Package ‘scRepertoire’

May 4, 2024

Title A toolkit for single-cell immune receptor profiling

Version 2.0.0

Description
scRepertoire is a toolkit for processing and analyzing single-cell T-cell receptor (TCR) and immunoglobulin (Ig). The scRepertoire framework supports use of 10x, AIRR, BD, MiXCR, Omniscope, TRUST4, and WAT3R single-cell formats. The functionality includes basic clonal analyses, repertoire summaries, distance-based clustering and interaction with the popular Seurat and SingleCellExperiment/Bioconductor R workflows.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.1

biocViews Software, ImmunoOncology, SingleCell, Classification, Annotation, Sequencing

Depends ggplot2, R (>= 4.0)

Imports cubature, dplyr, evmix, ggalluvial, ggdendro, ggraph, grDevices, igraph, iNEXT, methods, plyr, quantreg, Rcpp, reshape2, rjson, rlang, S4Vectors, SeuratObject, SingleCellExperiment, stats, stringr, stringdist, SummarizedExperiment, tidygraph, truncdist, utils, VGAM, hash

Suggests BiocManager, BiocStyle, circlize, knitr, rmarkdown, scales, scater, Seurat, spelling, testthat (>= 3.0.0), vdiffr

VignetteBuilder knitr

Config/testthat/edition 3

Language en-US

LinkingTo Rcpp


BugReports https://github.com/ncborcherding/scRepertoire/issues

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

git_branch RELEASE_3_19
Contents

scRepertoire-package .................................................. 3
addVariable ................................................................. 3
alluvialClones ............................................................. 4
clonalAbundance .......................................................... 6
clonalBias ................................................................. 7
clonalCluster ............................................................. 8
clonalCompare ........................................................... 9
clonalDiversity ........................................................... 11
clonalHomeostasis ......................................................... 13
clonalLength .............................................................. 14
clonalNetwork ............................................................. 15
clonalOccupy .............................................................. 16
clonalOverlap ............................................................ 18
clonalOverlay ............................................................. 19
clonalProportion ........................................................ 20
clonalQuant ............................................................... 21
clonalRarefaction ......................................................... 22
clonalScatter ............................................................. 24
clonalSizeDistribution .................................................. 25
combineBCR ............................................................... 27
combineExpression ....................................................... 28
combineTCR ............................................................... 29
contig_list ............................................................... 30
createHTOContigList .................................................... 31
exportClones ............................................................. 32
getCirclize ............................................................... 33
highlightClones .......................................................... 34
loadContigs ............................................................... 35
mini_contig_list .......................................................... 36
percentAA ................................................................. 36
percentGenes ............................................................ 37
percentKmer .............................................................. 38
percentVJ ................................................................. 39
positionalVJ .............................................................. 39
positionalEntropy ......................................................... 40
positionalProperty ....................................................... 41
Description

scRepertoire is a toolkit for processing and analyzing single-cell T-cell receptor (TCR) and immunoglobulin (Ig). The scRepertoire framework supports use of 10x, AIRR, BD, MiXCR, Omniscope, TRUST4, and WAT3R single-cell formats. The functionality includes basic clonal analyses, repertoire summaries, distance-based clustering and interaction with the popular Seurat and SingleCellExperiment/Bioconductor R workflows.

Author(s)

Maintainer: Nick Borcherding <ncborch@gmail.com>
Authors:
- Qile Yang <qile0317@gmail.com>
- Ksenia Safina <safina@broadinstitute.org>

See Also

Useful links:
- Report bugs at [https://github.com/ncborcherding/scRepertoire/issues](https://github.com/ncborcherding/scRepertoire/issues)

addVariable

Adding variables after combineTCR() or combineBCR()

Description

This function adds variables to the product of combineTCR, or combineBCR to be used in later visualizations. For each element, the function will add a column (labeled by variable.name) with the variable. The length of the variables parameter needs to match the length of the combined object.

Usage

addVariable(input.data, variable.name = NULL, variables = NULL)
Arguments

input.data  The product of `combineTCR` or `combineBCR`.
variable.name  The new column name/header.
variables  The exact values to add to each element of the list.

Value

input.data list with the variable column added to each element.

Examples

combined <- combineTCR(contig_list,
combined <- addVariable(combined,
  variable.name = "Type",
  variables = rep(c("B", "L"), 4))

alluvialClones  Alluvial plotting for single-cell object meta data

Description

View the proportional contribution of clones by Seurat or SCE object meta data after `combineExpression`. The visualization is based on the ggalluvial package, which requires the aesthetics to be part of the axes that are visualized. Therefore, alpha, facet, and color should be part of the the axes you wish to view or will add an additional stratum/column to the end of the graph.

Usage

alluvialClones(
  sc.data,
  cloneCall = "strict",
  chain = "both",
  y.axes = NULL,
  color = NULL,
  alpha = NULL,
  facet = NULL,
  exportTable = FALSE,
  palette = "inferno"
)
**alluvialClones**

**Arguments**

- **sc.data**: The single-cell object to visualize after `combineExpression`.
- **cloneCall**: How to call the clone - VDJC gene (**gene**), CDR3 nucleotide (**nt**), CDR3 amino acid (**aa**), VDJC gene + CDR3 nucleotide (**strict**) or a custom variable in the data.
- **chain**: Indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".
- **y.axes**: The columns that will separate the proportional visualization.
- **color**: The column header or clone(s) to be highlighted.
- **alpha**: The column header to have gradated opacity.
- **facet**: The column label to separate.
- **exportTable**: Exports a table of the data into the global environment in addition to the visualization.
- **palette**: Colors to use in visualization - input any hcl.pals.

**Value**

Alluvial ggplot comparing clone distribution.

**Examples**

```r
# Getting the combined contigs

# Getting a sample of a Seurat object
scRep_example <- get(data("scRep_example"))

# Using combineExpression()
scRep_example <- combineExpression(combined, scRep_example)
scRep_example$Patient <- substring(scRep_example$orig.ident, 1,3)

# Using alluvialClones()
alluvialClones(scRep_example,
               cloneCall = "gene",
               y.axes = c("Patient", "ident"),
               color = "ident")
```
clonalAbundance  

Demonstrate the relative abundance of clones by group or sample

Description
Displays the number of clones at specific frequencies by sample or group. Visualization can either be a line graph (scale = FALSE) using calculated numbers or density plot (scale = TRUE). Multiple sequencing runs can be group together using the group parameter. If a matrix output for the data is preferred, set exportTable = TRUE.

Usage

clonalAbundance(
  input.data,
  cloneCall = "strict",
  chain = "both",
  scale = FALSE,
  group.by = NULL,
  exportTable = FALSE,
  palette = "inferno"
)

Arguments

input.data  The product of combineTCR, combineBCR, or combineExpression.
cloneCall  How to call the clone - VDJC gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), VDJC gene + CDR3 nucleotide (strict) or a custom variable in the data.
chain  indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL"
scale  Converts the graphs into density plots in order to show relative distributions.
group.by  The variable to use for grouping
exportTable  Returns the data frame used for forming the graph to the visualization.
palette  Colors to use in visualization - input any hcl.pals.

