them into a list of group-specific counts (typically samples).

Usage

```
splitClonotypes(x, by, ...)

## S4 method for signature 'Matrix'
splitClonotypes(x, by)

## S4 method for signature 'matrix'
splitClonotypes(x, by)

## S4 method for signature 'SingleCellExperiment'
splitClonotypes(x, by, contigs = "contigs", clonoStats = "clonoStats")

## S4 method for signature 'clonoStats'
splitClonotypes(x, by)
```

Arguments

x	A Matrix of cell-level clonotype assignments (cells-by-clonotypes) or a SingleCellExperiment object with such a matrix stored in the clono slot of the colData.
by	A character vector or factor by which to split the clonotype counts. If x is a SingleCellExperiment object, this can also be a character, giving the name of the column from the colData to use as this variable. Similar to the group argument for clonoStats.
...	additional arguments.
contigs	character. If x is a SingleCellExperiment, the name of the SplitDataFrameList in the colData of x containing contig information.
clonoStats	character. If x is a SingleCellExperiment, the name of the element in the metadata of x that contains the output of clonoStats. Must include cell-level clonotype assignments (ie. assignment = TRUE).

Value

A list of Matrix objects providing the cell-level clonotype assignments for each unique value of by (if by denotes sample labels, each matrix in the list will contain the cells from a single sample).

Examples

```
example(addVDJtoSCE)
x <- clonoStats(contigs, assignment = TRUE)
splitClonotypes(x, by = sce$sample)
```

`summarizeClonotypes` *Get sample-level clonotype counts*

Description

Takes a matrix of cell-level clonotype counts and sums them within groups (typically samples).

Usage

```
summarizeClonotypes(x, by, ...)

## S4 method for signature 'Matrix'
summarizeClonotypes(
  x,
  by,
  mode = c("sum", "tab"),
  lang = c("r", "cpp"),
  BPPARAM = SerialParam()
)

## S4 method for signature 'SingleCellExperiment'
summarizeClonotypes(
  x,
  by = "sample",
  contigs = "contigs",
  clonoStats = "clonoStats",
  ...
)

## S4 method for signature 'matrix'
summarizeClonotypes(x, by, ...)

## S4 method for signature 'clonoStats'
summarizeClonotypes(x, by, ...)
```

Arguments

x	A (usually sparse) matrix of cell-level clonotype counts (cells are rows and clonotypes are columns). Alternatively, a `SingleCellExperiment` with such a matrix stored in the `colData`.

by	A character vector or factor by which to summarize the clonotype counts. If x is a SingleCellExperiment object, this can also be a character, giving the name of the column from the colData to use as this variable. Similar to the group argument for clonoStats.
...	additional arguments.
mode	Type of summarization to perform. Default is 'sum', which sums clonotype abundances within each sample (or level of 'by'). Alternative is 'tab', which constructs a table of clonotype frequencies (ie. singletons, doubletons, etc.) by sample.
lang	Indicates which implementation of the "tab" summarization to use. Options are 'r' (default) or 'cpp'. Only used if non-integer clonotype abundances are present and mode = "tab".
BPPARAM	A BiocParallelParam object specifying the parallel backend for distributed clonotype assignment operations (split by group). Default is BiocParallel::SerialParam().
contigs	character. If x is a SingleCellExperiment, the name of the SplitDataFrameList in the colData of x containing contig information.
clonoStats	character. If x is a SingleCellExperiment, the name of the element in the metadata of x that contains the output of clonoStats. Must include cell-level clonotype assignments (ie. assignment = TRUE).

Value

If mode = 'sum', returns a matrix clonotype abundances where each row corresponds to a clonotype and each column a value of by (if by denotes sample labels, this is a matrix of sample-level clonotype counts). If mode = 'tab', returns a matrix of clonotype frequencies, where each row corresponds to a frequency (singletons, doubletons, etc.) and each column a value of by.

Examples

```
example(addVDJtoSCE)
x <- clonoStats(contigs, assignment = TRUE)
summarizeClonotypes(x, by = sce$sample)
```

---

writeVDJcontigs *Write V(D)J contig data in 10X format*

Description

Write V(D)J data to a series of directories, each containing a CSV file with the data for an individual sample.

Usage

```
writeVDJcontigs(path, x, ...)

## S4 method for signature 'character,SplitDataFrameList'
writeVDJcontigs(path, x)
```

Arguments

path	A string containing the path to the output directory.
x	A SplitDataFrameList object containing V(D)J contig information, split by cell barcodes, as created by readVDJcontigs.
...	additional arguments.

