

# Package ‘scRepertoire’

February 4, 2026

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

**Version** 2.7.1

**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, 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 single-cell workflows.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.3

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

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

**Imports** dplyr, evmix, ggalluvial, ggdendro, ggraph, grDevices, igraph, immApex, iNEXT, Matrix, quantreg, Rcpp, rjson, rlang, S4Vectors, SeuratObject, SingleCellExperiment, SummarizedExperiment, tidygraph, purrr, lifecycle, methods

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

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Language** en-US

**LinkingTo** Rcpp

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

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

**Roxygen** list(markdown = TRUE)

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

**git\_branch** devel

**git\_last\_commit** b61e4da

**git\_last\_commit\_date** 2025-12-10

**Repository** Bioconductor 3.23

**Date/Publication** 2026-02-03

**Author** Nick Borcharding [aut, cre],

Qile Yang [aut],

Ksenia Safina [aut],

Justin Reimertz [ctb]

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

## Contents

scRepertoire-package . . . . .	3
._bind_contig_list . . . . .	4
._split_and_pad . . . . .	4
addVariable . . . . .	4
alluvialClones . . . . .	5
annotateInvariant . . . . .	6
clonalAbundance . . . . .	7
clonalBias . . . . .	9
clonalCluster . . . . .	10
clonalCompare . . . . .	13
clonalDiversity . . . . .	14
clonalHomeostasis . . . . .	17
clonalLength . . . . .	18
clonalNetwork . . . . .	19
clonalOccupy . . . . .	21
clonalOverlap . . . . .	22
clonalOverlay . . . . .	24
clonalProportion . . . . .	25
clonalQuant . . . . .	27
clonalRarefaction . . . . .	28
clonalScatter . . . . .	29
clonalSizeDistribution . . . . .	31
combineBCR . . . . .	33
combineExpression . . . . .	35
combineTCR . . . . .	36
contig_list . . . . .	37
createHTOContigList . . . . .	37
exportClones . . . . .	38
expression2List . . . . .	40
getCirclize . . . . .	41
getContigDoublets . . . . .	42
getHumanIgPseudoGenes . . . . .	43
highlightClones . . . . .	43
loadContigs . . . . .	44

percentAA . . . . .	45
percentGeneUsage . . . . .	46
percentKmer . . . . .	50
positionalEntropy . . . . .	51
positionalProperty . . . . .	53
quietVDJgenes . . . . .	55
scRep_example . . . . .	56
StartracDiversity . . . . .	56
subsetClones . . . . .	58

<b>Index</b>	<b>60</b>
--------------	-----------

---

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, 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 single-cell workflows.

## Author(s)

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

Authors:

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

Other contributors:

- Justin Reimertz <justin.reimertz@childrens.harvard.edu> [contributor]

## See Also

Useful links:

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

---

<code>._bind_contig_list</code>	<i>Bind a List of Contig Data Frames and Add Grouping Variable</i>
---------------------------------	--

---

### Description

Bind a List of Contig Data Frames and Add Grouping Variable

### Usage

```
._bind_contig_list(contig_list)
```

---

<code>._split_and_pad</code>	<i>Split String and Pad to a Fixed-Width Matrix</i>
------------------------------	---

---

### Description

Split String and Pad to a Fixed-Width Matrix

### Usage

```
._split_and_pad(x, split, n_cols)
```

---

<code>addVariable</code>	<i>Adding Variables After <code>combineTCR()</code> or <code>combineBCR()</code></i>
--------------------------	--

---

### 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>	A character string that defines the new variable to add.
<code>variables</code>	A character vector defining the desired column value for each list element. Must match the length of the product of <code>combineTCR()</code> or <code>combineBCR()</code> .

**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	<i>Alluvial Plotting for Single-Cell Object</i>
----------------	---

---

**Description**

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

**Usage**

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

**Arguments**

sc.data	The product of <a href="#">combineExpression()</a> .
cloneCall	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
chain	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).

<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>	If TRUE, returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a> .
<code>...</code>	Additional arguments passed to the ggplot theme

### Value

A ggplot object visualizing categorical distribution of clones, or a data.frame if `exportTable = TRUE`.

### 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$Patient <- substr(scRep_example$orig.ident, 1,3)

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

---

annotateInvariant	<i>Annotate invariant T cells (MAIT or iNKT) in single-cell TCR data</i>
-------------------	--

---

### Description

The `annotateInvariant()` function identifies potential mucosal-associated invariant T (MAIT) cells or invariant natural killer T (iNKT) cells from single-cell sequencing datasets based on their characteristic TCR usage. It extracts TCR chain information from the provided single-cell data, checks it against known invariant T-cell receptor criteria for either MAIT or iNKT cells, and returns a score indicating the presence (1) or absence (0) of these invariant cell populations for each individual cell. The function supports data from mouse and human samples, providing a convenient method to annotate specialized T-cell subsets within single-cell analyses.

**Usage**

```
annotateInvariant(
  input.data,
  type = c("MAIT", "iNKT"),
  species = c("mouse", "human")
)
```

**Arguments**

input.data	The product of <code>combineTCR()</code> or <code>combineExpression()</code> .
type	Character specifying the type of invariant cells to annotate (MAIT or iNKT).
species	Character specifying the species ('mouse' or 'human').

**Value**

A single-cell object or list with the corresponding annotation scores (0 or 1) added.

**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)

# Using annotateInvariant()
annotateInvariant(input.data = scRep_example, type = "MAIT", species = "human")
annotateInvariant(input.data = scRep_example, type = "iNKT", species = "human")
```

---

clonalAbundance

*Plot the Relative Abundance of Clones*


---

**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**

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
<code>cloneCall</code>	Defines the clonal sequence grouping. Accepted values are: <code>gene</code> (VDJC genes), <code>nt</code> (CDR3 nucleotide sequence), <code>aa</code> (CDR3 amino acid sequence), or <code>strict</code> (VDJC + nt). A custom column header can also be used.
<code>chain</code>	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
<code>scale</code>	Converts the graphs into density plots in order to show relative distributions.
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed by list element or active identity in the case of single-cell objects.
<code>order.by</code>	A character vector defining the desired order of elements of the <code>group.by</code> variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
<code>exportTable</code>	If TRUE, returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <code>hcl.pals</code> .
<code>...</code>	Additional arguments passed to the ggplot theme

**Value**

A ggplot object visualizing clonal abundance by group, or a data.frame if `exportTable = TRUE`.

**Author(s)**

Nick Borcharding, Justin Reimertz

**Examples**

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



```
# Using clonalAbundance()
clonalAbundance(combined,
                 cloneCall = "gene",
                 scale = FALSE)
```

---

clonalBias

---

*Calculate Clonal Bias 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	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
split.by	The variable to use for calculating the baseline frequencies. For example, "Type" for lung vs peripheral blood comparison
group.by	A column header in the metadata that bias will be based on.
n.boots	number of bootstraps to downsample.
min.expand	clone frequency cut off for the purpose of comparison.
exportTable	If TRUE, returns a data frame or matrix of the results instead of a plot.
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a> .
...	Additional arguments passed to the ggplot theme

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

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

---

clonalCluster

---

*Cluster clones by sequence similarity*


---

**Description**

This function clusters TCRs or BCRs based on the edit distance or alignment score of their CDR3 sequences. It can operate on either nucleotide (nt) or amino acid (aa) sequences and can optionally enforce that clones share the same V and/or J genes. The output can be the input object with an added metadata column for cluster IDs, a sparse adjacency matrix, or an igraph graph object representing the cluster network.

**Usage**

