

Package ‘scRepertoire’

March 28, 2021

Title A toolkit for single-cell immune receptor profiling

Version 1.0.0

Description scRepertoire was built to process data derived from the 10x Genomics Chromium Immune Profiling for both T-cell receptor (TCR) and immunoglobulin (Ig) enrichment workflows and subsequently interacts with the popular Seurat and SingleCellExperiment R packages.

License Apache License 2.0

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

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

Depends dplyr, ggalluvial, ggplot2, reshape2, R (>= 4.0)

Imports Biostrings, RColorBrewer, colorRamps, ggdendro, ggfittext, stringr, vegan, powerTCR, SummarizedExperiment, SingleCellExperiment, Seurat

Suggests knitr, rmarkdown, BiocStyle, scater, circlize, scales

VignetteBuilder knitr

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

git_branch RELEASE_3_12

git_last_commit 320c88b

git_last_commit_date 2020-11-23

Date/Publication 2021-03-27

Author Nick Borcharding [aut, cre]

Maintainer Nick Borcharding <ncborch@gmail.com>

R topics documented:

abundanceContig	2
addVariable	3
alluvialClonotypes	4
clonalDiversity	5
clonalHomeostasis	6
clonalOverlap	7
clonalProportion	8

clonesizeDistribution	9
combineBCR	10
combineExpression	11
combineTCR	12
compareClonotypes	13
contig_list	14
expression2List	14
getCirclize	15
highlightClonotypes	16
lengthContig	17
occupiedscRepertoire	18
quantContig	19
screp_example	19
stripBarcode	20
subsetContig	20
vizVgenes	21

Index	22
--------------	-----------

abundanceContig	<i>Demonstrate the relative abundance of clonotypes by group or sample.</i>
-----------------	---

Description

This function takes the output of `combineTCR()`, `combineBCR()`, or `expression2List()` and displays the number of clonotypes at specific frequencies by sample or group. Visualization can either be a line graph using calculated numbers or if `scale = TRUE`, the output will be a density plot. Multiple sequencing runs can be group together using the `group` parameter. If a matrix output for the data is preferred, set `exportTable = TRUE`.

Usage

```
abundanceContig(
  df,
  cloneCall = "gene+nt",
  scale = FALSE,
  group = NULL,
  exportTable = FALSE
)
```

Arguments

<code>df</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>expression2List()</code> .
<code>cloneCall</code>	How to call the clonotype - CDR3 gene (<code>gene</code>), CDR3 nucleotide (<code>nt</code>), CDR3 amino acid (<code>aa</code>), or C DR3 gene+nucleotide (<code>gene+nt</code>).
<code>scale</code>	Converts the graphs into denisty plots in order to show relative distributions.
<code>group</code>	The column header for which you would like to analyze the data.
<code>exportTable</code>	Exports a table of the data into the global environment in addition to the visualization.

Value

ggplot of the total or relative abundance of clonotypes across quanta

Examples

```
#Making combined contig data
x <- contig_list
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),
rep(c("P", "T"), 3), cells = "T-AB")
abundanceContig(combined, cloneCall = "gene", scale = FALSE)
```

addVariable

Adding variables after the combination of contigs.

Description

This function adds variables to the product of combineTCR() combineBCR() or expression2List() to be used in later visualizations. For each element, the function will add a column (labeled by name) with the variable. The length of the variable parameter needs to match the length of the combined object.

Usage

```
addVariable(df, name = NULL, variables = NULL)
```

Arguments

df	The product of combineTCR() combineBCR() or expression2List().
name	The column header to add.
variables	The exact values to add to each element of the list.

Value

list of contigs with a new column (name).

Examples

```
x <- contig_list
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),
rep(c("P", "T"), 3), cells = "T-AB")
combined <- addVariable(combined, name = "batch", variables = c(1,1,1,1,2,2))
```

alluvialClonotypes *Exploring interaction of clonotypes by seurat or SCE dynamics*

Description

View the proportional contribution of clonotypes 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 axes you wish to view or will add an additional stratum/column to the end of the graph.

