

# Package ‘scRepertoire’

December 31, 2024

**Title** A toolkit for single-cell immune receptor profiling

**Version** 2.2.1

**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** 7.3.2

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

**Depends** ggplot2, R (>= 4.0)

**Imports** assertthat, 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, purrr, lifecycle

**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

**URL** <https://www.borch.dev/uploads/scRepertoire/>

**BugReports** <https://github.com/ncborcherding/scRepertoire/issues>

**Roxygen** list(markdown = TRUE)

**git\_url** <https://git.bioconductor.org/packages/scRepertoire>

**git\_branch** RELEASE\_3\_20

**git\_last\_commit** 356de29

**git\_last\_commit\_date** 2024-11-04

**Repository** Bioconductor 3.20

**Date/Publication** 2024-12-30

**Author** Nick Borcherding [aut, cre],  
Qile Yang [aut],  
Ksenia Safina [aut]

**Maintainer** Nick Borcherding <ncborch@gmail.com>

## Contents

scRepertoire-package . . . . .	3
addVariable . . . . .	3
alluvialClones . . . . .	4
clonalAbundance . . . . .	5
clonalBias . . . . .	6
clonalCluster . . . . .	7
clonalCompare . . . . .	8
clonalDiversity . . . . .	10
clonalHomeostasis . . . . .	12
clonalLength . . . . .	13
clonalNetwork . . . . .	14
clonalOccupy . . . . .	15
clonalOverlap . . . . .	16
clonalOverlay . . . . .	18
clonalProportion . . . . .	19
clonalQuant . . . . .	20
clonalRarefaction . . . . .	21
clonalScatter . . . . .	22
clonalSizeDistribution . . . . .	23
combineBCR . . . . .	25
combineExpression . . . . .	26
combineTCR . . . . .	27
contig_list . . . . .	28
createHTOContigList . . . . .	29
exportClones . . . . .	29
getCirclize . . . . .	30
getContigDoublets . . . . .	32
highlightClones . . . . .	32
loadContigs . . . . .	33
mini_contig_list . . . . .	34
percentAA . . . . .	35
percentGenes . . . . .	36
percentKmer . . . . .	37
percentVJ . . . . .	38
positionalEntropy . . . . .	39
positionalProperty . . . . .	40
scRep_example . . . . .	41
StartracDiversity . . . . .	41
subsetClones . . . . .	42
vizGenes . . . . .	43

---

scRepertoire-package    *scRepertoire: A toolkit for single-cell immune receptor profiling*

---

### 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 Single-CellExperiment/Bioconductor R workflows.

### Author(s)

**Maintainer:** Nick Borcharding <ncborch@gmail.com>

Authors:

- Qile Yang <qile.yang@berkeley.edu>
- Ksenia Safina <safina@broadinstitute.org>

### See Also

Useful links:

- <https://www.borch.dev/uploads/scRepertoire/>
- Report bugs at <https://github.com/ncborcharding/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

<code>input.data</code>	The product of <code>combineTCR()</code> or <code>combineBCR()</code> .
<code>variable.name</code>	The new column name/header.
<code>variables</code>	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,
  samples = c("P17B", "P17L", "P18B", "P18L",
             "P19B", "P19L", "P20B", "P20L"))
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 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"
)
```

**Arguments**

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

**Value**

Alluvial ggplot comparing clone distribution.

**Examples**

```
#Getting the combined contigs
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "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 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,
  order.by = NULL,
  exportTable = FALSE,
  palette = "inferno"
)
```

**Arguments**

input.data	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
cloneCall	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order
exportTable	Returns the data frame used for forming the graph to the visualization.
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a> .

### Value

ggplot of the total or relative abundance of clones across quanta

### Examples

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                  "P19B", "P19L", "P20B", "P20L"))
clonalAbundance(combined,
                cloneCall = "gene",
                scale = FALSE)
```

---

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

```
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 <code>combineExpression()</code> .
cloneCall	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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 <a href="#">hcl.pals</a> .

**Value**

ggplot scatter plot with clone bias

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))

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

#Using combineExpresion()
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)
```

**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. If multiple sequences per chain are present, this function only compares the first sequence.

**Usage**

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

**Arguments**

input.data	The product of <code>combineTCR()</code> , <code>combineBCR()</code> or <code>combineExpression()</code> .
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 ( <b>NULL</b> ), clusters will be calculated across samples.
exportGraph	Return an igraph object of connected sequences ( <b>TRUE</b> ) or the amended input with a new cluster-based variable ( <b>FALSE</b> ).