```
clonalCluster(
  input.data,
  chain = "TRB",
  sequence = "aa",
  threshold = 0.85,
  group.by = NULL,
  dist_type = "levenshtein",
  dist_mat = "BLOSUM80",
```

```

    normalize = "length",
    gap_open = -10,
    gap_extend = -1,
    cluster.method = "components",
    cluster.prefix = "cluster.",
    use.V = TRUE,
    use.J = FALSE,
    exportAdjMatrix = FALSE,
    exportGraph = FALSE
)

```

## Arguments

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> or <code>combineExpression()</code> .
<code>chain</code>	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
<code>sequence</code>	Clustering based on either aa or nt sequences.
<code>threshold</code>	The similarity threshold. If $< 1$ , treated as normalized similarity (higher is stricter). If $\geq 1$ , treated as raw edit distance (lower is stricter).
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, clusters will be calculated across all sequences.
<code>dist_type</code>	The distance metric to use. Options: "levenshtein" (default), "hamming", "damerau" (allows transpositions), "nw" (Needleman-Wunsch), or "sw" (Smith-Waterman).
<code>dist_mat</code>	The substitution matrix to use for alignment-based metrics ("nw" or "sw"). Options: "BLOSUM45", "BLOSUM50", "BLOSUM62", "BLOSUM80" (default), "BLOSUM100", "PAM30", "PAM40", "PAM70", "PAM120", "PAM250", or "identity".
<code>normalize</code>	Method for normalizing distances. Options: "none", "maxlen" (divide by max sequence length), or "length" (default, divide by mean sequence length). If $\text{threshold} < 1$ , this controls how the similarity is calculated.
<code>gap_open</code>	Penalty for opening a gap in alignment metrics (default: -10).
<code>gap_extend</code>	Penalty for extending a gap in alignment metrics (default: -1).
<code>cluster.method</code>	The clustering algorithm to use. Defaults to "components", which finds connected subgraphs.
<code>cluster.prefix</code>	A character prefix to add to the cluster names (e.g., "cluster.").
<code>use.V</code>	If TRUE, sequences must share the same V gene to be clustered together.
<code>use.J</code>	If TRUE, sequences must share the same J gene to be clustered together.
<code>exportAdjMatrix</code>	If TRUE, the function returns a sparse adjacency matrix ( <code>dgCMatrix</code> ) of the network.
<code>exportGraph</code>	If TRUE, the function returns an <code>igraph</code> object of the sequence network.

## Details

The clustering process is as follows:

1. The function retrieves the relevant chain data from the input object.
2. It calculates the distance between all sequences within each group (or across the entire dataset if `group.by` is `NULL`).
3. An edge list is constructed, connecting sequences that meet the similarity threshold.
4. The threshold parameter behaves differently based on its value:
  - **threshold < 1 (e.g., 0.85):** Interpreted as a *normalized* distance. A higher value means greater similarity is required.
  - **threshold >= 1 (e.g., 2):** Interpreted as a maximum *raw* edit distance. A lower value means greater similarity is required.
5. **Distance Metrics:**
  - **Levenshtein/Hamming/Damerau:** Standard edit distance calculations.
  - **Alignment (NW/SW):** If `dist_type` is "nw" (Needleman-Wunsch) or "sw" (Smith-Waterman), alignment scores are calculated using the specified substitution matrix (`dist_mat`). These scores are converted to a distance-like metric for clustering.
6. An igraph graph is built from the edge list.
7. A clustering algorithm is run on the graph (default: connected components).

## Value

Depending on the export parameters, one of the following:

- An amended `input.data` object with a new metadata column containing cluster IDs (default).
- An igraph object if `exportGraph = TRUE`.
- A sparse `dgCMatrix` object if `exportAdjMatrix = TRUE`.

## Examples

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

# Standard Levenshtein clustering (85% similarity)
sub_combined <- clonalCluster(combined[c(1,2)],
                             chain = "TRA",
                             sequence = "aa",
                             threshold = 0.85)

# Alignment-based clustering using BLOSUM80
sub_combined_nw <- clonalCluster(combined[c(1,2)],
                                chain = "TRA",
                                dist_type = "nw",
                                dist_mat = "BLOSUM80",
                                threshold = 0.85)
```

```
# Export the graph object instead
graph_obj <- clonalCluster(combined[c(1,2)],
                           chain = "TRA",
                           exportGraph = TRUE)
```

clonalCompare

*Compare Clonal Abundance Across Variables*

## Description

This function visualizes the relative abundance of specific clones across different samples or groups. It is useful for tracking how the proportions of top clones change between conditions. The output can be an alluvial plot to trace clonal dynamics or an area plot to show compositional changes.

## 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

input.data	The product of <a href="#">combineTCR</a> , <a href="#">combineBCR</a> , or <a href="#">combineExpression</a> .
cloneCall	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
chain	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
samples	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>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed by list element or active identity in the case of single-cell objects.
<code>order.by</code>	A character vector defining the desired order of elements of the <code>group.by</code> variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
<code>graph</code>	The type of plot to generate. Accepted values are <code>alluvial</code> (default) or <code>area</code>
<code>proportion</code>	If TRUE (default), the y-axis will represent the proportional abundance of clones. If FALSE, the y-axis will represent raw clone counts.
<code>exportTable</code>	If TRUE, returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a>
<code>...</code>	Additional arguments passed to the ggplot theme

**Value**

A ggplot object visualizing proportions of clones by groupings, or a data.frame if `exportTable = TRUE`.

**Examples**

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

# Using clonalCompares()
clonalCompare(combined,
              top.clones = 5,
              samples = c("P17B", "P17L"),
              cloneCall="aa")
```

---

clonalDiversity

---

*Calculate Clonal Diversity*


---

**Description**

This function calculates a specified diversity metric for samples or groups within a dataset. To control for variations in library size, the function can perform bootstrapping with downsampling. It resamples each group to the size of the smallest group and calculates the diversity metric across multiple iterations, returning the mean value.

**Usage**

