Package 'Platypus'

July 21, 2025

Type Package

Title Single-Cell Immune Repertoire and Gene Expression Analysis

Description We present 'Platypus', an open-source software platform providing a user-friendly interface to investigate B-cell receptor and T-cell receptor repertoires from scSeq experiments. 'Platypus' provides a framework to automate and ease the analysis of single-cell immune repertoires while also incorporating transcriptional information involving unsupervised clustering, gene expression and gene ontology. This R version of 'Platypus' is part of the 'ePlatypus' ecosystem for computational analysis of immunogenomics data: Yermanos et al. (2021) <doi:10.1093/nargab/lqab023>, Cotet et al. (2023) <doi:10.1093/bioinformatics/btad553>.

Version 3.6.0

Date 2024-10-17

Maintainer Alexander Yermanos <daphne.v.ginneken@gmail.com>

Depends R(>= 4.0.0)

Imports Biostrings, cowplot, dplyr, ggalluvial, ggplot2, ggseqlogo, ggtree, jsonlite, knitr, magrittr, Matrix(>= 1.3-3), plyr, reshape2, seqinr, Seurat, SeuratObject(>= 4.1.3), stringdist, stringr, tibble, tidyr, utils, useful

Suggests AnnotationDbi, ape, BiocGenerics, biomaRt, circlize, cluster, doParallel, fgsea, ggrepel, ggridges, gridExtra, harmony, igraph, iNEXT, limma, kmer, msigdbr, phangorn, pheatmap, phytools, purrr, readr, readxl, rstudioapi, Rtsne, scales, sf, SingleCellExperiment, slingshot, tidytree, tidyselect, tidyverse, umap, vegan, viridis, testthat (>= 3.0.0),

License GPL-2

Encoding UTF-8

RoxygenNote 7.3.2

VignetteBuilder knitr

LazyData true

Config/testthat/edition 3

NeedsCompilation no

Author Alexander Yermanos [aut, cre],
Andreas Agrafiotis [ctb],
Victor Kreiner [ctb],
Tudor-Stefan Cotet [ctb],
Raphael Kuhn [ctb],
Danielle Shlesinger [ctb],
Jiami Han [ctb],
Vittoria Martinolli D'Arcy [ctb],
Lucas Stalder [ctb],
Daphne van Ginneken [ctb]

Repository CRAN

Date/Publication 2024-10-18 11:10:17 UTC

Contents

`GEX_cluster_genes` *Differentially expressed genes between clusters or data subsets*

Description

For more flexibility consider GEX_DEgenes(). Extracts the differentially expressed genes between two samples. This function uses the FindMarkers function from the Seurat package. Further parameter control can be accomplished by calling the function directly on the output of automate_GEX or VDJ_GEX_matrix.

Usage

```
GEX_cluster_genes(GEX, min.pct, filter, base, platypus.version)
```

Arguments

GEX	Output Seurat object of either automate_GEX for platypus.version v2 or of VDJ_GEX_matrix for platypus.version v3 (usually VDJ_GEX_matrix.output[[2]])
min.pct	The minimum percentage of cells expressing a gene in either of the two groups to be compared. Default is 0.25
filter	Character vector of initials of the genes to be filtered. Default is c("MT-", "RPL", "RPS"), which filters mitochondrial and ribosomal genes.
base	The base with respect to which logarithms are computed. Default: 2
platypus.version	is set automatically

Value

Returns a dataframe containing the output from the FindMarkers function, which contains information regarding the genes that are differentially regulated, statistics (p value and log fold change), and the percent of cells expressing the particular gene. Ech element in the list corresponds to the clusters in numerical order. For example, the first element in the list output[[1]] corresponds to the genes deferentially expressed in cluster 0 in GEX

Examples

```
try({GEX_cluster_genes(GEX = subset(Platypus::small_vgm[[2]],
  seurat_clusters %in% c(0,1)), min.pct = .25,
  filter = c("MT-", "RPL", "RPS"))})
```

GEX_cluster_genes_heatmap

Heatmap of cluster defining genes

Description

Produces a heatmap displaying the expression of the top genes that define each cluster in the Seurat object. The output heatmap is derived from DoHeatmap from Seurat and thereby can be edited using typical ggplot interactions. The number of genes per cluster and the nunber of cells to display can be specified by the user. Either the log fold change or the p value can be used to select the top n genes.