Value

Creates various subdirectories of the directory specified in path. Each subdirectory is named for a sample found in the dataset, x, and contains a CSV filed named filtered_contig_annotations.csv.

Examples

```
data('contigs')
loc <- tempdir()
writeVDJcontigs(loc, contigs)
```

Index

∗ **datasets**
contigs, 12

abundanceVDJ, 2
abundanceVDJ,clonoStats-method *(abundanceVDJ)*, 2
addVDJtoSCE, 3
addVDJtoSCE,character,SingleCellExperiment-method *(addVDJtoSCE)*, 3
addVDJtoSCE,SplitDataFrameList,SingleCellExperiment-method *(addVDJtoSCE)*, 3

barVDJ, 5
barVDJ,clonoStats-method *(barVDJ)*, 5
barVDJ,Matrix-method *(barVDJ)*, 5
barVDJ,matrix-method *(barVDJ)*, 5
BiocParallelParam, *9*, *21*
boxVDJ, 6
boxVDJ,matrix-method *(boxVDJ)*, 6

calculateDiversity, 7
calculateDiversity,clonoStats-method *(calculateDiversity)*, 7
calculateDiversity,SingleCellExperiment-method *(calculateDiversity)*, 7
clonoAbundance *(clonoStats-class)*, 10
clonoAbundance,clonoStats-method *(clonoStats-class)*, 10
clonoAbundance,SingleCellExperiment-method *(clonoStats-class)*, 10
clonoAssignment *(clonoStats-class)*, 10
clonoAssignment,clonoStats-method *(clonoStats-class)*, 10
clonoAssignment,SingleCellExperiment-method *(clonoStats-class)*, 10
clonoFrequency *(clonoStats-class)*, 10
clonoFrequency,clonoStats-method *(clonoStats-class)*, 10
clonoFrequency,SingleCellExperiment-method *(clonoStats-class)*, 10
clonoGroup *(clonoStats-class)*, 10
clonoGroup,clonoStats-method *(clonoStats-class)*, 10
clonoGroup,SingleCellExperiment-method *(clonoStats-class)*, 10
clonoNames *(clonoStats-class)*, 10
clonoNames,clonoStats-method *(clonoStats-class)*, 10
clonoNames,SingleCellExperiment-method *(clonoStats-class)*, 10
clonoStats, 8, *10–12*, *16*, *19*, *21*
clonoStats,clonoStats-method *(clonoStats)*, 8
clonoStats,SingleCellExperiment-method *(clonoStats)*, 8
clonoStats,SplitDataFrameList-method *(clonoStats)*, 8
clonoStats-class, 10
contigs, 12

DataFrameList, *15*

pieVDJ, 14
pieVDJ,clonoStats-method *(pieVDJ)*, 14
pieVDJ,Matrix-method *(pieVDJ)*, 14
pieVDJ,matrix-method *(pieVDJ)*, 14
prcomp, *17*

read10xCounts, *4*
readVDJcontigs, *4*, 15
readVDJcontigs,character-method *(readVDJcontigs)*, 15
runBreakaway, 16
runBreakaway,clonoStats-method *(runBreakaway)*, 16
runVDJPCA, 17
runVDJPCA,clonoStats-method *(runVDJPCA)*, 17

scatterVDJ, 17

scatterVDJ,matrix-method *(scatterVDJ)*,
17
show,clonoStats-method
(clonoStats-class), 10
SingleCellExperiment, *3, 4*, *20*
splitClonotypes, 19
splitClonotypes,clonoStats-method
(splitClonotypes), 19
splitClonotypes,Matrix-method
(splitClonotypes), 19
splitClonotypes,matrix-method
(splitClonotypes), 19
splitClonotypes,SingleCellExperiment-method
(splitClonotypes), 19
summarizeClonotypes, 20
summarizeClonotypes,clonoStats-method
(summarizeClonotypes), 20
summarizeClonotypes,Matrix-method
(summarizeClonotypes), 20
summarizeClonotypes,matrix-method
(summarizeClonotypes), 20
summarizeClonotypes,SingleCellExperiment-method
(summarizeClonotypes), 20

writeVDJcontigs, 21
writeVDJcontigs,character,SplitDataFrameList-method
(writeVDJcontigs), 21