```
clonalDiversity(
  input.data,
  cloneCall = "strict",
  metric = "shannon",
  chain = "both",
  group.by = NULL,
  order.by = NULL,
  x.axis = NULL,
  exportTable = FALSE,
  palette = "inferno",
  n.boots = 100,
  return.boots = FALSE,
  skip.boots = FALSE,
  ...
)
```

**Arguments**

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
<code>cloneCall</code>	Defines the clonal sequence grouping. Accepted values are: <code>gene</code> (VDJC genes), <code>nt</code> (CDR3 nucleotide sequence), <code>aa</code> (CDR3 amino acid sequence), or <code>strict</code> (VDJC + nt). A custom column header can also be used.
<code>metric</code>	The diversity metric to calculate. Must be a single string from the list of available metrics (see Details).
<code>chain</code>	The TCR/BCR chain to use. Use <code>both</code> to include both chains (e.g., TRA/TRB). Accepted values: <code>TRA</code> , <code>TRB</code> , <code>TRG</code> , <code>TRD</code> , <code>IGH</code> , <code>IGL</code> , <code>IGK</code> , <code>Light</code> (for both light chains), or <code>both</code> (for TRA/B and Heavy/Light).
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If <code>NULL</code> , data will be analyzed by list element or active identity in the case of single-cell objects.
<code>order.by</code>	A character vector defining the desired order of elements of the <code>group.by</code> variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
<code>x.axis</code>	An additional metadata variable to group samples along the x-axis.
<code>exportTable</code>	If <code>TRUE</code> , returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a> .
<code>n.boots</code>	The number of bootstrap iterations to perform (default is 100).
<code>return.boots</code>	If <code>TRUE</code> , returns all bootstrap values instead of the mean. Automatically enables <code>exportTable</code> .
<code>skip.boots</code>	If <code>TRUE</code> , disables downsampling and bootstrapping. The metric will be calculated on the full dataset for each group. Defaults to <code>FALSE</code> .
<code>...</code>	Additional arguments passed to the <code>ggplot</code> theme

## Details

The function operates by first splitting the dataset by the specified `group.by` variable.

**Downsampling and Bootstrapping:** To make a fair comparison between groups of different sizes, diversity metrics often require normalization. This function implements this by downsampling.

1. It determines the number of clones in the smallest group.
2. For each group, it performs `n.boots` iterations (default = 100).
3. In each iteration, it randomly samples the clones (with replacement) down to the size of the smallest group.
4. It calculates the selected diversity metric on this downsampled set.
5. The final reported diversity value is the mean of the results from all bootstrap iterations.

This process can be disabled by setting `skip.boots = TRUE`.

**Available Diversity Metrics (metric):** The function uses a registry of metrics imported from the `immApex` package. You can select one of the following:

- `"shannon"`: Shannon's Entropy. See [shannon\\_entropy](#).
- `"inv.simpson"`: Inverse Simpson Index. See [inv\\_simpson](#).
- `"gini.simpson"`: Gini-Simpson Index. See [gini\\_simpson](#).
- `"norm.entropy"`: Normalized Shannon Entropy. See [norm\\_entropy](#).
- `"pielou"`: Pielou's Evenness (same as `norm.entropy`). See [pielou\\_evenness](#).
- `"ace"`: Abundance-based Coverage Estimator. See [ace\\_richness](#).
- `"chao1"`: Chao1 Richness Estimator. See [chao1\\_richness](#).
- `"gini"`: Gini Coefficient for inequality. See [gini\\_coef](#).
- `"d50"`: The number of top clones making up 50% of the library. See [d50\\_dom](#).
- `"hill0"`, `"hill1"`, `"hill2"`: Hill numbers of order 0, 1, and 2. See [hill\\_q](#).

## Value

A `ggplot` object visualizing the diversity metric, or a `data.frame` if `exportTable = TRUE`.

## Author(s)

Andrew Malone, Nick Borcharding, Nathan Vanderkraan

## Examples

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

# Calculate Shannon diversity, grouped by sample
clonalDiversity(combined,
  cloneCall = "gene",
  metric = "shannon")
```



```
# Calculate Inverse Simpson without bootstrapping
clonalDiversity(combined,
  cloneCall = "aa",
  metric = "inv.simpson",
  skip.boots = TRUE)
```

---

clonalHomeostasis	<i>Plot Clonal Homeostasis of the Repertoire</i>
-------------------	--

---

## 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

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
<code>cloneSize</code>	The cut points of the proportions.
<code>cloneCall</code>	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
<code>chain</code>	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed by list element or active identity in the case of single-cell objects.

order.by	A character vector defining the desired order of elements of the group.by variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
exportTable	If TRUE, returns a data frame or matrix of the results instead of a plot.
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a> .
...	Additional arguments passed to the ggplot theme

**Value**

A ggplot object visualizing clonal homeostasis, or a data.frame if `exportTable = TRUE`.

**Examples**

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

---

clonalLength

---

*Plot the Distribution of Sequence Lengths*


---

**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**

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code>
<code>cloneCall</code>	Defines the clonal sequence grouping. Accepted values are: <code>nt</code> (CDR3 nucleotide sequence) or <code>aa</code> (CDR3 amino acid sequence)
<code>chain</code>	The TCR/BCR chain to use. Use both to include both chains (e.g., <code>TRA/TRB</code> ). Accepted values: <code>TRA</code> , <code>TRB</code> , <code>TRG</code> , <code>TRD</code> , <code>IGH</code> , <code>IGL</code> , <code>IGK</code> , <code>Light</code> (for both light chains), or <code>both</code> (for <code>TRA/B</code> and <code>Heavy/Light</code> ).
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., <code>"sample"</code> , <code>"treatment"</code> ). If <code>NULL</code> , data will be analyzed by list element or active identity in the case of single-cell objects.
<code>order.by</code>	A character vector defining the desired order of elements of the <code>group.by</code> variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
<code>scale</code>	Converts the graphs into density plots in order to show relative distributions.
<code>exportTable</code>	If <code>TRUE</code> , returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a>
<code>...</code>	Additional arguments passed to the <code>ggplot</code> theme

**Value**

A `ggplot` object visualizing the distributions by length, or a `data.frame` if `exportTable = TRUE`.

**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

---

*Visualize Clonal Network in Dimensional Reductions*


---

**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,
  cloneCall = "strict",
  chain = "both",
  reduction = "umap",
```