Usage

```
GEX_cluster_genes_heatmap(
  GEX,
  GEX_cluster_genes.output,
  n.genes.per.cluster,
  metric,
  max.cell,
  group.colors,
  slot,
  platypus.version
)
```

Arguments

GEX	Output Seurat object of either automate_GEX for platypus.version v2 or of VDJ_GEX_matrix for platypus.version v3 (usually VDJ_GEX_matrix.output[[2]])
GEX_cluster_genes.output	The output from the GEX_cluster_genes function - this should be a list with each list element corresponding to the genes, p values, logFC, pct expression for the genes deferentially regulated for each cluster.
n.genes.per.cluster	An integer value determining how many genes per cluster to display in the output heatmap. This number should be adjusted based on the number of clusters. Too many genes per cluster and clusters may cause a problem with the heatmap function in Seurat.
metric	The metric that dictates which are the top n genes returned. Possible options are "p.value" (default), "avg_logFC", "top_logFC", "bottom_logFC". "top_logFC" returns the top expressed genes for each cluster, whereas "bottom_logFC" returns the least expressed genes per cluster-both by log fold change.
max.cell	The max number of cells to display in the heatmap for each cluster, which corresponds to the number of columns. Default is set to 100 cells per cluster.
group.colors	Optional character vector. Array of colors with the same length as GEX_cluster_genes.output to color bars above the heatmap. Defaults to rainbow palette
slot	Seurat object slot from which to plot gene expression data.
platypus.version	is set automatically

Value

Returns a heatmap from the function DoHeatmap from the package Seurat, which is a ggplot object that can be modified or plotted. The number of genes is determined by the n.genes parameter and the number of cells per cluster is determined by the max.cell argument. This function gives a visual description of the top genes differentially expressed in each cluster.

Examples

```
try({
```

```
GEX_cluster_genes_output <- GEX_cluster_genes(GEX =
subset(Platypus::small_vgm[[2]],
seurat_clusters %in% c(0,1)), min.pct = .25
, filter = c("MT-", "RPL", "RPS"))

cluster_defining_gene_heatmap <- GEX_cluster_genes_heatmap(GEX =
Platypus::small_vgm[[2]]
,GEX_cluster_genes.output=GEX_cluster_genes_output
,n.genes.per.cluster=5,metric="p.value",max.cell=5)
})
```

`GEX_cluster_membership` *Cluster membership plots by sample*

Description

Plots the cluster membership for each of the distinct samples in the Seurat object from the automate_GEX function. The distinct samples are determined by "sample_id" field in the Seurat object.

Usage

```
GEX_cluster_membership(GEX, by.group, platypus.version)
```

Arguments

`GEX`	Output Seurat object containing gene expression data from automate_GEX (platypus.version = "v2") or VDJ_GEX_matrix (platypus.version = "v3", usually VDJ_GEX_matrix.output[[2]] that contained at least two distinct biological samples. The different biological samples correspond to integer values (v2) or factor values (v3) in the order of the working directories initially supplied to the automate_GEX function.
`by.group`	Logical indicating whether to look at the cluster distribution per group (using the group_id column). Default is set to FALSE.
`platypus.version`	Version of platypus to use. Defaults to "v2". If an output of the GEX_automate function is supplied, set to "v2". If an output of the VDJ_GEX_matrix function is supplied set to "v3"

Value

Returns a ggplot object in which the values on the x axis correspond to each cluster found in the Seurat object. The y axis corresponds to the percentage of cells found in each cluster. The bar and color corresponds to the distinct sample_id.