**Value**

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

**Examples**

```
# Getting the combined contigs
combined <- combineTCR(contig_list,
  samples = c("P17B", "P17L", "P18B", "P18L",
             "P19B", "P19L", "P20B", "P20L"))

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

---

clonalCompare	<i>Demonstrate the difference in clonal proportions / counts between clones</i>
---------------	---

---

**Description**

This function produces an alluvial or area graph of the proportion or count composition 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 / counts to be visualized.



**Usage**

```

clonalCompare(
  input.data,
  cloneCall = "strict",
  chain = "both",
  samples = NULL,
  clones = NULL,
  top.clones = NULL,
  highlight.clones = NULL,
  relabel.clones = FALSE,
  group.by = NULL,
  order.by = NULL,
  graph = "alluvial",
  proportion = TRUE,
  exportTable = FALSE,
  palette = "inferno"
)

```

**Arguments**

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

**Value**

ggplot of the proportion of total sequencing read of selecting clones

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                  "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 boot straps (**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,
  order.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 <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
cloneCall	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order

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 <a href="#">hcl.pals</a>
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:

$$Index = \frac{1}{(\sum_{i=1}^S p_i^2)}$$

#### 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:

$$Index = S_{obs} + \frac{n_1(n_1 - 1)}{2 * n_2 + 1}$$

#### 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.
- $S_{abund}$ ,  $S_{rare}$ ,  $C_{ace}$ , and  $F_1$  are parameters derived from the data.

### Value

ggplot of the diversity of clones by group

### Author(s)

Andrew Malone, Nick Borcharding

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))
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**

```
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,
  order.by = NULL,
  exportTable = FALSE,
  palette = "inferno"
)
```

**Arguments**

input.data	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
cloneSize	The cut points of the proportions.
cloneCall	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order
exportTable	Exports a table of the data into the global environment in addition to the visualization.
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a> .

**Value**

ggplot of the space occupied by the specific proportion of clones

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))
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,
  order.by = NULL,
  scale = FALSE,
  exportTable = FALSE,
  palette = "inferno"
)
```

**Arguments**

input.data	The product of <a href="#">combineTCR()</a> , <a href="#">combineBCR()</a> , or <a href="#">combineExpression()</a>
cloneCall	How to call the clone - CDR3 nucleotide ( <b>nt</b> ) or CDR3 amino acid ( <b>aa</b> )
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
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order description
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 <a href="#">hcl.pals</a>

**Value**

ggplot of the discrete or relative length distributions of clone sequences

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                  "P19B", "P19L", "P20B", "P20L"))
clonalLength(combined, cloneCall="aa", chain = "both")
```

---

clonalNetwork	<i>Visualize clonal network along reduced dimensions</i>
---------------	--

---

**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**

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

**Arguments**

sc.data	The single-cell object after <code>combineExpression()</code> .
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., <b>filter.clones</b> = 2000) or n of clones based on the minimum number of all the comparators (e.g., <b>filter.clone</b> = "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 ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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".
exportClones	Exports a table of clones that are shared across multiple identity groups and ordered by the total number of clone copies.
exportTable	Exports a table of the data into the global
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a> .

### Value

ggplot object

### Examples

```
## Not run:
#Getting the combined contigs
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                "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 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](#).

### Usage

```
clonalOccupy(
  sc.data,
  x.axis = "ident",
  label = TRUE,
  facet.by = NULL,
  order.by = NULL,
  proportion = FALSE,
```

```

na.include = FALSE,
exportTable = FALSE,
palette = "inferno"
)

```

### Arguments

sc.data	The single-cell object after <code>combineExpression()</code>
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
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order description
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 <a href="#">hcl.pals</a>