```

    group.by = "ident",
    filter.clones = NULL,
    filter.identity = NULL,
    filter.proportion = NULL,
    filter.graph = FALSE,
    exportClones = FALSE,
    exportTable = FALSE,
    palette = "inferno",
    ...
  )

```

## Arguments

<code>sc.data</code>	The single-cell object after <code>combineExpression()</code> .
<code>cloneCall</code>	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
<code>chain</code>	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
<code>reduction</code>	The name of the dimensional reduction of the single-cell object.
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). This will be the nodes overlaid onto the graph.
<code>filter.clones</code>	Use to select the top n clones (e.g., <code>filter.clones`**` = 2000</code> ) or n of clones based on the minimum ( <code>= "min"</code> ).
<code>filter.identity</code>	Display the network for a specific level of the indicated identity.
<code>filter.proportion</code>	Remove clones from the network below a specific proportion.
<code>filter.graph</code>	Remove the reciprocal edges from the half of the graph, allowing for cleaner visualization.
<code>exportClones</code>	Exports a table of clones that are shared across multiple identity groups and ordered by the total number of clone copies.
<code>exportTable</code>	If TRUE, returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a> .
<code>...</code>	Additional arguments passed to the ggplot theme

## 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

---

*Plot cloneSize by Variable in Single-Cell Objects*


---

## 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

<code>sc.data</code>	The single-cell object after <code>combineExpression()</code>
<code>x.axis</code>	The variable in the meta data to graph along the x.axis.
<code>label</code>	Include the number of clone in each category by x.axis variable
<code>facet.by</code>	The column header used for faceting the graph

<code>order.by</code>	A character vector defining the desired order of elements of the <code>group.by</code> variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
<code>proportion</code>	Convert the stacked bars into relative proportion
<code>na.include</code>	Visualize NA values or not
<code>exportTable</code>	If TRUE, returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a>
<code>...</code>	Additional arguments passed to the ggplot theme

**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 combineExpression()
scRep_example <- combineExpression(combined, scRep_example)

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

---

clonalOverlap

*Examining the clonal overlap between groups or samples*


---

**Description**

This functions allows for the calculation and visualizations of various overlap metrics for clones. The methods include overlap coefficient (`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 = c("overlap", "morisita", "jaccard", "cosine", "raw"),
  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>
cloneCall	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
method	The method to calculate the overlap, morisita, jaccard, cosine indices or raw for the base numbers
chain	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
group.by	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed by list element or active identity in the case of single-cell objects.
order.by	A character vector defining the desired order of elements of the group.by variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
exportTable	If TRUE, returns a data frame or matrix of the results instead of a plot.
palette	Colors to use in visualization - input any <code>hcl.pals</code>
...	Additional arguments passed to the ggplot theme

### 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**

A ggplot object visualizing clonal overlap or a data.frame if exportTable = TRUE.

**Examples**

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

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

---

clonalOverlay	<i>Visualize Distribution of Clonal Frequency</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,
  pt.size = 0.5,
  pt.alpha = 1,
  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.



<code>bins</code>	The number of contours to the overlay
<code>pt.size</code>	The point size for plotting (default is 0.5)
<code>pt.alpha</code>	The alpha value for plotting (default is 1)
<code>facet.by</code>	meta data variable to facet the comparison
<code>...</code>	Additional arguments passed to the ggplot theme

**Value**

A ggplot object visualizing distributions of clones along a dimensional reduction within the single-cell object

**Author(s)**

Francesco Mazziotto, 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"))

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

# Using clonalOverlay()
clonalOverlay(scRep_example,
              reduction = "umap",
              cutpoint = 3,
              bins = 5)
```

---

clonalProportion

---

*Plot 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**

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
<code>clonalSplit</code>	The cut points for the specific clones, default = <code>c(10, 100, 1000, 10000, 30000, 1e+05)</code>
<code>cloneCall</code>	Defines the clonal sequence grouping. Accepted values are: <code>gene</code> (VDJC genes), <code>nt</code> (CDR3 nucleotide sequence), <code>aa</code> (CDR3 amino acid sequence), or <code>strict</code> (VDJC + nt). A custom column header can also be used.
<code>chain</code>	The TCR/BCR chain to use. Use <code>both</code> to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or <code>both</code> (for TRA/B and Heavy/Light).
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If <code>NULL</code> , data will be analyzed by list element or active identity in the case of single-cell objects.
<code>order.by</code>	A character vector defining the desired order of elements of the <code>group.by</code> variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
<code>exportTable</code>	If <code>TRUE</code> , returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a>
<code>...</code>	Additional arguments passed to the <code>ggplot</code> theme

**Value**

A `ggplot` object dividing space occupied by ranks of clones or a `data.frame` if `exportTable = TRUE`.

**Examples**

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

# Using clonalProportion()
clonalProportion(combined, cloneCall = "gene")
```

clonalQuant

*Plot Number or Proportions of Clones***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**

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
<code>cloneCall</code>	Defines the clonal sequence grouping. Accepted values are: <code>gene</code> (VDJC genes), <code>nt</code> (CDR3 nucleotide sequence), <code>aa</code> (CDR3 amino acid sequence), or <code>strict</code> (VDJC + nt). A custom column header can also be used.
<code>chain</code>	The TCR/BCR chain to use. Use <code>both</code> to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or <code>both</code> (for TRA/B and Heavy/Light).
<code>scale</code>	Converts the graphs into percentage of unique clones
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If <code>NULL</code> , data will be analyzed by list element or active identity in the case of single-cell objects.
<code>order.by</code>	A character vector defining the desired order of elements of the <code>group.by</code> variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
<code>exportTable</code>	If <code>TRUE</code> , returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a>
<code>...</code>	Additional arguments passed to the <code>ggplot</code> theme

**Value**

A `ggplot` object visualizing the total or relative number of clones or a `data.frame` if `exportTable = TRUE`.

## Examples

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

# Using clonalQuant()
clonalQuant(combined,
            cloneCall="strict",
            scale = TRUE)
```

---

clonalRarefaction	<i>Calculate rarefaction based on the abundance of clones</i>
-------------------	---

---

## 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

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
<code>cloneCall</code>	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
<code>chain</code>	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).

group.by	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed by list element or active identity in the case of single-cell objects.
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 bootstrap replicates used to derive confidence intervals for the diversity estimates. More replicates can provide a more reliable measure of statistical variability.
exportTable	If TRUE, returns a data frame or matrix of the results instead of a plot.
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a> .
...	Additional arguments passed to the ggplot theme

### Examples

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

# Using clonalRarefaction()
clonalRarefaction(combined[c(1,2)], cloneCall = "gene", n.boots = 3)
```

---

clonalScatter

*Scatter Plot Comparing Clones Across 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

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
<code>cloneCall</code>	Defines the clonal sequence grouping. Accepted values are: <code>gene</code> (VDJC genes), <code>nt</code> (CDR3 nucleotide sequence), <code>aa</code> (CDR3 amino acid sequence), or <code>strict</code> (VDJC + nt). A custom column header can also be used.
<code>x.axis</code>	name of the list element to appear on the x.axis.
<code>y.axis</code>	name of the list element to appear on the y.axis.
<code>chain</code>	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
<code>dot.size</code>	either total or the name of the list element to use for size of dots.
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed by list element or active identity in the case of single-cell objects.
<code>graph</code>	graph either the clonal "proportion" or "count".
<code>exportTable</code>	If TRUE, returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a> .
<code>...</code>	Additional arguments passed to the ggplot theme

## Value

A ggplot object visualizing clonal dynamics between two groupings or a data.frame if `exportTable = TRUE`.

## Examples

```

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

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

```

---

clonalSizeDistribution

*Plot powerTCR Clustering Based on Clonal Size*


---

## 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	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
chain	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
method	The clustering parameter for the dendrogram.
threshold	Numerical vector containing the thresholds the grid search was performed over.
group.by	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed as by list element or active identity in the case of single-cell objects.
exportTable	If TRUE, returns a data frame or matrix of the results instead of a plot.
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a> .
...	Additional arguments passed to the ggplot theme

## 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

A ggplot object visualizing dendrogram of clonal size distribution or a data.frame if exportTable = TRUE.

## Author(s)

Hillary Koch

## Examples

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

# Using clonalSizeDistribution()
clonalSizeDistribution(combined,
                       cloneCall = "strict",
                       method="ward.D2")
```



combineBCR

*Combine B Cell Receptor Contig Data***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 based on the edit distance clustering from [clonalCluster\(\)](#). The CTstrict column is formatted as Heavy\_Light (underscore-separated) for downstream compatibility. Connected clones are labeled with cluster.X, while unconnected clones (singlets) are labeled with the V gene and CDR3 sequence (e.g., IGHV3-64.CAKSYS...\_IGKV3-15.CQQYSN...).