Value
ggplot of the total or relative abundance of clones across quanta

Examples

#Making combined contig data
combined <- combineTCR(contig_list,
            "P19B","P19L", "P20B", "P20L"))
clonalAbundance(combined,
clonalBias

Examine skew of clones towards a cluster or compartment

Description

The metric seeks to quantify how individual clones are skewed towards a specific cellular compartment or cluster. A clone bias of 1 - indicates that a clone is composed of cells from a single compartment or cluster, while a clone bias of 0 - matches the background subtype distribution. Please read and cite the following manuscript if using clonalBias.

Usage

```r
clonalBias(
  sc.data,
  cloneCall = "strict",
  split.by = NULL,
  group.by = NULL,
  n.boots = 20,
  min.expand = 10,
  exportTable = FALSE,
  palette = "inferno"
)
```

Arguments

- **sc.data**: The single-cell object after `combineExpression`.
- **cloneCall**: How to call the clone - VDJC gene (`gene`), CDR3 nucleotide (`nt`), CDR3 amino acid (`aa`), VDJC gene + CDR3 nucleotide (`strict`) or a custom variable in the data.
- **split.by**: The variable to use for calculating the baseline frequencies. For example, "Type" for lung vs peripheral blood comparison.
- **group.by**: The variable to use for calculating bias.
- **n.boots**: Number of bootstraps to downsample.
- **min.expand**: Clone frequency cut off for the purpose of comparison.
- **exportTable**: Returns the data frame used for forming the graph.
- **palette**: Colors to use in visualization - input any `hcl.pals`.

Value

`ggplot` scatter plot with clone bias
Examples

# Making combined contig data
combined <- combineTCR(contig_list,
            "P19B","P19L", "P20B", "P20L"))

# Getting a sample of a Seurat object
scRep_example <- get(data("scRep_example"))

# Using combineExpression()
scRep_example <- combineExpression(combined, scRep_example)
scRep_example$Patient <- substring(scRep_example$orig.ident,1,3)

# Using clonalBias()
clonalBias(scRep_example,
    cloneCall = "aa",
    split.by = "Patient",
    group.by = "seurat_clusters",
    n.boots = 5,
    min.expand = 2)

---

clonalCluster  
Clustering adaptive receptor sequences by edit distance

Description

This function uses edit distances of either the nucleotide or amino acid sequences of the CDR3 and V genes to cluster similar TCR/BCRs together. As a default, the function takes the input from combineTCR, combineBCR or combineExpression and amends a cluster to the data frame or meta data. If exportGraph is set to TRUE, the function returns an igraph object of the connected sequences.

Usage

clonalCluster(
    input.data,
    chain = "TRB",
    sequence = "aa",
    samples = NULL,
    threshold = 0.85,
    group.by = NULL,
    exportGraph = FALSE
)
**clonalCompare**  

**Arguments**

- **input.data**: The product of `combineTCR`, `combineBCR` or `combineExpression`.
- **chain**: Indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".
- **sequence**: Clustering based on either "aa" or "nt".
- **samples**: The specific samples to isolate for visualization.
- **threshold**: The normalized edit distance to consider. The higher the number the more similarity of sequence will be used for clustering.
- **group.by**: The column header used for to group contigs. If (NULL), clusters will be calculated across samples.
- **exportGraph**: Return an igraph object of connected sequences (TRUE) or the amended input with a new cluster-based variable (FALSE).

**Value**

Either amended input with edit-distanced clusters added or igraph object of connect sequences

**Examples**

```r
# Getting the combined contigs
combined <- combineTCR(contig_list,

sub_combined <- clonalCluster(combined[c(1,2)],
    chain = "TRA",
    sequence = "aa")
```

```
clonalCompare  

Demonstrate the difference in clonal proportion between clones

**Description**

This function produces an alluvial or area graph of the proportion of the indicated clones for all or selected samples (using the `samples` parameter). Individual clones can be selected using the `clones` parameter with the specific sequence of interest or using the `top.clones` parameter with the top n clones by proportion to be visualized.

**Usage**

```r
clonalCompare(
    input.data,
    cloneCall = "strict",
    chain = "both",
    samples = NULL,
```
clonalCompare

clones = NULL,
top.clones = NULL,
highlight.clones = NULL,
relabel.clones = FALSE,
group.by = NULL,
graph = "alluvial",
exportTable = FALSE,
palette = "inferno"
)

Arguments

input.data The product of combineTCR, combineBCR, or combineExpression.
cloneCall How to call the clone - VDJC gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), VDJC gene + CDR3 nucleotide (strict) or a custom variable in the data.
chain Indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".
samples The specific samples to isolate for visualization.
clones The specific clonal sequences of interest.
top.clones The top number of clonal sequences per group. (e.g., top.clones = 5)
highlight.clones Clonal sequences to highlight, if present, all other clones returned will be grey.
relabel.clones Simplify the legend of the graph by returning clones that are numerically indexed.
group.by If using a single-cell object, the column header to group the new list. NULL will return the active identity or cluster.
graph The type of graph produced, either "alluvial" or "area".
exportTable Returns the data frame used for forming the graph.
palette Colors to use in visualization - input any hcl.pals.

Value

ggplot of the proportion of total sequencing read of selecting clones

Examples

#Making combined contig data
combined <- combineTCR(contig_list,
"P19B","P19L", "P20B", "P20L"))
clonalCompare(combined,
top.clones = 5,
samples = c("P17B", "P17L"),
cloneCall="aa")
clonalDiversity

Calculate the clonal diversity for samples or groupings

Description

This function calculates traditional measures of diversity - Shannon, inverse Simpson, normalized entropy, Gini-Simpson, Chao1 index, and abundance-based coverage estimators (ACE) measure of species evenness by sample or group. The function automatically down samples the diversity metrics using 100 bootstraps (n.boots = 100) and outputs the mean of the values. The group parameter can be used to condense the individual samples. If a matrix output for the data is preferred, set exportTable = TRUE.

Usage

clonalDiversity(
  input.data,
  cloneCall = "strict",
  chain = "both",
  group.by = NULL,
  x.axis = NULL,
  metrics = c("shannon", "inv.simpson", "norm.entropy", "gini.simpson", "chao1", "ACE"),
  exportTable = FALSE,
  palette = "inferno",
  n.boots = 100,
  return.boots = FALSE,
  skip.boots = FALSE
)

Arguments

input.data The product of combineTCR, combineBCR, or combineExpression.
cloneCall How to call the clone - VDJC gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), VDJC gene + CDR3 nucleotide (strict) or a custom variable in the data.
chain indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".
group.by Variable in which to combine for the diversity calculation.
x.axis Additional variable grouping that will space the sample along the x-axis.
metrics The indices to use in diversity calculations - "shannon", "inv.simpson", "norm.entropy", "gini.simpson", "chao1", "ACE".
exportTable Exports a table of the data into the global environment in addition to the visualization.
palette Colors to use in visualization - input any hcl.pals.
n.boots number of bootstraps to down sample in order to get mean diversity.
return.boots export boot strapped values calculated - will automatically exportTable = TRUE.
skip.boots remove down sampling and boot strapping from the calculation.
Details

The formulas for the indices and estimators are as follows:

**Shannon Index:**

\[
Index = - \sum p_i \log(p_i)
\]

**Inverse Simpson Index:**

\[
Index = \frac{1}{\left( \sum_{i=1}^{S} p_i^2 \right)}
\]

**Normalized Entropy:**

\[
Index = - \frac{\sum_{i=1}^{S} p_i \ln(p_i)}{\ln(S)}
\]

**Gini-Simpson Index:**

\[
Index = 1 - \sum_{i=1}^{S} p_i^2
\]

**Chao1 Index:**

\[
Index = S_{obs} + \frac{n_1(n_1 - 1)}{2 \times n_2 + 1}
\]

**Abundance-based Coverage Estimator (ACE):**

\[
Index = S_{abund} + \frac{S_{rare}}{C_{ace}} + \frac{F_1}{C_{ace}}
\]

Where:

* \(p_i\) is the proportion of species \(i\) in the dataset.
* \(S\) is the total number of species.
* \(n_1\) and \(n_2\) are the number of singletons and doubletons, respectively.
* \(S_{abund}, S_{rare}, C_{ace}, \text{ and } F_1\) are parameters derived from the data.

Value

ggplot of the diversity of clones by group

Author(s)

Andrew Malone, Nick Borcherding

Examples