### Value

Stacked bar plot of counts of cells by clone frequency group

### Examples

```

#Getting the combined contigs
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                  "P19B", "P19L", "P20B", "P20L"))

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

#Using combineExpresion()
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 (**overlap**), Morisita's overlap index (**morisita**), Jaccard index (**jaccard**), cosine similarity (**cosine**) or the exact number of clonal overlap (**raw**).



**Usage**

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

**Arguments**

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code>
<code>cloneCall</code>	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) or a custom variable in the data
<code>method</code>	The method to calculate the "overlap", "morisita", "jaccard", "cosine" indices or "raw" for the base numbers
<code>chain</code>	indicate if both or a specific chain should be used - e.g. "both", "TRA", "TRG", "IGH", "IGL"
<code>group.by</code>	The variable to use for grouping
<code>order.by</code>	A vector of specific plotting order or "alphanumeric" to plot groups in order
<code>exportTable</code>	Returns the data frame used for forming the graph
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a>

**Details**

The formulas for the indices are as follows:

**Overlap Coefficient:**

$$overlap = \frac{\sum \min(a, b)}{\min(\sum a, \sum b)}$$

**Raw Count Overlap:**

$$raw = \sum \min(a, b)$$

**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,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))

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

---

clonalOverlay	<i>Visualize distribution of clonal frequency overlaid on dimensional reduction plots</i>
---------------	---

---

**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,
  cut.category = "clonalFrequency",
  cutpoint = 30,
  bins = 25,
  facet.by = NULL
)
```

**Arguments**

sc.data	The single-cell object after <code>combineExpression()</code> .
reduction	The dimensional reduction to visualize.
cut.category	Meta data variable of the single-cell object to use for filtering.
cutpoint	The overlay cut point to include, this corresponds to the cut.category variable in the meta data of the single-cell object.
bins	The number of contours to the overlay
facet.by	meta data variable to facet the comparison

**Value**

ggplot object

**Author(s)**

Francesco Mazziotta, Nick Borchering

**Examples**

```
#Getting the combined contigs
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))

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

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

#Using clonalOverlay()
clonalOverlay(scRep_example,
              reduction = "umap",
              cutpoint = 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,
  order.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 ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order
exportTable	Exports a table of the data into the global. environment in addition to the visualization
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a>

### Value

ggplot of the space occupied by the specific rank of clones

### Examples

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                "P19B", "P19L", "P20B", "P20L"))
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,
  order.by = NULL,
  exportTable = FALSE,
  palette = "inferno"
)
```

**Arguments**

input.data	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
cloneCall	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order
exportTable	Returns the data frame used for forming the graph
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a>

**Value**

ggplot of the total or relative unique clones

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                  "P19B", "P19L", "P20B", "P20L"))

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: 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 <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
cloneCall	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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 <a href="#">hcl.pals</a> .

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))
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 <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
cloneCall	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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 <a href="#">hcl.pals</a> .

**Value**

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

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))

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 <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
cloneCall	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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 <a href="#">hcl.pals</a> .

### Details

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

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

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
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))
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 [igraph::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,
  filterNonproductive = TRUE
)
```

**Arguments**

<code>input.data</code>	List of filtered contig annotations or outputs from <a href="#">loadContigs()</a> .
<code>samples</code>	The labels of samples (required).
<code>ID</code>	The additional sample labeling (optional).
<code>call.related.clones</code>	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.
filterNonproductive	This option will allow for the removal of nonproductive chains if the variable exists in the contig data. Default is set to TRUE to remove nonproductive contigs.

### Value

List of clones for individual cell barcodes

### Examples

```
#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 match the barcodes in the output of the `combineTCR()` or `combineBCR()`.

### Usage

```
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 <code>combineTCR()</code> , <code>combineBCR()</code> or a list of both <code>c(combineTCR(), combineBCR())</code> .
sc.data	The Seurat or Single-Cell Experiment (SCE) object to attach
cloneCall	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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 column label in the combined clones in which clone frequency will be calculated. <b>NULL</b> or " <b>none</b> " will keep the format of input.data.
proportion	Whether to proportion ( <b>TRUE</b> ) or total frequency ( <b>FALSE</b> ) of the clone based on the group.by variable.
filterNA	Method to subset Seurat/SCE object of barcodes without clone information
cloneSize	The bins for the grouping based on proportion or frequency. If proportion is <b>FALSE</b> and the cloneSizes are not set high enough based on frequency, the upper limit of cloneSizes will be automatically updated.S
addLabel	This will add a label to the frequency header, allowing the user to try multiple group.by variables or recalculate frequencies after subsetting the data.

**Value**

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

**Examples**

```
#Getting the combined contigs
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                  "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)
```

---

 combineTCR

*Combining 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.

**Usage**

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

**Arguments**

<code>input.data</code>	List of filtered contig annotations or outputs from <code>loadContigs()</code> .
<code>samples</code>	The labels of samples (recommended).
<code>ID</code>	The additional sample labeling (optional).
<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.
<code>filterNonproductive</code>	This option will allow for the removal of nonproductive chains if the variable exists in the contig data. Default is set to TRUE to remove nonproductive contigs.

**Value**

List of clones for individual cell barcodes

**Examples**