**Usage**

```
combineBCR(
  input.data,
  samples = NULL,
  ID = NULL,
  chain = "both",
  sequence = "nt",
  dist_type = "levenshtein",
  dist_mat = "BLOSUM80",
  normalize = "length",
  gap_open = -10,
  gap_extend = -1,
  call.related.clones = TRUE,
  group.by = NULL,
  threshold = 0.85,
  cluster.method = "components",
  use.V = TRUE,
  use.J = TRUE,
  removeNA = FALSE,
  removeMulti = FALSE,
  filterMulti = TRUE,
  filterNonproductive = TRUE
)
```

**Arguments**

input.data	List of filtered contig annotations or outputs from <a href="#">loadContigs()</a> .
samples	A character vector of sample labels. Must be the same length as the input list.
ID	An optional character vector for additional sample identifiers.
chain	The chain to use for clustering when call.related.clones = TRUE. Passed to <a href="#">clonalCluster()</a> . Default is "both".

sequence	The sequence type ("nt" or "aa") to use for clustering. Passed to <code>clonalCluster()</code> . Default is "nt".
dist_type	The distance metric to use. Options: "levenshtein" (default), "hamming", "damerau", "nw" (Needleman-Wunsch), or "sw" (Smith-Waterman).
dist_mat	The substitution matrix to use for alignment-based metrics ("nw" or "sw"). Options include "BLOSUM62", "PAM30", etc.
normalize	Method for normalizing distances. Options: "none" (default), "maxlen", or "length".
gap_open	Penalty for opening a gap in alignment metrics (default: -10).
gap_extend	Penalty for extending a gap in alignment metrics (default: -1).
call.related.clones	Logical. If TRUE, uses <code>clonalCluster()</code> to identify related clones based on sequence similarity. If FALSE, defines clones by the exact V-gene and CDR3 amino acid sequence.
group.by	The column header used for to group clones. If ('NULL'), clusters will be calculated across samples.
threshold	The similarity threshold passed to <code>clonalCluster()</code> if <code>call.related.clones</code> = TRUE. See <code>?clonalCluster</code> for details.
cluster.method	The clustering algorithm to use. Defaults to "components", which finds connected subgraphs.
use.V	Logical. If TRUE, sequences must share the same V gene to be clustered together.
use.J	Logical. If TRUE, sequences must share the same J gene to be clustered together.
removeNA	This will remove any chain without values.
removeMulti	Logical. If TRUE, removes cells that have more than one distinct heavy or light chain after processing.
filterMulti	Logical. If TRUE, filters multi-chain cells to retain only the most abundant IGH and IGL/IGK chains.
filterNonproductive	Logical. If TRUE, removes non-productive contigs from the analysis.

## Value

A list of data frames, where each data frame represents a sample. Each row corresponds to a unique cell barcode, with columns detailing the BCR chains and the assigned clone ID.

## 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)
```

**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	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
chain	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL (for both light chains), both.
group.by	A column header in lists to group the analysis by (e.g., "sample", "treatment"). If NULL, will be based on the list element.
proportion	Whether to proportion (TRUE) or total frequency (FALSE) 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 FALSE and the cloneSizes are not set high enough based on frequency, the upper limit of cloneSizes will be automatically updated.

`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

*Combine T Cell Receptor Contig Data*

---

### 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

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

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

---

contig_list	<i>A List of Eight Single-cell TCR Sequencing Runs.</i>
-------------	---

---

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	<i>Deconvolute Contig Information from Multiplexed Experiments</i>
---------------------	--

---

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**

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

**Value**

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

**Examples**

```
## 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

---

*Export Clonal Data in Various Formats*


---

**Description**

Exports clonal information (gene sequences, amino acids, nucleotides) from `scRepertoire` objects into a file or a data frame. The output format can be tailored for compatibility with different analysis workflows.

**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 for exporting clones. Options are: <code>paired</code> , <code>airr</code> , <code>TCRMatch</code> , <code>tcrapheno</code> , <code>immunarch</code> .
<code>group.by</code>	The variable in the metadata to use for grouping. If <code>NULL</code> , data will be grouped by the sample names.
<code>write.file</code>	If <code>TRUE</code> (default), saves the output to a CSV file. If <code>FALSE</code> , returns the data frame or list to the R environment.
<code>dir</code>	The directory where the output file will be saved. Defaults to the current working directory.
<code>file.name</code>	The name of the file to be saved.

**Details**

The `format` parameter determines the structure of the output:

- `paired`: Exports a data frame where each row represents a barcode, with paired chain information (amino acid, nucleotide, genes) in separate columns.
- `airr`: Exports a data frame that adheres to the Adaptive Immune Receptor Repertoire (AIRR) Community format, with each row representing a single receptor chain.
- `TCRMatch`: Exports a data frame specifically for the TCRMatch algorithm, containing the TRB chain amino acid sequence and clonal frequency.
- `tcrapheno`: Exports a data frame compatible with the `tcrapheno` pipeline, with TRA and TRB chains in separate columns.
- `immunarch`: Exports a list containing a data frame and metadata formatted for use with the `immunarch` package.

**Value**

A data frame or list in the specified format, either returned to the R environment or saved as a CSV file.

**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"))

# Export as a paired data frame and save to a file
exportClones(combined, format = "paired", file.name = "paired_clones.csv")
```

```
# Return an AIRR-formatted data frame to the environment
airr_df <- exportClones(combined, format = "airr", write.file = FALSE)

## End(Not run)
```

---

expression2List	<b>DEPRECATED</b> <i>Take the meta data in seurat/SCE and place it into a list</i>
-----------------	--

---

## Description

### [Deprecated]

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.

This function is deprecated as of version 2 due to the confusion it caused to many users. Users are encouraged to remain with the abstraction barrier of combined single cell objects and the outputs of `combineTCR()` and `combineBCR()` for all functions.

We discourage the use of this function, but if you have to use it, set the force argument to TRUE.

## Usage

```
expression2List(sc, ..., force = FALSE)
```

## Arguments

sc	output of <code>combineExpression()</code> .
...	previously the group or split.by argument, indicating the column header to group the new list by. This should strictly be one argument and is an ellipsis for backwards compatibility. Everything after the first argument is ignored.
force	logical. If not TRUE (default), a deprecation error will be thrown. Otherwise the function will run but not guaranteed to be stable.

## Value

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



---

**getCirclize***Generate Data Frame to Plot Cord Diagram*

---

**Description**

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

**Usage**

```
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>	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
<code>group.by</code>	A column header in the metadata to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed by active identity.
<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 group.by 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	<i>Get Contig Doublets</i>
-------------------	----------------------------

---

## 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.

---

getHumanIgPseudoGenes    *Get Human Immunoglobulin pseudogenes*

---

### Description

This function returns a character vector of human immunoglobulin pseudogenes. These are also the genes that are removed from the variable gene list in [quietVDJgenes\(\)](#).

### Usage

```
getHumanIgPseudoGenes()
```

### Value

Character vector of human immunoglobulin pseudogenes.