Usage

```
alluvialClonotypes(
  sc,
  cloneCall = c("gene", "nt", "aa", "gene+nt"),
  y.axes = NULL,
  color = NULL,
  alpha = NULL,
  facet = NULL
)
```

Arguments

sc	The seurat or SCE object to visualize after combineExpression(). For SCE objects, the cluster variable must be in the meta data under "cluster".
cloneCall	How to call the clonotype - CDR3 gene (gene), CDR3 nucleotide (nt) or CDR3 amino acid (aa), or CDR3 gene+nucleotide (gene+nt).
y.axes	The columns that will separate the proportional visualizations.
color	The column header or clonotype(s) to be highlighted.
alpha	The column header to have gradient opacity.
facet	The column label to separate.

Value

Alluvial ggplot comparing clonotype distribution across selected parameters.

Examples

```
#Getting the combined contigs
combined <- combineTCR(contig_list, rep(c("PX", "PY", "PZ"), each=2),
  rep(c("P", "T"), 3), cells ="T-AB")

#Getting a sample of a Seurat object
screp_example <- get(data("screp_example"))
sce <- suppressMessages(Seurat::UpdateSeuratObject(screp_example))
sce <- Seurat::as.SingleCellExperiment(sce)

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

#Using alluvialClonotypes()
```

```
alluvialClonotypes(sce, cloneCall = "gene",  
y.axes = c("Patient", "cluster"), color = "cluster")
```

clonalDiversity

Examine the clonal diversity of samples

Description

This function calculates traditional measures of diversity - Shannon, inverse Simpson, Chao1 index, and abundance-based coverage estimators (ACE) by sample or group. 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(  
  df,  
  cloneCall = "gene+nt",  
  group = "samples",  
  exportTable = FALSE  
)
```

Arguments

<code>df</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>expression2List()</code> .
<code>cloneCall</code>	How to call the clonotype - CDR3 gene (gene), CDR3 nucleotide (nt) or CDR3 amino acid (aa), or CDR3 gene+nucleotide (gene+nt).
<code>group</code>	The column header for which you would like to analyze the data.
<code>exportTable</code>	Exports a table of the data into the global environment in addition to the visualization

Value

ggplot of the diversity of clonotype sequences across list

Examples

```
#Making combined contig data  
x <- contig_list  
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),  
rep(c("P", "T"), 3), cells = "T-AB")  
clonalDiversity(combined, cloneCall = "gene")
```

clonalHomeostasis *Examining the clonal homeostasis*

Description

This function calculates the space occupied by clonotype proportions. The grouping of these clonotypes is based on the parameter `cloneTypes`, at default, `cloneTypes` will group the clonotypes 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 `cloneTypes` paramter. If a matrix output for the data is preferred, set `exportTable = TRUE`.

Usage

```
clonalHomeostasis(
  df,
  cloneTypes = c(Rare = 1e-04, Small = 0.001, Medium = 0.01, Large = 0.1, Hyperexpanded
    = 1),
  cloneCall = "gene+nt",
  exportTable = FALSE
)
```

Arguments

<code>df</code>	The product of <code>CombineContig()</code> or <code>expression2List()</code>
<code>cloneTypes</code>	The cutpoints of the proportions.
<code>cloneCall</code>	How to call the clonotype - CDR3 gene (gene), CDR3 nucleotide (nt) or CDR3 amino acid (aa), or CDR3 gene+nucleotide (gene+nt).
<code>exportTable</code>	Exports a table of the data into the global environment in addition to the visualization

Value

ggplot of the space occupied by the specific propotion of clonotypes

Examples

```
#Making combined contig data
x <- contig_list
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),
  rep(c("P", "T"), 3), cells="T-AB")
clonalHomeostasis(combined, cloneCall = "gene")
```

Description

This functions allows for the caclulation and visualizations of the overlap coefficient or morisita index for clonotypes using the product of combineTCR(), combineBCR() or expression2list(). The overlap coefficient is calculated using the intersection of clonotypes divided by the length of the smallest componenet. Morisita index is estimating the dispersion of a population, more information can be found [here](<https://en.wikipedia.org/wiki/Morisita>) If a matrix output for the data is preferred, set exportTable = TRUE.

Usage