```
combined <- combineTCR(contig_list,
  samples = c("P17B", "P17L", "P18B", "P18L",
    "P19B", "P19L", "P20B", "P20L"))
```

---

`contig_list`

*A list of 8 single-cell T cell receptor sequences runs.*

---

**Description**

A list of 8 `filtered_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()` take 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

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

### Arguments

<code>contig</code>	The filtered contig annotation file from multiplexed experiment
<code>sc.data</code>	The Seurat or Single-Cell Experiment object.
<code>group.by</code>	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.
<code>multi.run</code>	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

```
## 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)
```

---

exportClones     *Exporting clones*

---

### Description

This function saves a csv file of clones (genes, amino acid, and nucleotide sequences) by barcodes. **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**

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

**Arguments**

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

**Value**

CSV file of the paired sequences.

**Author(s)**

Jonathan Noonan, Nick Borcharding

**Examples**

```
## Not run:
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))

exportClones(combined,
             format = "paired")

## End(Not run)
```

---

<code>getCirclize</code>	<i>Generate data frame to be used with circlize R package to visualize clones 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. 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

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

## Arguments

<code>sc.data</code>	The single-cell object after <code>combineExpression()</code> .
<code>cloneCall</code>	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) or a custom variable in the data.
<code>group.by</code>	The group header for which you would like to analyze the data.
<code>proportion</code>	Calculate the relationship unique clones ( <code>proportion = FALSE</code> ) or normalized by proportion ( <code>proportion = TRUE</code> )
<code>include.self</code>	Include counting the clones within a single <code>group.by</code> comparison

## Value

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

## Author(s)

Dillon Corvino, Nick Borcharding

## Examples

```
#Getting the combined contigs  
combined <- combineTCR(contig_list,  
  samples = c("P17B", "P17L", "P18B", "P18L",  
             "P19B", "P19L", "P20B", "P20L"))  
  
#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,  
  group.by = "seurat_clusters")
```

---

getContigDoublets      *Get Contig Doublets*

---

### Description

#### [Experimental]

This function identifies potential doublets by finding common barcodes between TCR and BCR outputs. It extracts unique barcodes from each list of dataframes, finds the intersection of the barcodes, and joins the resulting data.

### Usage

```
getContigDoublets(tcrOutput, bcrOutput)
```

### Arguments

tcrOutput	Output of <code>combineTCR()</code> . A list of data.frames containing TCR contig information, each dataframe must have a barcode column.
bcrOutput	Output of <code>combineBCR()</code> . A list of data.frames containing BCR contig information, each dataframe must have a barcode column.

### Value

A dataframe of barcodes that exist in both the TCR and BCR data, with columns from both sets of data. There will be an additional column `contigType` of type factor with levels 'TCR' and 'BCR' indicating the origin of the contig - this will be the new first column.

If there are no doublets, the returned data.frame will have the same colnames but no rows.

---

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 <code>combineExpression()</code>
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,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))

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

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

#Using highlightClones()
scRep_example <- highlightClones(scRep_example,
                                 cloneCall= "aa",
                                 sequence = c("CVVSDNTGGFKTIF_CASSVRRERANTGELFF"))
```

---

loadContigs

*Loading the contigs derived from single-cell sequencing*

---

**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**

```
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", "Dandelion", "JSON", "MiXCR", "ParseBio", "OmniScope", "TRUST4", "WAT3R", and "Immcantation"

**Details**

The files that this function parses includes:

- 10X = "filtered\_contig\_annotations.csv"
- AIRR = "airr\_rearrangement.tsv"
- BD = "Contigs\_AIRR.tsv"
- Dandelion = "all\_contig\_dandelion.tsv"

- Immcantation = "data.tsv"
- JSON = ".json"
- ParseBio = "barcode\_report.tsv"
- MiXCR = "clones.tsv"
- Omniscope = ".csv"
- TRUST4 = "barcode\_report.tsv"
- WAT3R = "barcode\_results.csv"

### Value

List of contigs for compatibility with `combineTCR()` or `combineBCR()`

### Examples

```
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	<i>Processed subset of contig_list</i>
------------------	--

---

### 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](#).

### Usage

```
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

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

## Arguments

input.data	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
chain	"TRA", "TRB", "TRG", "TRG", "IGH", "IGL".
group.by	The variable to use for grouping.
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order
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 <a href="#">hcl.pals</a> .

## Value

ggplot of stacked bar graphs of amino acid proportions

## Examples