```r
#Making combined contig data
combined <- combineTCR(contig_list,
clonalDiversity(combined, cloneCall = "gene")
```
clonalHomeostasis

Examining the clonal homeostasis of the repertoire

Description

This function calculates the space occupied by clone proportions. The grouping of these clones is based on the parameter `cloneSize`, at default, `cloneSize` will group the clones into bins of Rare = 0 to 0.0001, Small = 0.0001 to 0.001, etc. To adjust the proportions, change the number or labeling of the cloneSize parameter. If a matrix output for the data is preferred, set `exportTable = TRUE`.

Usage

```r
clonalHomeostasis(
  input.data,
  cloneSize = c(Rare = 1e-04, Small = 0.001, Medium = 0.01, Large = 0.1, Hyperexpanded = 1),
  cloneCall = "strict",
  chain = "both",
  group.by = NULL,
  exportTable = FALSE,
  palette = "inferno"
)
```

Arguments

- `input.data` The product of `combineTCR`, `combineBCR`, or `combineExpression`.
- `cloneSize` The cut points of the proportions.
- `cloneCall` How to call the clone - VDJC gene (`gene`), CDR3 nucleotide (`nt`), CDR3 amino acid (`aa`), VDJC gene + CDR3 nucleotide (`strict`) or a custom variable in the data.
- `chain` indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".
- `group.by` The variable to use for grouping.
- `exportTable` Exports a table of the data into the global environment in addition to the visualization.
- `palette` Colors to use in visualization - input any `hcl.pals`.

Value

`ggplot` of the space occupied by the specific proportion of clones
Examples

#Making combined contig data
combined <- combineTCR(contig_list,
clonalHomeostasis(combined, cloneCall = "gene")

clonalLength

Demonstrate the distribution of clonal length

Description

This function displays either the nucleotide (nt) or amino acid (aa) sequence length. The sequence length visualized can be selected using the chains parameter, either the combined clone (both chains) or across all single chains. Visualization can either be a histogram or if scale = TRUE, the output will be a density plot. Multiple sequencing runs can be group together using the group.by parameter.

Usage

clonalLength(
  input.data, 
  cloneCall = "aa", 
  chain = "both", 
  group.by = NULL, 
  scale = FALSE, 
  exportTable = FALSE, 
  palette = "inferno"
)

Arguments

input.data The product of combineTCR, combineBCR, or combineExpression.
cloneCall How to call the clone - CDR3 nucleotide (nt) or CDR3 amino acid (aa).
chain indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".
group.by The variable to use for grouping.
scale Converts the graphs into density plots in order to show relative distributions.
exportTable Returns the data frame used for forming the graph.
palette Colors to use in visualization - input any hcl.pals.

Value

ggplot of the discrete or relative length distributions of clone sequences
Examples

```r
# Making combined contig data
combined <- combineTCR(contig_list,
clonalLength(combined, cloneCall="aa", chain = "both")
```

---

**clonalNetwork**  
*Visualize clonal network along reduced dimensions*

Description

This function generates a network based on clonal proportions of an indicated identity and then superimposes the network onto a single-cell object dimensional reduction plot.

Usage

```r
clonalNetwork(
  sc.data,
  reduction = "umap",
  group.by = "ident",
  filter.clones = NULL,
  filter.identity = NULL,
  filter.proportion = NULL,
  filter.graph = FALSE,
  cloneCall = "strict",
  chain = "both",
  exportTable = FALSE,
  exportClones = FALSE,
  palette = "inferno"
)
```

Arguments

- **sc.data**  
The single-cell object after `combineExpression`.
- **reduction**  
The name of the dimensional reduction of the single-cell object.
- **group.by**  
The variable to use for the nodes.
- **filter.clones**  
Use to select the top n clones (e.g., `filter.clones = 2000`) or n of clones based on the minimum number of all the comparators (e.g., `filter.clones = "min"`).
- **filter.identity**  
Display the network for a specific level of the indicated identity.
- **filter.proportion**  
Remove clones from the network below a specific proportion.
- **filter.graph**  
Remove the reciprocal edges from the half of the graph, allowing for cleaner visualization.
cloneCall  How to call the clone - VDJC gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), VDJC gene + CDR3 nucleotide (strict) or a custom variable in the data.

chain  indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".

exportTable  Exports a table of the data into the global environment in addition to the visualization.

exportClones  Exports a table of clones that are shared across multiple identity groups and ordered by the total number of clone copies.

palette  Colors to use in visualization - input any hcl.pals.

Value

ggplot object

Examples

## Not run:
#Getting the combined contigs
combined <- combineTCR(contig_list,

#Getting a sample of a Seurat object
scRep_example <- get(data("scRep_example"))

#Using combineExpression()
scRep_example <- combineExpression(combined, scRep_example)

#Using clonalNetwork()
clonalNetwork(scRep_example,
    reduction = "umap",
    group.by = "seurat_clusters")

## End(Not run)

clonalOccupy  Visualize the number of single cells with cloneSizes by cluster

Description

View the count of clones frequency group in Seurat or SCE object meta data after combineExpression. The visualization will take the new meta data variable "cloneSize" and plot the number of cells with each designation using a secondary variable, like cluster. Credit to the idea goes to Drs. Carmona and Andreatta and their work with ProjectTIL.
**clonalOccupy**

**Usage**

```r
clonalOccupy(
  sc.data,  
  x.axis = "ident",  
  label = TRUE,  
  facet.by = NULL,  
  proportion = FALSE,  
  na.include = FALSE,  
  exportTable = FALSE,  
  palette = "inferno"
)
```

**Arguments**

- **sc.data**  
The single-cell object after `combineExpression`.
- **x.axis**  
The variable in the meta data to graph along the x.axis.
- **label**  
Include the number of clone in each category by x.axis variable.
- **facet.by**  
The column header used for faceting the graph.
- **proportion**  
Convert the stacked bars into relative proportion.
- **na.include**  
Visualize NA values or not.
- **exportTable**  
Exports a table of the data into the global environment in addition to the visualization.
- **palette**  
Colors to use in visualization - input any hcl.pals.

**Value**

Stacked bar plot of counts of cells by clone frequency group

**Examples**

```r
#Getting the combined contigs
combined <- combineTCR(contig_list, 
  "P19B","P19L", "P20B", "P20L"))

#Getting a sample of a Seurat object
scRep_example <- get(data("scRep_example"))

#Using combineExpression()
scRep_example <- combineExpression(combined, scRep_example)