```
clonalOverlap(  
  df,  
  cloneCall = c("gene", "nt", "aa", "gene+nt"),  
  method = c("overlap", "morisita"),  
  exportTable = FALSE  
)
```

Arguments

df	The product of combineTCR(), combineBCR(), or expression2List().
cloneCall	How to call the clonotype - CDR3 gene (gene), CDR3 nucleotide (nt) or CDR3 amino acid (aa), or CDR3 gene+nucleotide (gene+nt).
method	The method to calculate the overlap, either the overlap coefficient or morisita index.
exportTable	Exports a table of the data into the global environment in addition to the visualization

Value

ggplot of the clonotypic overlap between elements of a list

Examples

```
#Making combined contig data  
x <- contig_list  
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),  
  rep(c("P", "T"), 3), cells = "T-AB")  
  
clonalOverlap(combined, cloneCall = "gene", method = "overlap")
```

clonalProportion	<i>Examining the clonal space occupied by specific clonotypes</i>
------------------	---

Description

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

Usage

```
clonalProportion(  
  df,  
  split = c(10, 100, 1000, 10000, 30000, 1e+05),  
  cloneCall = "gene+nt",  
  exportTable = FALSE  
)
```

Arguments

<code>df</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>expression2List()</code> .
<code>split</code>	The cutpoints for the specific clonotypes.
<code>cloneCall</code>	How to call the clonotype - CDR3 gene (gene), CDR3 nucleotide (nt) or CDR3 amino acid (aa), or CDR3 gene+nucleotide (gene+nt).
<code>exportTable</code>	Exports a table of the data into the global environment in addition to the visualization

Value

ggplot of the space occupied by the specific rank of clonotypes

Examples

```
#Making combined contig data  
x <- contig_list  
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),  
  rep(c("P", "T"), 3), cells = "T-AB")  
clonalProportion(combined, cloneCall = "gene")
```

clonesizeDistribution *Hierarchical clustering of clonotypes on clonotype size and Jensen-Shannon divergence*

Description

This function produces a hierarchical clustering of clonotypes by sample using the Jensen-Shannon distance and discrete gamma-GPD spliced threshold model in the [powerTCR R package] (<https://bioconductor.org/packages/>). Please read and cite PMID: 30485278 if using the function for analyses. If a matrix output for the data is preferred set `exportTable = TRUE`.

Usage

```
clonesizeDistribution(  
  df,  
  cloneCall = "gene+nt",  
  method = "ward.D2",  
  exportTable = FALSE  
)
```

Arguments

<code>df</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>expression2List()</code> .
<code>cloneCall</code>	How to call the clonotype - CDR3 gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), or CDR3 gene+nucleotide (gene+nt).
<code>method</code>	The clustering parameter for the dendrogram.
<code>exportTable</code>	Returns the data frame used for forming the graph.

Value

ggplot dendrogram of the clone size distribution

Examples

```
#Making combined contig data  
x <- contig_list  
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),  
  rep(c("P", "T"), 3), cells="T-AB")  
clonesizeDistribution(combined, cloneCall = "gene+nt", method="ward.D2")
```

`combineBCR`*Combining the list of B Cell Receptor contigs*

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,

Usage

```
combineBCR(  
  df,  
  samples = NULL,  
  ID = NULL,  
  removeNA = FALSE,  
  removeMulti = FALSE  
)
```

Arguments

<code>df</code>	List of filtered contig annotations from 10x Genomics.
<code>samples</code>	The labels of samples.
<code>ID</code>	The additional sample labeling option.
<code>removeNA</code>	This will remove any chain without values.
<code>removeMulti</code>	This will remove barcodes with greater than 2 chains.

Value

List of clonotypes for individual cell barcodes

See Also

[combineExpression](#). Unlike `combineTCR()`, `combineBCR` produces a column `CTstrict` of an index of nucleotide sequence and the corresponding `v-gene`. This index automatically calculates the Hamming distance between sequences of the same length and will index sequences with > 0.85 normalized Hamming distance with the same `ID`. If nucleotide sequences meet the threshold, `":HD"` will be added to the `CTstrict` column string.

Examples