```
#Making combined contig data  
combined <- combineTCR(contig_list,  
  samples = c("P17B", "P17L", "P18B", "P18L",  
             "P19B", "P19L", "P20B", "P20L"))  
  
percentAA(combined,  
  chain = "TRB",  
  aa.length = 20)
```

percentGenes

*Examining the VDJ gene usage across clones***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 [percentVJ\(\)](#).

**Usage**

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

**Arguments**

input.data	The product of <a href="#">combineTCR()</a> , <a href="#">combineBCR()</a> , or <a href="#">combineExpression()</a> .
chain	"TRA", "TRB", "TRG", "TRG", "IGH", "IGL".
gene	"V", "D" or "J"
group.by	The variable to use for grouping
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order
exportTable	Returns the data frame used for forming the graph.
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a> .

**Value**

ggplot of percentage of indicated genes as a heatmap

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))

percentGenes(combined,
             chain = "TRB",
             gene = "Vgene")
```

percentKmer

*Examining the relative composition of kmer motifs in clones.***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,
  order.by = NULL,
  motif.length = 3,
  top.motifs = 30,
  exportTable = FALSE,
  palette = "inferno"
)
```

**Arguments**

input.data	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code>
chain	"TRA", "TRB", "TRG", "TRG", "IGH", "IGL"
cloneCall	How to call the clone - CDR3 nucleotide ( <b>nt</b> ) or CDR3 amino acid ( <b>aa</b> )
group.by	The variable to use for grouping
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order
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 <a href="#">hcl.pals</a>

**Value**

ggplot of percentage of kmers as a heatmap

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
  samples = c("P17B", "P17L", "P18B", "P18L",
    "P19B", "P19L", "P20B", "P20L"))

percentKmer(combined,
  chain = "TRB",
  motif.length = 3)
```

percentVJ

*Quantifying the V and J gene usage across clones***Description**

This function the proportion V and J genes used by grouping variables for an indicated **chain** to produce a matrix of VJ gene pairings.

**Usage**

```
percentVJ(
  input.data,
  chain = "TRB",
  group.by = NULL,
  order.by = NULL,
  exportTable = FALSE,
  palette = "inferno"
)
```

**Arguments**

input.data	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
chain	"TRA", "TRB", "TRG", "TRG", "IGH", "IGL"
group.by	The variable to use for grouping
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order
exportTable	Returns the data frame used for forming the graph
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a> .

**Value**

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

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
  samples = c("P17B", "P17L", "P18B", "P18L",
    "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

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

### Arguments

input.data	The product of <a href="#">combineTCR()</a> , <a href="#">combineBCR()</a> , or <a href="#">combineExpression()</a>
chain	"TRA", "TRB", "TRG", "TRC", "IGH", "IGL"
group.by	The variable to use for grouping
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order
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 <a href="#">hcl.pals</a>

### Value

ggplot of line graph of diversity by position

### Examples

```
#Making combined contig data
combined <- combineTCR(contig_list,
  samples = c("P17B", "P17L", "P18B", "P18L",
             "P19B", "P19L", "P20B", "P20L"))
positionalEntropy(combined,
  chain = "TRB",
  aa.length = 20)
```

---

positionalProperty      *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,
  order.by = NULL,
  aa.length = 20,
  method = "Atchley",
  exportTable = FALSE,
  palette = "inferno"
)
```

### Arguments

input.data	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code>
chain	"TRA", "TRB", "TRG", "TRG", "IGH", "IGL"
group.by	The variable to use for grouping
order.by	A vector of specific plotting order or "alphanumeric" to plot groups in order
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 <a href="#">hcl.pals</a>

### 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



**Author(s)**

Florian Bach, Nick Borcharding

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))

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**

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

**Arguments**

sc.data	The single-cell object after <code>combineExpression()</code> . For SCE objects, the cluster variable must be in the meta data under "cluster".
cloneCall	How to call the clone - VDJC gene ( <b>gene</b> ), CDR3 nucleotide ( <b>nt</b> ), CDR3 amino acid ( <b>aa</b> ), VDJC gene + CDR3 nucleotide ( <b>strict</b> ) 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 <a href="#">hcl.pals</a> .

**Value**

ggplot object of Startrac diversity metrics

**Author(s)**

Liangtao Zheng

**Examples**