#Using clonalOccupy
clonalOccupy(scRep_example, x.axis = "ident")
table <- clonalOccupy(scRep_example, x.axis = "ident", exportTable = TRUE)
```
clonalOverlap

Examining the clonal overlap between groups or samples

Description

This functions allows for the calculation and visualizations of various overlap metrics for clones. The methods include overlap coefficient (\textit{overlap}), Morisita's overlap index (\textit{morisita}), Jaccard index (\textit{jaccard}), cosine similarity (\textit{cosine}) or the exact number of clonal overlap (\textit{raw}).

Usage

```r
clonalOverlap(
  input.data, 
  cloneCall = "strict",
  method = NULL,
  chain = "both",
  group.by = NULL,
  exportTable = FALSE,
  palette = "inferno"
)
```

Arguments

- \texttt{input.data} The product of \texttt{combineTCR}, \texttt{combineBCR}, or \texttt{combineExpression}.
- \texttt{cloneCall} How to call the clone - VDJC gene (\texttt{gene}), CDR3 nucleotide (\texttt{nt}), CDR3 amino acid (\texttt{aa}), VDJC gene + CDR3 nucleotide (\texttt{strict}) or a custom variable in the data.
- \texttt{method} The method to calculate the "overlap", "morisita", "jaccard", "cosine" indices or "raw" for the base numbers.
- \texttt{chain} indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL"
- \texttt{group.by} The variable to use for grouping.
- \texttt{exportTable} Returns the data frame used for forming the graph.
- \texttt{palette} Colors to use in visualization - input any hcl.pals.

Details

The formulas for the indices are as follows:

\textbf{Overlap Coefficient:}

\[
\text{overlap} = \frac{\sum \min(a, b)}{\min(\sum a, \sum b)}
\]

\textbf{Raw Count Overlap:}

\[
\text{raw} = \sum \min(a, b)
\]
Morisita Index:
\[
morisita = \frac{\sum ab}{(\sum a)(\sum b)}
\]

Jaccard Index:
\[
jaccard = \frac{\sum \min(a,b)}{\sum a + \sum b - \sum \min(a,b)}
\]

Cosine Similarity:
\[
cosine = \frac{\sum ab}{\sqrt{(\sum a^2)(\sum b^2)}}
\]

Where:
- \( a \) and \( b \) are the abundances of species \( i \) in groups A and B, respectively.

Value

ggplot of the overlap of clones by group

Examples

#Making combined contig data
combined <- combineTCR(contig_list,

clonalOverlap(combined,
cloneCall = "aa",
method = "jaccard")

clonalOverlay

Visualize distribution of clonal frequency overlaid on dimensional reduction plots

Description

This function allows the user to visualize the clonal expansion by overlaying the cells with specific clonal frequency onto the dimensional reduction plots in Seurat. Credit to the idea goes to Drs Andreatta and Carmona and their work with ProjectTIL.

Usage

clonalOverlay(
sc.data,
reduction = NULL,
freq.cutpoint = 30,
bins = 25,
facet.by = NULL
)
Arguments

- **sc.data**: The single-cell object after `combineExpression`.
- **reduction**: The dimensional reduction to visualize.
- **freq.cutpoint**: The overlay cut point to include, this corresponds to the Frequency variable in the single-cell object.
- **bins**: The number of contours to the overlay.
- **facet.by**: The meta data variable to facet the comparison.

Value

`ggplot` object

Author(s)

Francesco Mazziotta, Nick Borcherding

Examples

```r
# Getting the combined contigs
combined <- combineTCR(contig_list,

# Getting a sample of a Seurat object
scRep_example <- get(data("scRep_example"))

# Using combineExpression()
scRep_example <- combineExpression(combined,
                                  scRep_example)