---

highlightClones    *Highlighting Specific Clones*

---

### 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 <a href="#">combineExpression()</a>
cloneCall	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
sequence	The specific sequence or sequence to highlight

### Value

Single-cell 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 combineExpression()
scRep_example <- combineExpression(combined,
                                  scRep_example)

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

---

loadContigs

---

Load Immune Receptor Sequencing Contigs

---

## Description

This function loads and processes contig data from various single-cell immune receptor sequencing formats. It reads data from a directory (recursively) or from an already loaded list/data frame, transforms it to a common structure, and returns a list of contigs ready for downstream analysis with `combineTCR()` or `combineBCR()`.

Supported file formats and their expected file names:

- 10X: "filtered\_contig\_annotations.csv"
- AIRR: "airr\_rearrangement.tsv"
- BD: "Contigs\_AIRR.tsv"
- Dandelion: "all\_contig\_dandelion.tsv"
- Immcantation: "\_data.tsv" (or similar)
- "JSON": ".json"
- ParseBio: "barcode\_report.tsv"
- MiXCR: "clones.tsv"
- TRUST4: "barcode\_report.tsv"
- WAT3R: "barcode\_results.csv"

## Usage

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

**Arguments**

input	A directory path containing contig files or a list/data frame of pre-loaded contig data.
format	A string specifying the data format. Must be one of: auto, 10X, AIRR, BD, Dandelion, JSON, MiXCR, ParseBio, TRUST4, WAT3R, or Immcantation. If "auto", the function attempts automatic format detection.

**Value**

A list of contigs formatted for use with `combineTCR()` or `combineBCR()`. Rows containing only NA values (aside from the barcode) are dropped.

**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")
```

percentAA

*Plot 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**

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
<code>chain</code>	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed by list element or active identity in the case of single-cell objects.
<code>order.by</code>	A character vector defining the desired order of elements of the <code>group.by</code> variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
<code>aa.length</code>	The maximum length of the CDR3 amino acid sequence.
<code>exportTable</code>	If TRUE, returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <code>hcl.pals</code> .
<code>...</code>	Additional arguments passed to the ggplot theme

**Value**

A ggplot object visualizing amino acid by proportion or a data.frame if `exportTable = TRUE`.

**Examples**

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

# Using percentAA()
percentAA(combined,
          chain = "TRB",
          aa.length = 20)
```

---

percentGeneUsage

*Summarizes and Visualizes Gene Usage*

---

**Description**

This function quantifies and visualizes the usage of V, D, or J genes, or gene pairings across T or B cell clones.

**Usage**

```
percentGeneUsage(
  input.data,
  chain = "TRB",
  genes = "TRBV",
```

```

    group.by = NULL,
    order.by = NULL,
    summary.fun = c("percent", "proportion", "count"),
    plot.type = "heatmap",
    exportTable = FALSE,
    palette = "inferno",
    ...
)

vizGenes(
  input.data,
  x.axis = "TRBV",
  y.axis = NULL,
  group.by = NULL,
  plot = "heatmap",
  order.by = NULL,
  summary.fun = c("percent", "proportion", "count"),
  exportTable = FALSE,
  palette = "inferno"
)

percentGenes(
  input.data,
  chain = "TRB",
  gene = "Vgene",
  group.by = NULL,
  order.by = NULL,
  exportTable = FALSE,
  summary.fun = c("percent", "proportion", "count"),
  palette = "inferno"
)

percentVJ(
  input.data,
  chain = "TRB",
  group.by = NULL,
  order.by = NULL,
  summary.fun = c("percent", "proportion", "count"),
  exportTable = FALSE,
  palette = "inferno"
)

```

### Arguments

input.data	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code> .
chain	The TCR/BCR chain to use. Accepted values: TRA, TRB, TRG, TRD, IGH, IGL (for both light chains)
genes	A character vector specifying the gene loci to analyze. Can be a single gene e.g.,

	"TRBV" or "IGHJ" or a pair for genes analysis (e.g., c("TRBV", "TRAV"), or "TRBV", "TRBJ").
group.by	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed as by list element or active identity in the case of single-cell objects.
order.by	A character vector defining the desired order of elements of the group.by variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
summary.fun	Character string choosing the summary statistic - "percent" (default), "proportion", or "count".
plot.type	The type of plot to return: "heatmap" (default for paired loci, also available for single loci), or "barplot" (for single loci).
exportTable	If TRUE, returns a data frame or matrix of the results instead of a plot.
palette	Colors to use in visualization - input any <a href="#">hcl.pals</a> .
...	Additional arguments passed to the ggplot theme
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.
plot	The type of plot to return - heatmap or barplot.
gene	Vgene, Dgene or Jgene

## Value

A ggplot object displaying a heatmap or bar plot of gene usage. If `exportTable = TRUE`, a matrix or data frame of the raw data is returned.

## Examples

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

# Visualize single gene (TRBV) usage as a heatmap, grouped by sample
percentGeneUsage(combined,
                 genes = "TRBV",
                 group.by = "sample",
                 plot.type = "heatmap",
                 summary.fun = "percent")

# Visualize single gene (TRBV) usage as a barplot, grouped by sample
percentGeneUsage(combined,
                 genes = "TRBV",
                 group.by = "sample",
                 plot.type = "barplot",
                 summary.fun = "count")

# Visualize paired gene (TRBV-TRBJ) usage as a heatmap
```



```
percentGeneUsage(combined[1:2],
  genes = c("TRBV", "TRBJ"),
  group.by = "sample",
  plot.type = "heatmap",
  summary.fun = "proportion")

# Export the raw data table for single gene usage
trbv_usage_table <- percentGeneUsage(combined,
  genes = "TRBV",
  group.by = "sample",
  exportTable = TRUE,
  summary.fun = "count")

# Export the raw data table for paired gene usage
trbv_trbj_usage_table <- percentGeneUsage(combined,
  genes = c("TRBV", "TRBJ"),
  group.by = "sample",
  exportTable = TRUE,
  summary.fun = "percent")

# Visualize paired gene (TRAV-TRAJ) usage as a heatmap
vizGenes(combined[1:2],
  x.axis = "TRAV",
  y.axis = "TRAJ",
  group.by = "sample",
  summary.fun = "count")

# Visualize cross-chain gene pairing (TRBV-TRAV)
vizGenes(combined[1:2],
  x.axis = "TRBV",
  y.axis = "TRAV",
  group.by = "sample",
  summary.fun = "percent")

# Quantify and visualize TRA V-gene usage as a heatmap
percentGenes(combined,
  chain = "TRA",
  gene = "Vgene",
  group.by = "sample",
  summary.fun = "percent")

# Quantify TRA J-gene usage and export the table
traj_usage_table <- percentGenes(combined,
  chain = "TRA",
  gene = "Jgene",
  group.by = "sample",
  exportTable = TRUE,
  summary.fun = "count")

# Quantify and visualize TRB V-J gene pairings as a heatmap
percentVJ(combined[1:2],
```

```

      chain = "TRB",
      group.by = "sample",
      summary.fun = "percent")

# 2. Quantify TRA V-J gene pairings and export the table
trav_traj_table <- percentVJ(combined,
                             chain = "TRA",
                             group.by = "sample",
                             exportTable = TRUE,
                             summary.fun = "proportion")

```

percentKmer

*Analyze K-mer Motif Composition*

## Description

This function calculates and visualizes the frequency of k-mer motifs for either nucleotide (nt) or amino acid (aa) sequences. It produces a heatmap showing the relative composition of the most variable motifs across samples or groups.

## Usage

```

percentKmer(
  input.data,
  chain = "TRB",
  cloneCall = "aa",
  group.by = NULL,
  order.by = NULL,
  motif.length = 3,
  min.depth = 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	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
cloneCall	Defines the clonal sequence grouping. Accepted values are: nt (CDR3 nucleotide sequence) or aa (CDR3 amino acid sequence).
group.by	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed as by list element or active identity in the case of single-cell objects.