```
#Data derived from the 10x Genomics intratumoral NSCLC B cells  
BCR <- read.csv("https://ncborcherding.github.io/vignettes/b_contigs.csv",  
stringsAsFactors = FALSE)  
combined <- combineBCR(BCR, samples = "Patient1", ID = "Time1")
```

combineExpression *Adding clonotype information to a seurat or SCE 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 of the clonotypes by sequencing run (groupBy = "none"). To change how the frequencies are calculated, select a column header for the groupBy variable. Importantly, before using combineExpression() ensure the barcodes of the seurat or SCE object match the barcodes in the output of the combinedContig() call. Check changeNames() to change the prefix of the seurat object. If the dominant clonotypes have a greater frequency than 500, adjust the cloneTypes variable.

Usage

```
combineExpression(
  df,
  sc,
  cloneCall = "gene+nt",
  groupBy = "none",
  cloneTypes = c(None = 0, Single = 1, Small = 5, Medium = 20, Large = 100,
    Hyperexpanded = 500),
  filterNA = FALSE
)
```

Arguments

df	The product of CombineTCR() or CombineBCR().
sc	The seurat or SingleCellExperiment (SCE) object to attach
cloneCall	How to call the clonotype - CDR3 gene (gene), CDR3 nucleotide (nt) CDR3 amino acid (aa), or CDR3 gene+nucleotide (gene+nt).
groupBy	The column label in the combined contig object in which clonotype frequency will be calculated.
cloneTypes	The bins for the grouping based on frequency
filterNA	Method to subset seurat object of barcodes without clonotype information

Value

seurat or SingleCellExperiment object with attached clonotype information

Examples

```
#Getting the combined contigs
combined <- combineTCR(contig_list, rep(c("PX", "PY", "PZ"), each=2),
  rep(c("P", "T"), 3), cells = "T-AB")

#Getting a sample of a Seurat object
screp_example <- get(data("screp_example"))
sce <- suppressMessages(Seurat::UpdateSeuratObject(screp_example))
sce <- Seurat::as.SingleCellExperiment(sce)
```

```
#Using combineExpression()
sce <- combineExpression(merged, sce)
```

 combineTCR

Combining the list of T Cell Receptor contigs

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, @seealso [combineExpression](#). Several levels of filtering exist - `remove` or `filterMulti` are parameters that control how the function deals with barcodes with multiple chains recovered.

Usage

```
combineTCR(
  df,
  samples = NULL,
  ID = NULL,
  cells = c("T-AB", "T-GD"),
  removeNA = FALSE,
  removeMulti = FALSE,
  filterMulti = FALSE
)
```

Arguments

<code>df</code>	List of filtered contig annotations from 10x Genomics.
<code>samples</code>	The labels of samples.
<code>ID</code>	The additional sample labeling option.
<code>cells</code>	The type of T cell - T cell-AB or T cell-GD
<code>removeNA</code>	This will remove any chain without values.
<code>removeMulti</code>	This will remove barcodes with greater than 2 chains.
<code>filterMulti</code>	This option will allow for the selection of the 2 corresponding chains with the highest expression for a single barcode.

Value

List of clonotypes for individual cell barcodes

Examples

```
combineTCR(contig_list, rep(c("PX", "PY", "PZ"), each=2),
  rep(c("P", "T"), 3), cells ="T-AB")
```

compareClonotypes *Demonstrate the difference in clonal proportion between clonotypes*

Description

This function produces an alluvial or area graph of the proportion of the indicated clonotypes for all or selected samples. Clonotypes can be selected using the clonotypes parameter with the specific sequence of interest or using the number parameter with the top n clonotypes by proportion to be visualized. If multiple clonotypes have the same proportion and are within the selection by the number parameter, all the clonotypes will be visualized. In this instance, if less clonotypes are desired, reduce the number parameter.

Usage