# Using clonalOverlay()
clonalOverlay(scRep_example,
             reduction = "umap",
             freq.cutpoint = 0.3,
             bins = 5)
```

**clonalProportion**

Examining the clonal space occupied by specific clones

Description

This function calculates the relative clonal space occupied by the clones. The grouping of these clones is based on the parameter **clonalSplit**, at default, **clonalSplit** will group the clones into bins of 1:10, 11:100, 101:1001, etc. To adjust the clones selected, change the numbers in the variable split. If a matrix output for the data is preferred, set **exportTable** = TRUE.
Usage

clonalProportion(
  input.data,
  clonalSplit = c(10, 100, 1000, 10000, 30000, 1e+05),
  cloneCall = "strict",
  chain = "both",
  group.by = NULL,
  exportTable = FALSE,
  palette = "inferno"
)

Arguments

input.data The product of combineTCR, combineBCR, or combineExpression.
clonalSplit The cut points for the specific clones.
cloneCall How to call the clone - VDJC gene (gene), CDR3 nucleotide (nt), CDR3 amino
acid (aa), VDJC gene + CDR3 nucleotide (strict) or a custom variable in the
data.
chain indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG",
"IGH", "IGL".
group.by The variable to use for grouping.
exportTable Exports a table of the data into the global. environment in addition to the visu-
alization.
palette Colors to use in visualization - input any hcl.pals.

Value

ggplot of the space occupied by the specific rank of clones

Examples

#Making combined contig data
combined <- combineTCR(contig_list,
clonalProportion(combined, cloneCall = "gene")

---

clonalQuant Quantify the unique clones by group or sample

Description

This function quantifies unique clones. The unique clones can be either reported as a raw output or
scaled to the total number of clones recovered using the scale parameter.
Usage

clonalQuant(
  input.data,
  cloneCall = "strict",
  chain = "both",
  scale = FALSE,
  group.by = NULL,
  exportTable = FALSE,
  palette = "inferno"
)

Arguments

input.data The product of combineTCR, combineBCR, or combineExpression.
cloneCall How to call the clone - VDJC gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), VDJC gene + CDR3 nucleotide (strict) or a custom variable in the data.
chain indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL"
scale Converts the graphs into percentage of unique clones.
group.by The column header used for grouping.
exportTable Returns the data frame used for forming the graph.
palette Colors to use in visualization - input any hcl.pals.

Value

ggplot of the total or relative unique clones

Examples

#Making combined contig data
combined <- combineTCR(contig_list,
clonalQuant(combined, cloneCall="strict", scale = TRUE)

clonalRarefaction Calculate rarefaction based on the abundance of clones

Description

This functions uses the Hill numbers of order q: species richness (q = 0), Shannon diversity (q = 1), the exponential of Shannon entropy and Simpson diversity (q = 2, the inverse of Simpson concentration) to compute diversity estimates for rarefaction and extrapolation. The function relies on the iNEXT R package. Please read and cite the manuscript if using this function. The input into the iNEXT calculation is abundance, incidence-based calculations are not supported.
Usage

clonalRarefaction(
  input.data,
  cloneCall = "strict",
  chain = "both",
  group.by = NULL,
  plot.type = 1,
  hill.numbers = 0,
  n.boots = 20,
  exportTable = FALSE,
  palette = "inferno"
)

Arguments

input.data The product of `combineTCR`, `combineBCR`, or `combineExpression`.
cloneCall How to call the clone - VDJC gene (`gene`), CDR3 nucleotide (`nt`), CDR3 amino acid (`aa`), VDJC gene + CDR3 nucleotide (`strict`) or a custom variable in the data.
chain indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".
group.by The variable to use for grouping.
plot.type sample-size-based rarefaction/extrapolation curve (type = 1); sample completeness curve (type = 2); coverage-based rarefaction/extrapolation curve (type = 3).
hill.numbers The Hill numbers to be plotted out (0 - species richness, 1 - Shannon, 2 - Simpson)
n.boots The number of bootstraps to downsample in order to get mean diversity.
exportTable Exports a table of the data into the global environment in addition to the visualization.
palette Colors to use in visualization - input any hcl.pals.

Examples

#Making combined contig data
clonalRarefaction(combined[c(1,2)], cloneCall = "gene", n.boots = 3)
clonalScatter

Scatter plot comparing the clonal expansion of two samples

Description

This function produces a scatter plot directly comparing the specific clones between two samples. The clones will be categorized by counts into singlets or expanded, either exclusive or shared between the selected samples.

Usage

clonalScatter(
  input.data,            
  cloneCall = "strict",  
  x.axis = NULL,         
  y.axis = NULL,         
  chain = "both",        
  dot.size = "total",    
  group.by = NULL,       
  graph = "proportion",  
  exportTable = FALSE,   
  palette = "inferno"
)

Arguments

input.data  The product of `combineTCR`, `combineBCR`, or `combineExpression`.
cloneCall   How to call the clone - VDJC gene (`gene`), CDR3 nucleotide (`nt`), CDR3 amino acid (`aa`), VDJC gene + CDR3 nucleotide (`strict`) or a custom variable in the data.
x.axis      name of the list element to appear on the x.axis.
y.axis      name of the list element to appear on the y.axis.
chain       indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".
dot.size    either total or the name of the list element to use for size of dots.
group.by    The variable to use for grouping.
graph       graph either the clonal "proportion" or "count".
exportTable Returns the data frame used for forming the graph.
palette     Colors to use in visualization - input any hcl.pals.

Value

ggplot of the relative clone numbers between two sequencing runs or groups
Examples

# Making combined contig data
combined <- combineTCR(contig_list,

clonalScatter(combined,
x.axis = "P17B",
y.axis = "P17L",
graph = "proportion")

---

clonalSizeDistribution

**Hierarchical clustering of clones using Gamma-GPD spliced threshold model**

Description

This function produces a hierarchical clustering of clones by sample using discrete gamma-GPD spliced threshold model. If using this model please read and cite powerTCR (more info available at PMID: 30485278).

Usage

clonalSizeDistribution(
  input.data, 
  cloneCall = "strict", 
  chain = "both", 
  method = "ward.D2", 
  threshold = 1, 
  group.by = NULL, 
  exportTable = FALSE, 
  palette = "inferno"
)

Arguments

- **input.data** The product of `combineTCR`, `combineBCR`, or `combineExpression`.
- **cloneCall** How to call the clone - VDJC gene (**gene**), CDR3 nucleotide (**nt**), CDR3 amino acid (**aa**), VDJC gene + CDR3 nucleotide (**strict**) or a custom variable in the data.
- **chain** indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".
- **method** The clustering parameter for the dendrogram.
- **threshold** Numerical vector containing the thresholds the grid search was performed over.
- **group.by** The variable to use for grouping.
- **exportTable** Returns the data frame used for forming the graph.
- **palette** Colors to use in visualization - input any hcl.pals.
Details

The probability density function (pdf) for the **Generalized Pareto Distribution (GPD)** is given by:

\[
f(x|\mu, \sigma, \xi) = \frac{1}{\sigma} \left( 1 + \xi \left( \frac{x - \mu}{\sigma} \right) \right)^{-\left(1 + \xi\right)}
\]

Where:

- \(\mu\) is a location parameter
- \(\sigma > 0\) is a scale parameter
- \(\xi\) is a shape parameter
- \(x \geq \mu\) if \(\xi \geq 0\) and \(\mu \leq x \leq \mu - \sigma/\xi\) if \(\xi < 0\)

The probability density function (pdf) for the **Gamma Distribution** is given by:

\[
f(x|\alpha, \beta) = \frac{x^{\alpha-1}e^{-x/\beta}}{\beta^\alpha \Gamma(\alpha)}
\]

Where:

- \(\alpha > 0\) is the shape parameter
- \(\beta > 0\) is the scale parameter
- \(x \geq 0\)
- \(\Gamma(\alpha)\) is the gamma function of \(\alpha\)

Value

ggplot dendrogram of the clone size distribution

Author(s)

Hillary Koch

Examples

#Making combined contig data
clonalSizeDistribution(combined, cloneCall = "strict", method="ward.D2")
combineBCR

Combining the list of B cell receptor contigs into clones

Description

This function consolidates a list of BCR sequencing results to the level of the individual cell barcodes. Using the samples and ID parameters, the function will add the strings as prefixes to prevent issues with repeated barcodes. The resulting new barcodes will need to match the Seurat or SCE object in order to use, combineExpression. Unlike combineTCR, combineBCR produces a column CTstrict of an index of nucleotide sequence and the corresponding V gene. This index automatically calculates the Levenshtein distance between sequences with the same V gene and will index sequences using a normalized Levenshtein distance with the same ID. After which, clone clusters are called using the components function. Clones that are clustered across multiple sequences will then be labeled with "Cluster" in the CTstrict header.

Usage

combineBCR(
  input.data,
  samples = NULL,
  ID = NULL,
  call.related.clones = TRUE,
  threshold = 0.85,
  removeNA = FALSE,
  removeMulti = FALSE,
  filterMulti = TRUE
)

Arguments

input.data List of filtered contig annotations or outputs from loadContigs.
samples The labels of samples
ID The additional sample labeling (optional).
call.related.clones Use the nucleotide sequence and V gene to call related clones. Default is set to TRUE. FALSE will return a CTstrict or strict clone as V gene + amino acid sequence.
threshold The normalized edit distance to consider. The higher the number the more similarity of sequence will be used for clustering.
removeNA This will remove any chain without values.
removeMulti This will remove barcodes with greater than 2 chains.
filterMulti This option will allow for the selection of the highest-expressing light and heavy chains, if not calling related clones.
Value

List of clones for individual cell barcodes

Examples

```r
#Data derived from the 10x Genomics intratumoral NSCLC B cells
BCR <- read.csv("https://www.borch.dev/uploads/contigs/b_contigs.csv")
combined <- combineBCR(BCR,
samples = "Patient1",
threshold = 0.85)
```

```
combineExpression

Adding clone information to a single-cell object

Description

This function adds the immune receptor information to the Seurat or SCE object to the meta data. By default this function also calculates the frequencies and proportion of the clones by sequencing run (group.by = NULL). To change how the frequencies/proportions are calculated, select a column header for the group.by variable. Importantly, before using `combineExpression` ensure the barcodes of the single-cell object object match the barcodes in the output of the `combineTCR` or `combineBCR`.