<code>order.by</code>	A character vector defining the desired order of elements of the <code>group.by</code> variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
<code>motif.length</code>	The length of the kmer to analyze
<code>min.depth</code>	Minimum count a motif must reach to be retained in the output ( $\geq 1$ ). <b>Default:</b> 3.
<code>top.motifs</code>	Return the n most variable motifs as a function of median absolute deviation
<code>exportTable</code>	If TRUE, returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a>
<code>...</code>	Additional arguments passed to the <code>ggplot</code> theme

### Details

The function first calculates k-mer frequencies for each sample/group. By default, it then identifies the 30 most variable motifs based on the Median Absolute Deviation (MAD) across all samples and displays their frequencies in a heatmap.

### Value

A `ggplot` object displaying a heatmap of motif percentages. If `exportTable = TRUE`, a matrix of the raw data is returned.

### Examples

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

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

### 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**

<code>input.data</code>	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code>
<code>chain</code>	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed as by list element or active identity in the case of single-cell objects.
<code>order.by</code>	A character vector defining the desired order of elements of the <code>group.by</code> variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
<code>aa.length</code>	The maximum length of the CDR3 amino acid sequence.
<code>method</code>	The method to calculate the entropy/diversity - "shannon", "inv.simpson", "gini.simpson", "norm.entropy", "pielou", "hill0", "hill1", "hill2"
<code>exportTable</code>	If TRUE, returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a>
<code>...</code>	Additional arguments passed to the ggplot theme

**Value**

A ggplot object displaying entropy or diversity by amino acid position. If `exportTable = TRUE`, a matrix of the raw data is returned.

**Examples**

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

# Using positionalEntropy()
positionalEntropy(combined,
  chain = "TRB",
  aa.length = 20)
```

---

positionalProperty      *Plot Positional Physicochemical Property Analysis*


---

## Description

This function analyzes the physicochemical properties of amino acids at each position along the CDR3 sequence. It calculates the mean property value and the 95% confidence interval for each position across one or more groups, visualizing the results as a line plot with a confidence ribbon.

## Usage

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

## Arguments

input.data	The product of <code>combineTCR()</code> , <code>combineBCR()</code> , or <code>combineExpression()</code>
chain	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
group.by	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed by list element or active identity in the case of single-cell objects.
order.by	A character vector defining the desired order of elements of the group.by variable. Alternatively, use <code>alphanumeric</code> to sort groups automatically.
aa.length	The maximum length of the CDR3 amino acid sequence.
method	Character string (one of the supported names) Defaults to "atchleyFactors", but includes: "crucianiProperties", "FASGAI", "kideraFactors", "MSWHIM", "ProtFP", "stScales", "tScales", "VHSE", "zScales"
exportTable	If TRUE, returns a data frame or matrix of the results instead of a plot.
palette	Colors to use in visualization - input any <code>hcl.pals</code>
...	Additional arguments passed to the <code>ggplot</code> theme

**Details**

The function uses one of several established physicochemical property scales to convert amino acid sequences into numerical vectors. More information for the individual methods can be found at the following citations:

**atchleyFactors:** [citation](#)

**crucianiProperties:** [citation](#)

**FASGAI:** [citation](#)

**kideraFactors:** [citation](#)

**MSWHIM:** [citation](#)

**ProtFP:** [citation](#)

**stScales:** [citation](#)

**tScales:** [citation](#)

**VHSE:** [citation](#)

**zScales:** [citation](#)

**Value**

A ggplot object displaying property by amino acid position. If `exportTable = TRUE`, a matrix of the raw data is returned.

**Author(s)**

Florian Bach, Nick Borchering

**Examples**

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

# Using positionalProperty()
positionalProperty(combined,
                  chain = "TRB",
                  method = "atchleyFactors",
                  aa.length = 20)
```

---

quietVDJgenes*Remove TCR and BCR genes from variable gene results*

---

## Description

Most single-cell workflows use highly-expressed and highly-variable genes for the initial calculation of PCA and subsequent dimensional reduction. This function will remove the TCR and/or BCR genes from the variable features in a Seurat object or from a vector (potentially generated by the Bioconductor scran workflow).

## Usage

```
quietVDJgenes(input.data, ...)  
  
quietTCRgenes(input.data, ...)  
  
## Default S3 method:  
quietTCRgenes(input.data, ...)  
  
## S3 method for class 'Seurat'  
quietTCRgenes(input.data, assay = NULL, ...)  
  
quietBCRgenes(input.data, ...)  
  
## Default S3 method:  
quietBCRgenes(input.data, ...)  
  
## S3 method for class 'Seurat'  
quietBCRgenes(input.data, assay = NULL, ...)
```

## Arguments

input.data	Single-cell object in Seurat format or vector of variable genes to use in reduction
...	Reserved for future arguments
assay	The Seurat assay slot to use to remove immune receptor genes from, NULL value will default to the default assay

## Value

Seurat object or vector list with TCR genes removed.

## Author(s)

Nicky de Vrij, Nikolaj Pagh, Nick Borcharding, Qile Yang

## Examples

```
example <- quietVDJgenes(scRep_example)
scRep <- quietTCRgenes(scRep_example)
ibex_example <- quietBCRgenes(scRep_example)
```

---

scRep_example	<i>A Seurat Object of 500 Single T cells,</i>
---------------	---

---

## 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	<i>Calculate Startrac-based Diversity Indices</i>
-------------------	---

---

## Description

This function utilizes the STARTRAC approach to calculate T cell diversity metrics based on the work of Zhang et al. (2018, Nature) [PMID: 30479382](#). It can compute three distinct indices: clonal expansion (`expa`), cross-tissue migration (`migr`), and state transition (`tran`).

## Usage

```
StartracDiversity(
  sc.data,
  cloneCall = "strict",
  chain = "both",
  index = c("expa", "migr", "tran"),
  type = NULL,
  group.by = NULL,
  pairwise = NULL,
  exportTable = FALSE,
  palette = "inferno",
  ...
)
```



**Arguments**

<code>sc.data</code>	The single-cell object after <code>combineExpression()</code> . For SCE objects, the cluster variable must be in the meta data under "cluster".
<code>cloneCall</code>	Defines the clonal sequence grouping. Accepted values are: gene (VDJC genes), nt (CDR3 nucleotide sequence), aa (CDR3 amino acid sequence), or strict (VDJC + nt). A custom column header can also be used.
<code>chain</code>	The TCR/BCR chain to use. Use both to include both chains (e.g., TRA/TRB). Accepted values: TRA, TRB, TRG, TRD, IGH, IGL, IGK, Light (for both light chains), or both (for TRA/B and Heavy/Light).
<code>index</code>	A character vector specifying which indices to calculate. Options: "expa", "migr", "tran". Default is all three.
<code>type</code>	The metadata variable that specifies tissue type for migration analysis.
<code>group.by</code>	A column header in the metadata or lists to group the analysis by (e.g., "sample", "treatment"). If NULL, data will be analyzed as by list element or active identity in the case of single-cell objects.
<code>pairwise</code>	The metadata column to be used for pairwise comparisons. Set to the type variable for pairwise migration or "cluster" for pairwise transition.
<code>exportTable</code>	If TRUE, returns a data frame or matrix of the results instead of a plot.
<code>palette</code>	Colors to use in visualization - input any <a href="#">hcl.pals</a> .
<code>...</code>	Additional arguments passed to the ggplot theme

**Details**

The function requires a type variable in the metadata, which specifies the tissue origin or any other categorical variable for migration analysis.