```
#Getting the combined contigs
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                "P19B", "P19L", "P20B", "P20L"))

#Getting a sample of a Seurat object
scRep_example <- get(data("scRep_example"))
scRep_example <- combineExpression(combined, scRep_example)
scRep_example$Patient <- substring(scRep_example$orig.ident,1,3)
scRep_example$Type <- substring(scRep_example$orig.ident,4,4)

#Using StartracDiversity()
StartracDiversity(scRep_example,
                 type = "Type",
                 group.by = "Patient")
```

---

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 <code>combineTCR()</code> or <code>combineBCR()</code> .
name	The column header/name to use for subsetting.
variables	The values to subset by, must be in the <code>names(input.data)</code> .

**Value**

list of contigs that have been filtered for the name parameter

**Examples**

```
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))
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**

```
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 <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
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 <a href="#">hcl.pals</a> .

**Value**

ggplot bar diagram or heatmap of gene usage

**Examples**

```
#Making combined contig data
combined <- combineTCR(contig_list,
                      samples = c("P17B", "P17L", "P18B", "P18L",
                                   "P19B", "P19L", "P20B", "P20L"))

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

# Index

## \* Data

contig\_list, 28  
mini\_contig\_list, 34  
scRep\_example, 41

## \* Loading\_and\_Processing\_Contigs

addVariable, 3  
combineBCR, 25  
combineTCR, 27  
createHTOContigList, 29  
exportClones, 29  
loadContigs, 33  
subsetClones, 42

## \* SC\_Functions

alluvialClones, 4  
clonalBias, 6  
clonalNetwork, 14  
clonalOccupy, 15  
clonalOverlay, 18  
combineExpression, 26  
getCirclize, 30  
highlightClones, 32  
StartracDiversity, 41

## \* Summarize\_Repertoire

percentAA, 35  
percentGenes, 36  
percentKmer, 37  
percentVJ, 38  
positionalEntropy, 39  
positionalProperty, 40

## \* Visualizing\_Clones

clonalAbundance, 5  
clonalCluster, 7  
clonalCompare, 8  
clonalDiversity, 10  
clonalHomeostasis, 12  
clonalLength, 13  
clonalOverlap, 16  
clonalProportion, 19  
clonalQuant, 20  
clonalRarefaction, 21  
clonalScatter, 22  
clonalSizeDistribution, 23  
vizGenes, 43

## \* datasets

mini\_contig\_list, 34

## \* internal

scRepertoire-package, 3

addVariable, 3  
alluvialClones, 4

chordDiagram, 31  
clonalAbundance, 5  
clonalBias, 6  
clonalBias(), 6  
clonalCluster, 7  
clonalCompare, 8  
clonalDiversity, 10  
clonalDiversity(), 39  
clonalHomeostasis, 12  
clonalLength, 13  
clonalNetwork, 14  
clonalOccupy, 15  
clonalOverlap, 16  
clonalOverlay, 18  
clonalProportion, 19  
clonalQuant, 20  
clonalRarefaction, 21  
clonalScatter, 22  
clonalSizeDistribution, 23  
combineBCR, 9, 25  
combineBCR(), 3, 5, 7, 8, 10, 12, 13, 17, 19,  
21–24, 26, 27, 29, 30, 32–40, 42, 43  
combineExpression, 9, 26  
combineExpression(), 4, 5, 7, 8, 10, 12–19,  
21–27, 30–32, 35–40, 42, 43  
combineTCR, 9, 27  
combineTCR(), 3, 5, 7, 8, 10, 12, 13, 17, 19,  
21–27, 29, 30, 32–40, 42, 43  
contig\_list, 28  
contig\_list(), 34  
createHTOContigList, 29  
createHTOContigList(), 29  
exportClones, 29  
getCirclize, 30

getContigDoublets, 32

hcl.pals, 4, 6, 7, 9, 11–13, 15–17, 20–24,  
35–40, 42, 44

highlightClones, 32

igraph::components(), 25

iNEXT::iNEXT(), 21

loadContigs, 33

loadContigs(), 25, 28

mini\_contig\_list, 34

percentAA, 35

percentGenes, 36

percentKmer, 37

percentVJ, 38

percentVJ(), 36

positionalEntropy, 39

positionalProperty, 40

scRep\_example, 41

scRepertoire (scRepertoire-package), 3

scRepertoire-package, 3

StartracDiversity, 41

subsetClones, 42

vizGenes, 43