Usage

```r
combineExpression(
    input.data, 
    sc.data, 
    cloneCall = "strict", 
    chain = "both", 
    group.by = NULL, 
    proportion = TRUE, 
    filterNA = FALSE, 
    cloneSize = c(Rare = 1e-04, Small = 0.001, Medium = 0.01, Large = 0.1, Hyperexpanded = 1), 
    addLabel = FALSE
)
```

Arguments

- **input.data**
  - The product of `combineTCR`, `combineBCR` or a list of both `combineTCR`, `combineBCR`.
- **sc.data**
  - The Seurat or Single-Cell Experiment (SCE) object to attach
- **cloneCall**
  - How to call the clone - VDJC gene (`gene`), CDR3 nucleotide (`nt`), CDR3 amino acid (`aa`), VDJC gene + CDR3 nucleotide (`strict`) or a custom variable in the data.
combineTCR

Combine the list of T cell receptor contigs into clones

**Description**

This function consolidates a list of TCR sequencing results to the level of the individual cell barcodes. Using the `samples` and `ID` parameters, the function will add the strings as prefixes to prevent issues with repeated barcodes. The resulting new barcodes will need to match the Seurat or SCE object in order to use, `combineExpression`. Several levels of filtering exist - `removeNA`, `removeMulti`, or `filterMulti` are parameters that control how the function deals with barcodes with multiple chains recovered.

**Value**

Single-cell object with clone information added to meta data information

**Examples**

```r
#Getting the combined contigs
combined <- combineTCR(contig_list,
    "P19B","P19L", "P20B", "P20L"))

#Getting a sample of a Seurat object
scRep_example <- get(data("scRep_example"))

#Using combineExpression()
scRep_example <- combineExpression(combined, scRep_example)
```
Usage

```r
combineTCR(
  input.data,
  samples = NULL,
  ID = NULL,
  removeNA = FALSE,
  removeMulti = FALSE,
  filterMulti = FALSE
)
```

Arguments

- **input.data**: List of filtered contig annotations or outputs from `loadContigs`.
- **samples**: The labels of samples (recommended).
- **ID**: The additional sample labeling (optional).
- **removeNA**: This will remove any chain without values.
- **removeMulti**: This will remove barcodes with greater than 2 chains.
- **filterMulti**: This option will allow for the selection of the 2 corresponding chains with the highest expression for a single barcode.

Value

List of clones for individual cell barcodes

Examples

```r
combined <- combineTCR(contig_list,
```

Description

A list of 8 ‘filterd_contig_annotations.csv’ files outputted from 10X Cell Ranger. More information on the data can be found in the following manuscript.
createHTOContigList  Generate a contig list from a multiplexed experiment

Description

This function reprocess and forms a list of contigs for downstream analysis in scRepertoire, `createHTOContigList`. It takes the filtered contig annotation output and the single-cell RNA object to create the list. If using an integrated single-cell object, it is recommended to split the object by sequencing run and remove extra prefixes and suffixes on the barcode before using `createHTOContigList`. Alternatively, the variable `multi.run` can be used to separate a list of contigs by a meta data variable. This may have issues with the repeated barcodes.

Usage

```r
createHTOContigList(contig, sc.data, group.by = NULL, multi.run = NULL)
```

Arguments

- `contig`: The filtered contig annotation file from multiplexed experiment.
- `sc.data`: The Seurat or Single-Cell Experiment object.
- `group.by`: One or more meta data headers to create the contig list based on. If more than one header listed, the function combines them into a single variable.
- `multi.run`: If using integrated single-cell object, the meta data variable that indicates the sequencing run.

Value

Returns a list of contigs as input for `combineBCR` or `combineTCR`.

Examples

```r
## Not run:
filtered.contig <- read.csv(".../Sample/outs/filtered_contig_annotations.csv")

contig.list <- createHTOContigList(contig = filtered.contig,
                                     sc.data = Seurat.Obj,
                                     group.by = "HTO_maxID")

## End(Not run)
```
Exporting clones

Description
This function saves a csv file of clones (genes, amino acid, and nucleotide sequences) by barcodes. The format determines the structure of the csv file - paired will export sequences by barcodes and include multiple chains, airr will export a data frame that is consistent with the AIRR format, and TCRMatch will export a data frame that has the TRB chain with count information.

Usage

```r
exportClones(
  input.data,
  format = "paired",
  group.by = NULL,
  write.file = TRUE,
  dir = NULL,
  file.name = "clones.csv"
)
```

Arguments

- `input.data`: The product of `combineTCR`, `combineBCR`, or `combineExpression`.
- `format`: The format to export the clones - "paired", "airr", or "TCRMatch".
- `group.by`: The variable to use for grouping.
- `write.file`: TRUE, save the file or FALSE, return a data.frame
- `dir`: directory location to save the csv
- `file.name`: the csv file name

Value

CSV file of the paired sequences.

Author(s)
Jonathan Noonan, Nick Borcherding

Examples

```r
## Not run:
#Making combined contig data
combined <- combineTCR(contig_list,
exportClones(combined,
             format = "paired")
```
getCirclize

Generate data frame to be used with circlize R package to visualize clones as a chord diagram.

Description

This function will take the meta data from the product of `combineExpression` and generate a relational data frame to be used for a chord diagram. Each cord will represent the number of clone unique and shared across the multiple `group.by` variable. If using the downstream circlize R package, please read and cite the following manuscript. If looking for more advance ways for circular visualizations, there is a great cookbook for the circlize package.

Usage

```r
getCirclize(sc.data, cloneCall = "strict", group.by = NULL, proportion = FALSE)
```

Arguments

- **sc.data**: The single-cell object after `combineExpression`.
- **cloneCall**: How to call the clone - VDJC gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), VDJC gene + CDR3 nucleotide (strict) or a custom variable in the data.
- **group.by**: The group header for which you would like to analyze the data.
- **proportion**: Calculate the relationship unique clones (proportion = FALSE) or normalized by proportion (proportion = TRUE)

Value

A data frame of shared clones between groups format for `chordDiagram`

Author(s)

Dillon Corvino, Nick Borcherding

Examples

```r
#Getting the combined contigs
cmp <- combineTCR(contig_list,

#Getting a sample of a Seurat object
scRep_example <- get(data("scRep_example"))
scRep_example <- combineExpression(combined,
```
highlightClones

Highlighting specific clones in Seurat

Description

Use a specific clonal sequence to highlight on top of the dimensional reduction in single-cell object.

Usage

highlightClones(
  sc.data,
  cloneCall = c("gene", "nt", "aa", "strict"),
  sequence = NULL
)

Arguments

sc.data The single-cell object to attach after `combineExpression`
cloneCall How to call the clone - VDJC gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), VDJC gene + CDR3 nucleotide (strict) or a custom variable in the data.
sequence The specific sequence or sequence to highlight

Value

Single-cell object object with new meta data column for indicated clones

Examples

#Getting the combined contigs
combined <- combineTCR(contig_list,
                 "P19B","P19L", "P20B", "P20L"))

#Getting a sample of a Seurat object
scRep_example <- get(data("scRep_example"))

#Using combineExpression()
scRep_example <- combineExpression(combined,
                                    scRep_example)
# Using highlightClones()
```r
scRep_example <- highlightClones(scRep_example,
    cloneCall = "aa",
    sequence = c("CVVSDNTGGFKTIF_CASSVRERANTGELFF"))
```

## loadContigs

**Description**

This function generates a contig list and formats the data to allow for function with `combineTCR` or `combineBCR`. If using data derived from filtered outputs of 10X Genomics, there is no need to use this function as the data is already compatible.