```
compareClonotypes(
  df,
  cloneCall = "gene+nt",
  samples = NULL,
  clonotypes = NULL,
  numbers = NULL,
  graph = "alluvial",
  exportTable = FALSE
)
```

Arguments

df	The product of combineTCR(), combineBCR(), or expression2List()
cloneCall	How to call the clonotype - CDR3 gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), or CDR3 gene+nucleotide (gene+nt).
samples	The specific samples to isolate for visualization.
clonotypes	The specific sequences of interest.
numbers	The top number clonotype sequences.
graph	The type of graph produced, either "alluvial" or "area".
exportTable	Returns the data frame used for forming the graph.

Value

ggplot of the proportion of total sequencing read of selecting clonotypes

Examples

```
#Making combined contig data
x <- contig_list
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),
  rep(c("P", "T"), 3), cells="T-AB")
compareClonotypes(combined, numbers = 10,
  samples = c("PX_P", "PX_T"), cloneCall="aa")
```

contig_list	<i>A data set of T cell contigs as a list outputed from the filter_contig_annotation files.</i>
-------------	---

Description

A data set of T cell contigs as a list outputed from the filter_contig_annotation files.

expression2List	<i>Allows users to take the meta data in seurat/SCE and place it into a list that will work with all the functions</i>
-----------------	--

Description

Allows users to perform more fundamental measures of clonotype analysis using the meta data from the seurat or SCE object. For Seurat objects the active identity is automatically added as "cluster". Remaining grouping parameters or SCE or Seurat objects must appear in the meta data.

Usage

```
expression2List(sc, group)
```

Arguments

sc	object after combineExpression().
group	The column header to group the new list by

Value

list derived from the meta data of single-cell object with elements divided by the group parameter

Examples

```
#Getting the combined contigs
combined <- combineTCR(contig_list, rep(c("PX", "PY", "PZ"), each=2),
rep(c("P", "T"), 3), cells ="T-AB")

#Getting a sample of a Seurat object
screp_example <- get(data("screp_example"))
sce <- suppressMessages(Seurat::UpdateSeuratObject(screp_example))
sce <- Seurat::as.SingleCellExperiment(sce)

#Using expression2List
newList <- expression2List(sce, group = "seurat_clusters")
```

getCirclize	<i>Generate data frame to be used with circlize R package to visualize clonotypes as a chord diagram.</i>
-------------	---

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. The output is a measure of relative clonotype overlap between groups and does not reflect exact clonotype matches between groups.

Usage

```
getCirclize(sc, cloneCall = "gene+nt", groupBy = NULL, proportion = FALSE)
```

Arguments

<code>sc</code>	object after <code>combineExpression()</code> .
<code>cloneCall</code>	How to call the clonotype - CDR3 nucleotide (nt), CDR3 amino acid (aa).
<code>groupBy</code>	The group header for which you would like to analyze the data.
<code>proportion</code>	Binary will calculate relationship as unique clonotypes (<code>proportion = TRUE</code>) or proportion of unique clonotypes (<code>proportion = FALSE</code>)

Value

data frame of shared clonotypes between groups

Author(s)

Dillon Corvino, Nick Borcharding

Examples

```
#Getting the combined contigs
combined <- combineTCR(contig_list, rep(c("PX", "PY", "PZ"), each=2),
rep(c("P", "T"), 3), cells ="T-AB")

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

#Getting data frame output for Circlize
circles <- getCirclize(screp_example, groupBy = "seurat_clusters")
```

highlightClonotypes *Highlighting specific clonotypes in Seurat*

Description

Use a specific clonotype sequence to highlight on top of the dimensional reduction in seurat object.

Usage

```
highlightClonotypes(  
  sc,  
  cloneCall = c("gene", "nt", "aa", "gene+nt"),  
  sequence = NULL  
)
```

Arguments

sc	The seurat object to attach
cloneCall	How to call the clonotype - CDR3 gene (gene), CDR3 nucleotide (nt), CDR3 amino acid (aa), or CDR3 gene+nucleotide (gene+nt).
sequence	The specific sequence or sequence to highlight

Value

DimPlot with highlighted clonotypes

Examples

```
## Getting the combined contigs  
combined <- combineTCR(contig_list, rep(c("PX", "PY", "PZ"), each=2),  
  rep(c("P", "T"), 3), cells = "T-AB")  
  