**Indices:**

- **expa (Clonal Expansion):** Measures the extent of clonal proliferation within a T cell cluster. It is calculated as 1 - normalized Shannon entropy. A higher value indicates greater expansion of a few clones.
- **migr (Cross-Tissue Migration):** Quantifies the movement of clonal T cells across different tissues (as defined by the type parameter). It is based on the entropy of a clonotype's distribution across tissues.
- **tran (State Transition):** Measures the developmental transition of clonal T cells between different functional clusters. It is based on the entropy of a clonotype's distribution across clusters.

**Pairwise Analysis:** The pairwise parameter enables the calculation of migration or transition between specific pairs of tissues or clusters, respectively.

- For migration (`index = "migr"`), set `pairwise` to the type column (e.g., `pairwise = "Type"`).
- For transition (`index = "tran"`), set `pairwise` to "cluster".

**Value**

A ggplot object visualizing STARTRAC diversity metrics or `data.frame` if `exportTable = TRUE`.

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)

# Calculate a single index (expansion)
StartracDiversity(scRep_example,
                  type = "Type",
                  group.by = "Patient",
                  index = "expa")

# Calculate pairwise transition
StartracDiversity(scRep_example,
                  type = "Type",
                  group.by = "Patient",
                  index = "tran",
                  pairwise = "cluster")
```

---

subsetClones	<i>Subset The Product of combineTCR() or combineBCR()</i>
--------------	---

---

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 in the variable.

**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"))
```

# Index

## \* Data

contig\_list, [37](#)  
scRep\_example, [56](#)

## \* Loading\_and\_Processing\_Contigs

addVariable, [4](#)  
combineBCR, [33](#)  
combineTCR, [36](#)  
createHTOContigList, [37](#)  
exportClones, [38](#)  
subsetClones, [58](#)

## \* SC\_Functions

alluvialClones, [5](#)  
clonalBias, [9](#)  
clonalNetwork, [19](#)  
clonalOccupy, [21](#)  
clonalOverlay, [24](#)  
combineExpression, [35](#)  
getCirclize, [41](#)  
highlightClones, [43](#)  
StartracDiversity, [56](#)

## \* Summarize\_Repertoire

percentAA, [45](#)  
percentGeneUsage, [46](#)  
percentKmer, [50](#)  
positionalEntropy, [51](#)  
positionalProperty, [53](#)

## \* Visualizing\_Clones

clonalAbundance, [7](#)  
clonalCluster, [10](#)  
clonalCompare, [13](#)  
clonalDiversity, [14](#)  
clonalHomeostasis, [17](#)  
clonalLength, [18](#)  
clonalOverlap, [22](#)  
clonalProportion, [25](#)  
clonalQuant, [27](#)  
clonalRarefaction, [28](#)  
clonalScatter, [29](#)  
clonalSizeDistribution, [31](#)

percentGeneUsage, [46](#)

## \* internal

.\_bind\_contig\_list, [4](#)  
.\_split\_and\_pad, [4](#)  
expression2List, [40](#)  
getHumanIgPseudoGenes, [43](#)  
scRepertoire-package, [3](#)  
.\_bind\_contig\_list, [4](#)  
.\_split\_and\_pad, [4](#)

ace\_richness, [16](#)  
addVariable, [4](#)  
alluvialClones, [5](#)  
annotateInvariant, [6](#)  
annotateInvariant(), [6](#)

chao1\_richness, [16](#)  
chordDiagram, [41](#)  
clonalAbundance, [7](#)  
clonalBias, [9](#)  
clonalBias(), [9](#)  
clonalCluster, [10](#)  
clonalCluster(), [33](#)  
clonalCompare, [13](#)  
clonalDiversity, [14](#)  
clonalDiversity(), [51](#)  
clonalHomeostasis, [17](#)  
clonalLength, [18](#)  
clonalNetwork, [19](#)  
clonalOccupy, [21](#)  
clonalOverlap, [22](#)  
clonalOverlay, [24](#)  
clonalProportion, [25](#)  
clonalQuant, [27](#)  
clonalRarefaction, [28](#)  
clonalScatter, [29](#)  
clonalSizeDistribution, [31](#)  
combineBCR, [13](#), [33](#)  
combineBCR(), [4](#), [8](#), [11](#), [15](#), [17](#), [19](#), [23](#), [26–28](#),  
[30](#), [31](#), [35](#), [38](#), [40](#), [42](#), [44–47](#), [50](#), [52](#),

53, 58  
combineExpression, 13, 35  
combineExpression(), 5, 7–9, 11, 15, 17,  
19–21, 23, 24, 26–28, 30, 31, 33, 35,  
36, 40, 41, 43, 46, 47, 50, 52, 53, 57  
combineTCR, 13, 36  
combineTCR(), 4, 7, 8, 11, 15, 17, 19, 23,  
26–28, 30, 31, 33, 35, 38, 40, 42,  
44–47, 50, 52, 53, 58  
contig\_list, 37  
createHTOContigList, 37  
createHTOContigList(), 37  
  
d50\_dom, 16  
  
exportClones, 38  
expression2List, 40  
  
getCirclize, 41  
getContigDoublets, 42  
getHumanIgPseudoGenes, 43  
gini\_coef, 16  
gini\_simpson, 16  
  
hcl.pals, 6, 8, 9, 14, 15, 18–20, 22, 23, 26,  
27, 29–31, 46, 48, 51–53, 57  
highlightClones, 43  
hill\_q, 16  
  
iNEXT::iNEXT(), 28  
inv\_simpson, 16  
  
loadContigs, 44  
loadContigs(), 33, 37  
  
norm\_entropy, 16  
  
percentAA, 45  
percentGenes (percentGeneUsage), 46  
percentGeneUsage, 46  
percentKmer, 50  
percentVJ (percentGeneUsage), 46  
pielou\_evenness, 16  
positionalEntropy, 51  
positionalProperty, 53  
  
quietBCRgenes (quietVDJgenes), 55  
quietTCRgenes (quietVDJgenes), 55  
quietVDJgenes, 55  
quietVDJgenes(), 43  
  
scRep\_example, 56  
scRepertoire (scRepertoire-package), 3  
scRepertoire-package, 3  
shannon\_entropy, 16  
StartracDiversity, 56  
subsetClones, 58  
  
vizGenes (percentGeneUsage), 46