**Usage**

```r
loadContigs(input, format = "10X")
```

**Arguments**

- `input`: The directory in which contigs are located or a list with contig elements
- `format`: The format of the single-cell contig, currently supporting: "10X", "AIRR", "BD", "JSON", "MiXCR", "Omniscope", "TRUST4", and "WAT3R"

**Details**

The files that this function parses includes:

- 10X = "filtered_contig_annotations.csv"
- AIRR = "airr_rearrangement.tsv"
- BD = "Contigs_AIRR.tsv"
- Immcantation = "data.tsv"
- JSON = ".json"
- MiXCR = "clones.tsv"
- Omniscope = ".csv"
- TRUST4 = "barcode_report.tsv"
- WAT3R = "barcode_results.csv"

**Value**

List of contigs for compatibility with `combineTCR` or `combineBCR"
Examples

```r
TRUST4 <- read.csv("https://www.borch.dev/uploads/contigs/TRUST4_contigs.csv")
contig.list <- loadContigs(TRUST4, format = "TRUST4")

BD <- read.csv("https://www.borch.dev/uploads/contigs/BD_contigs.csv")
contig.list <- loadContigs(BD, format = "BD")

WAT3R <- read.csv("https://www.borch.dev/uploads/contigs/WAT3R_contigs.csv")
contig.list <- loadContigs(WAT3R, format = "WAT3R")
```

---

### mini_contig_list

**Processed subset of `contig_list`**

**Description**

A list of 8 data frames of T cell contigs outputted from the `filtered_contig_annotation` files, but subsetted to 365 valid T cells which correspond to the same barcodes found in `scRep_example`. The data is originally derived from the following [manuscript](https://www.borch.dev/uploads/contigs/TRUST4_contigs.csv).

**Usage**

```r
data("mini_contig_list")
```

**Format**

An R `list` of `data.frame` objects

**See Also**

`contig_list`

---

### percentAA

**Examining the relative amino acid composition by position**

**Description**

This function the proportion of amino acids along the residues of the CDR3 amino acid sequence.

**Usage**

```r
percentAA(
  input.data,
  chain = "TRB",
  group.by = NULL,
  aa.length = 20,
  exportTable = FALSE,
  palette = "inferno"
)
```
percentGenes

Arguments

input.data  The product of combineTCR, combineBCR, or combineExpression.
chain       "TRA", "TRB", "TRG", "IGH", "IGL".
group.by    The variable to use for grouping.
aa.length   The maximum length of the CDR3 amino acid sequence.
exportTable Returns the data frame used for forming the graph.
palette     Colors to use in visualization - input any hcl.pals.

Value

ggplot of stacked bar graphs of amino acid proportions

Examples

#Making combined contig data
combined <- combineTCR(contig_list,
percentAA(combined,
    chain = "TRB",
    aa.length = 20)

Description

This function the proportion V or J genes used by grouping variables. This function only quantifies
single gene loci for indicated chain. For examining VJ pairing, please see code percentVJ

Usage

percentGenes(
    input.data,
    chain = "TRB",
    gene = "Vgene",
    group.by = NULL,
    exportTable = FALSE,
    palette = "inferno"
)
percentKmer

Arguments

input.data  The product of `combineTCR`, `combineBCR`, or `combineExpression`.
chain       "TRA", "TRB", "TRG", "IGH", "IGL".
gene        "V", "D" or "J".
group.by    The variable to use for grouping.
exportTable Returns the data frame used for forming the graph.
palette     Colors to use in visualization - input any hcl.pals.

Value

`ggplot` of percentage of indicated genes as a heatmap

Examples

# Making combined contig data
combined <- combineTCR(contig_list,
                       "P19B","P19L", "P20B", "P20L"))
percentGenes(combined,
             chain = "TRB",
             gene = "Vgene")

Description

This function the of kmer for nucleotide (nt) or amino acid (aa) sequences. Select the length of the kmer to quantify using the `motif.length` parameter.

Usage

percentKmer(
  input.data,  
  chain = "TRB",
  cloneCall = "aa",
  group.by = NULL,
  motif.length = 3,
  top.motifs = 30,
  exportTable = FALSE,
  palette = "inferno"
  )
Arguments

input.data  The product of combineTCR, combineBCR, or combineExpression.
chain      "TRA", "TRB", "TRG", "IGH", "IGL".
cloneCall  How to call the clone - CDR3 nucleotide (nt) or CDR3 amino acid (aa).
group.by   The variable to use for grouping.
motif.length The length of the kmer to analyze.
top.motifs Return the n most variable motifs as a function of median absolute deviation.
exportTable Returns the data frame used for forming the graph.
palette    Colors to use in visualization - input any hcl.pals.

Value

ggplot of percentage of kmers as a heatmap

Examples

#Making combined contig data
combined <- combineTCR(contig_list,
percentKmer(combined,
            chain = "TRB",
            motif.length = 3)
**Arguments**

- **input.data**  The product of `combineTCR, combineBCR, or combineExpression`.
- **chain**  "TRA", "TRB", "TRG", "IGH", "IGL"
- **group.by**  The variable to use for grouping.
- **exportTable**  Returns the data frame used for forming the graph
- **palette**  Colors to use in visualization - input any `hcl.pals`.

**Value**

ggplot of percentage of V and J gene pairings as a heatmap

**Examples**

```r
#Making combined contig data
combined <- combineTCR(contig_list, 
"P19B","P19L", "P20B", "P20L"))
percentVJ(combined, chain = "TRB")
```

---

**positionalEntropy**  *Examining the diversity of amino acids by position*

**Description**

This function the diversity amino acids along the residues of the CDR3 amino acid sequence. Please see `clonalDiversity` for more information on the underlying methods for diversity/entropy calculations. Positions without variance will have a value reported as 0 for the purposes of comparison.

**Usage**

```r
positionalEntropy(
  input.data, 
  chain = "TRB", 
  group.by = NULL, 
  aa.length = 20, 
  method = "norm.entropy", 
  exportTable = FALSE, 
  palette = "inferno"
)
```
Arguments

input.data  The product of `combineTCR`, `combineBCR`, or `combineExpression`.
chain       "TRA", "TRB", "TRG", "IGH", "IGL".
group.by    The variable to use for grouping.
aa.length   The maximum length of the CDR3 amino acid sequence.
method      The method to calculate the entropy/diversity - "shannon", "inv.simpson", "norm.entropy".
exportTable Returns the data frame used for forming the graph.
palette     Colors to use in visualization - input any `hcl.pals`.

Value

ggplot of line graph of diversity by position

Examples

#Making combined contig data
combined <- combineTCR(contig_list,

positionEntropy(combined,
                 chain = "TRB",
                 aa.length = 20)

positionalEntropy

Examining the mean property of amino acids by position

Description

This function calculates the mean selected property for amino acids along the residues of the CDR3 amino acid sequence. The ribbon surrounding the individual line represents the 95 confidence interval.