## Getting a sample of a Seurat object  
screp_example <- get(data("screp_example"))  
  
## Using combineExpression()  
screp_example <- combineExpression(combined, screp_example )  
  
## Using highlightClonotype()  
screp_example <- highlightClonotypes(screp_example, cloneCall= "aa",  
  sequence = c("CAVNGGSQGNLIF_CSAEREDTDTQYF"))
```

`lengthContig`*Demonstrate the distribution of lengths filtered contigs.*

Description

This function takes the output of `combineTCR()`, `combineBCR()`, or `expression2List()` and 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 clonotype (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` parameter. If a matrix output for the data is preferred, set `exportTable = TRUE`.

Usage

```
lengthContig(  
  df,  
  cloneCall = "aa",  
  group = NULL,  
  scale = FALSE,  
  chains = "combined",  
  exportTable = FALSE  
)
```

Arguments

<code>df</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>expression2List()</code>
<code>cloneCall</code>	How to call the clonotype - CDR3 nucleotide (nt), CDR3 amino acid (aa).
<code>group</code>	The group header for which you would like to analyze the data.
<code>scale</code>	Converts the graphs into density plots in order to show relative distributions.
<code>chains</code>	Whether to keep clonotypes "combined" or visualize by chain.
<code>exportTable</code>	Returns the data frame used for forming the graph.

Value

ggplot of the discrete or relative length distributions of clonotype sequences

Examples

```
#Making combined contig data  
x <- contig_list  
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),  
  rep(c("P", "T"), 3), cells = "T-AB")  
lengthContig(combined, cloneCall="aa", chains = "combined")
```

occupiedscRepertoire	<i>Visualize the number of single cells with clonotype frequencies by cluster</i>
----------------------	---

Description

View the count of clonotypes frequency group in seurat or SCE object meta data after combineExpression(). The visualization will take the new meta data variable "cloneType" and plot the number of cells with each designation using a secondary variable, like cluster. Credit to the idea goes to Dr. Carmonia and his work with [ProjectTIL](<https://github.com/carmonalab/ProjecTILs>).

Usage

```
occupiedscRepertoire(sc, x.axis = "cluster", exportTable = FALSE)
```

Arguments

sc	The seurat or SCE object to visualize after combineExpression(). For SCE objects, the cluster variable must be in the meta data under "cluster".
x.axis	The variable in the meta data to graph along the x.axis
exportTable	Exports a table of the data into the global environment in addition to the visualization

Value

Stacked bar plot of counts of cells by clonotype frequency group

Examples

```
#Getting the combined contigs
combined <- combineTCR(contig_list, rep(c("PX", "PY", "PZ"), each=2),
rep(c("P", "T"), 3), cells = "T-AB")

#Getting a sample of a Seurat object
screp_example <- get(data("screp_example"))
sce <- suppressMessages(Seurat::UpdateSeuratObject(screp_example))
sce <- Seurat::as.SingleCellExperiment(sce)

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

#Using occupiedscRepertoire()
occupiedscRepertoire(sce, x.axis = "cluster")
table <- occupiedscRepertoire(sce, x.axis = "cluster", exportTable = TRUE)
```

quantContig	<i>Quantify the unique clonotypes in the filtered contigs.</i>
-------------	--

Description

This function takes the output from `combineTCR()`, `combineBCR()`, or `expression2List()` and quantifies unique clonotypes. The unique clonotypes can be either reported as a raw output or scaled to the total number of clonotypes recovered using the `scale` parameter. Multiple sequencing runs can be group together using the `group` parameter. If a matrix output for the data is preferred, set `exportTable = TRUE`.

Usage

```
quantContig(
  df,
  cloneCall = "gene+nt",
  scale = FALSE,
  group = NULL,
  exportTable = FALSE
)
```

Arguments

<code>df</code>	The product of <code>combineTCR()</code> <code>combineBCR()</code> or <code>expression2List()</code> .
<code>cloneCall</code>	How to call the clonotype - CDR3 gene (<code>gene</code>), CDR3 nucleotide (<code>nt</code>), CDR3 amino acid (<code>aa</code>), or CDR3 gene+nucleotide (<code>gene+nt</code>).
<code>scale</code>	Converts the graphs into percentage of unique clonotypes.
<code>group</code>	The column header used for grouping.
<code>exportTable</code>	Returns the data frame used for forming the graph,

Value

ggplot of the total or relative unique clonotypes

Examples