Usage

positionalProperty(
  input.data,
  chain = "TRB",
  group.by = NULL,
  aa.length = 20,
  method = "Atchley",
  exportTable = FALSE,
  palette = "inferno"
)
### Arguments

**input.data**
- The product of `combineTCR`, `combineBCR`, or `combineExpression`.

**chain**
- "TRA", "TRB", "TRG", "IGH", "IGL".

**group.by**
- The variable to use for grouping.

**aa.length**
- The maximum length of the CDR3 amino acid sequence.

**method**
- The method to calculate the property - "Atchley", "Kidera", "stScales", "tScales", or "VHSE".

**exportTable**
- Returns the data frame used for forming the graph.

**palette**
- Colors to use in visualization - input any `hcl.pals`.

### Details

More information for the individual methods can be found at the following citations:

- **Atchley**: [citation](#)
- **Kidera**: [citation](#)
- **stScales**: [citation](#)
- **tScales**: [citation](#)
- **VHSE**: [citation](#)

### Value

`ggplot` of line graph of diversity by position

### Examples

```r
#Making combined contig data
combined <- combineTCR(contig_list, 

positionalProperty(combined, 
chain = "TRB", 
method = "Atchley", 
aa.length = 20)
```

---

**scRep_example**

* A Seurat object of 500 single T cells.

---

**Description**

The object is compatible with 'contig_list' and the TCR sequencing data can be added with 'combineExpression'. The data is from 4 patients with acute respiratory distress, with samples taken from both the lung and peripheral blood. More information on the data can be found in the following manuscript.
**StartracDiversity**  
*Startrac-based diversity indices for single-cell RNA-seq*

**Description**

This function utilizes the Startrac approach derived from PMID: 30479382. Required to run the function, the "type" variable needs to include the difference in where the cells were derived. The output of this function will produce 3 indices: **expa** (clonal expansion), **migra** (cross-tissue migration), and **trans** (state transition). In order to understand the underlying analyses of the outputs please read and cite the linked manuscript.

**Usage**

```r
StartracDiversity(
  sc.data,
  cloneCall = "strict",
  chain = "both",
  type = NULL,
  group.by = NULL,
  exportTable = FALSE,
  palette = "inferno"
)
```

**Arguments**

- `sc.data` The single-cell object after `combineExpression`. For SCE objects, the cluster variable must be in the meta data under "cluster".
- `cloneCall` How to call the clone - VDJC gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), VDJC gene + CDR3 nucleotide (strict) or a custom variable in the data.
- `chain` indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL".
- `type` The variable in the meta data that provides tissue type.
- `group.by` The variable in the meta data to group by, often samples.
- `exportTable` Returns the data frame used for forming the graph.
- `palette` Colors to use in visualization - input any hcl.pals.

**Value**

`ggplot` object of Startrac diversity metrics

**Author(s)**

Liangtao Zheng
subsetClones

Subset the product of `combineTCR()` or `combineBCR()`

Description

This function allows for the subsetting of the product of `combineTCR` or `combineBCR` by the name of the individual list element.

Usage

```
subsetClones(input.data, name, variables = NULL)
```

Arguments

- `input.data`: The product of `combineTCR` or `combineBCR`.
- `name`: The column header/name to use for subsetting.
- `variables`: The values to subset by, must be in the names(input.data).

Value

List of contigs that have been filtered for the name parameter.

Examples

```
combined <- combineTCR(contig_list, 

subset <- subsetClones(combined, name = "sample", variables = c("P17B"))
```
**vizGenes**

*Visualizing the distribution of gene usage*

**Description**

This function will allow for the visualizing the distribution of the any VDJ and C gene of the TCR or BCR using heatmap or bar chart. This function requires assumes two chains were used in defining clone, if not, it will default to the only chain present regardless of the chain parameter.

**Usage**

```r
vizGenes(
  input.data,
  x.axis = "TRBV",
  y.axis = NULL,
  group.by = NULL,
  plot = "heatmap",
  order = "gene",
  scale = TRUE,
  exportTable = FALSE,
  palette = "inferno"
)
```

**Arguments**

- `input.data`: The product of `combineTCR`, `combineBCR`, or `combineExpression`.
- `x.axis`: Gene segments to separate the x-axis, such as "TRAV", "TRBD", "IGKJ".
- `y.axis`: Variable to separate the y-axis, can be both categorical or other gene gene segments, such as "TRAV", "TRBD", "IGKJ".
- `group.by`: Variable in which to group the diversity calculation.
- `plot`: The type of plot to return - heatmap or barplot.
- `order`: Categorical variable to organize the x-axis, either "gene" or "variance"
- `scale`: Converts the individual count of genes to proportion using the total respective repertoire size
- `exportTable`: Returns the data frame used for forming the graph.
- `palette`: Colors to use in visualization - input any hcl.pals.

**Value**

`ggplot` bar diagram or heatmap of gene usage
Examples

```r
# Making combined contig data
combined <- combineTCR(contig_list,

vizGenes(combined,
         x.axis = "TRBV",
         y.axis = NULL,
         plot = "heatmap")
```
Index

* **Data**
  
  contig_list, 30
  mini_contig_list, 36
  scRep_example, 42

* **Loading_and_Processing_Contigs**
  
  addVariable, 3
  combineBCR, 27
  combineTCR, 29
  createHTOContigList, 31
  exportClones, 32
  loadContigs, 35
  subsetClones, 44

* **SC_Functions**
  
  alluvialClones, 4
  clonalBias, 7
  clonalNetwork, 15
  clonalOccupy, 16
  clonalOverlay, 19
  combineExpression, 28
  getCirclize, 33
  highlightClones, 34
  StartracDiversity, 43

* **Summarize_Repertoire**
  
  percentAA, 36
  percentGenes, 37
  percentKmer, 38
  percentVJ, 39
  positionalEntropy, 40
  positionalProperty, 41

* **Visualizing_Clones**
  
  clonalAbundance, 6
  clonalCluster, 8
  clonalCompare, 9
  clonalDiversity, 11
  clonalHomeostasis, 13
  clonalLength, 14
  clonalOverlap, 18
  clonalProportion, 20
  clonalQuant, 21
  clonalRarefaction, 22
  clonalScatter, 24
  clonalSizeDistribution, 25
  vizGenes, 45

* **datasets**
  
  mini_contig_list, 36

* **internal**
  
  scRepertoire-package, 3

addVariable, 3
alluvialClones, 4
chordDiagram, 33
clonalAbundance, 6
clonalBias, 7, 7
clonalCluster, 8
clonalCompare, 9
clonalDiversity, 11, 40
clonalHomeostasis, 13
clonalLength, 14
clonalNetwork, 15
clonalOccupy, 16
clonalOverlap, 18
clonalOverlay, 19
clonalProportion, 20
clonalQuant, 21
clonalRarefaction, 22
clonalScatter, 24
clonalSizeDistribution, 25
combineBCR, 3, 4, 6, 8–11, 13, 14, 18, 21–25, 27, 28, 31, 32, 35, 37–42, 44, 45
combineExpression, 4–11, 13–18, 20–25, 27, 28, 29, 32–34, 37–43, 45
combineTCR, 3, 4, 6, 8–11, 13, 14, 18, 21–25, 27, 28, 29, 31, 32, 35, 37–42, 44, 45
components, 27
contig_list, 30, 36
createHTOContigList, 31, 31
exportClones, 32
getIndex

getCirclize, 33

hcl.pals, 5–7, 10, 11, 13, 14, 16–18, 21–25, 37–43, 45
highlightClones, 34

iNEXT, 22

loadContigs, 27, 30, 35

mini_contig_list, 36

percentAA, 36
percentGenes, 37
percentKmer, 38
percentVJ, 37, 39
positionalEntropy, 40
positionalProperty, 41

scRep_example, 42
scRepertoire(scRepertoire-package), 3
scRepertoire-package, 3
StartracDiversity, 43
subsetClones, 44

vizGenes, 45