```
#Making combined contig data
x <- contig_list
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),
  rep(c("P", "T"), 3), cells = "T-AB")
quantContig(combined, cloneCall="gene+nt", scale = TRUE)
```

<code>screp_example</code>	<i>A <code>seurat</code> object of 1000 single T cells derived from 3 clear cell renal carcinoma patients.</i>
----------------------------	--

Description

A `seurat` object of 1000 single T cells derived from 3 clear cell renal carcinoma patients.

stripBarcode	<i>Removing any additional prefixes to the barcodes of filtered contigs.</i>
--------------	--

Description

Removing any additional prefixes to the barcodes of filtered contigs.

Usage

```
stripBarcode(contigs, column = 1, connector = "_", num_connects = 3)
```

Arguments

contigs	The raw loaded filtered_contig_annotation.csv
column	The column in which the barcodes are listed
connector	The type of character in which is attaching the default barcode with any other characters
num_connects	The number of strings combined with the connectors

Value

list with the suffixes of the barcodes removed.

Examples

```
stripBarcode(contig_list[[1]], column = 1, connector = "_", num_connects = 1)
```

subsetContig	<i>Subset the product of combineTCR() combineBCR() or expression2List()</i>
--------------	---

Description

This function allows for the subsetting of the product of combineTCR() combineBCR() or expression2List() by the name of the individual list element. In general the names of are samples + _ + ID, allowing for users to subset the product of combineTCR(), combineBCR(), or expression2List() across a string or individual name.

Usage

```
subsetContig(df, name, variables = NULL)
```

Arguments

df	The product of combineTCR(), combineBCR(), or expression2List().
name	The column header you'd like to use to subset.
variables	The values to subset by, must be in the names(df).

Value

list of contigs that have been filtered for the name parameter

Examples

```
x <- contig_list
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),
rep(c("P", "T"), 3), cells="T-AB")
subset <- subsetContig(combined, name = "sample", variables = c("PX"))
```

vizVgenes

Visualizing the distribution of TCR V gene usage

Description

This function will allow for the visualizing the distribution of the V-genes of the TCR by categorical variables.

Usage

```
vizVgenes(
  df,
  TCR = "TCR1",
  facet.x = "sample",
  facet.y = NULL,
  fill = NULL,
  exportTable = FALSE
)
```

Arguments

df	The product of combineTCR(), combineBCR(), or expression2List().
TCR	Which TCR chain to use, TCR1 = TCRA or TCR2 = TCRB
facet.x	Categorical variable which to separate by along x-axis
facet.y	Categorical variable which to separate by along y-axis
fill	Categorical variable which to add color to bar chart
exportTable	Returns the data frame used for forming the graph.

Value

ggplot bar diagram of vgene counts

Examples

```
#Making combined contig data
x <- contig_list
combined <- combineTCR(x, rep(c("PX", "PY", "PZ"), each=2),
rep(c("P", "T"), 3), cells="T-AB")

vizVgenes(combined, TCR = "TCR1", facet.x = "sample")
```

Index

abundanceContig, [2](#)
addVariable, [3](#)
alluvialClonotypes, [4](#)

clonalDiversity, [5](#)
clonalHomeostasis, [6](#)
clonalOverlap, [7](#)
clonalProportion, [8](#)
clonesizeDistribution, [9](#)
combineBCR, [10](#)
combineExpression, [10](#), [11](#), [12](#)
combineTCR, [12](#)
compareClonotypes, [13](#)
contig_list, [14](#)

expression2List, [14](#)

getCirclize, [15](#)

highlightClonotypes, [16](#)

lengthContig, [17](#)

occupiedscRepertoire, [18](#)

quantContig, [19](#)

screp_example, [19](#)
stripBarcode, [20](#)
subsetContig, [20](#)

vizVgenes, [21](#)