Examples

```
try({GEX_cluster_membership(GEX= Platypus::small_vgm[[2]],
platypus.version = "v3")})
```

GEX_coexpression_coefficient

Coexpression of selected genes

Description

Returns eiter a plot or numeric data of coexpression levels of selected genes.Coexpression % is calculated as the quotient of double positive cells (counts \> 0) and the sum of total cells positive for either genes.

Usage

```
GEX_coexpression_coefficient(GEX, genes, subsample.n, plot.dotmap)
```

Arguments

GEX	GEX seurat object generated with VDJ_GEX_matrix (VDJ_GEX_matrix.output\[\[2\]\])
genes	Character vector. At least 2 genes present in rownames(GEX). Use "all" to include all genes. The number of comparisons to make is the length(genes)! (factorial). More than 100 genes are not recommended.
subsample.n	Interger. Number of cells to subsample. If set to 100, 100 cells will be randomly sampled for the calculation
plot.dotmap	Boolean. Whether to return a plot

Value

Returns a dataframe if pot.dotmap == FALSE or a ggplot if plot.dotmap == TRUE detailing the coexpression levels of selected genes within the given cell population

Examples

```
GEX_coexpression_coefficient(GEX = Platypus::small_vgm[[2]]
, genes = c("CD19", "CD83"), subsample.n = "none", plot.dotmap = FALSE)
```

GEX_DEgenes *Wrapper for differential gene expression analysis and plotting*

Description

Extracts the differentially expressed genes between two groups of cells. These groups are defined as cells having either of two entries (group1, group2) in the grouping.column of the input Seurat object metadata This function uses the FindMarkers function from the Seurat package.

Usage

```
GEX_DEgenes(
  GEX,
  FindMarkers.out,
  grouping.column,
  group1,
  group2,
  min.pct,
  filter,
  return.plot,
  logFC,
  color.p.threshold,
  color.log.threshold,
  color.by.threshold,
  up.genes,
  down.genes,
  base,
  label.n.top.genes,
  genes.to.label,
  platypus.version,
  size.top.colorbar
)
```

Arguments

GEX	Output Seurat object from automate_GEX or VDJ_GEX_matrix_function (VDJ_GEX_matrix.output[[2]] function that contained at least two distinct biological groups.
FindMarkers.out	OPTIONAL: the output of the FindMarkers function. This skips the DEG calculation step and outputs desired plots. All plotting parameters function as normal. Grouping parameters and min.pct are ignored.
grouping.column	Character. A column name of GEX@meta.data. In this column, group1 and group2 should be found. Defaults to "sample_id". Could also be set to "seurat_clusters" to generate DEGs between cells of 2 chosen clusters.
group1	either character or integer specifying the first group of cells that should be compared. (e.g. "s1" if sample_id is used as grouping.column)

group2 either character or integer specifying the first group of cells that should be compared. (e.g. "s2" if sample_id is used as grouping.column)

min.pct The minimum percentage of cells expressing a gene in either of the two groups to be compared.

filter Character vector of initials of the genes to be filtered. Default is c("MT-", "RPL", "RPS"), which filters mitochondrial and ribosomal genes.

return.plot Character specifying if a "heatmap", "heatmap" or a "volcano" or "none" is to be returned. If not "none" then @return is a list where the first element is a dataframe and the second a plot (see @return). Defaults to none

logFC Logical specifying whether the genes will be displayed based on logFC (TRUE) or pvalue (FALSE).

color.p.threshold
numeric specifying the adjusted p-value threshold for geom_points to be colored. Default is set to 0.01.

color.log.threshold
numeric specifying the absolute logFC threshold for geom_points to be colored. Default is set to 0.25.

color.by.threshold
Boolean. Set to TRUE to color by color.p.threshold and color.log.threshold. Set to FALSE for a continuous color scale by fold change.

up.genes FOR HEATMAP Integer specifying the number of upregulated genes to be shown.

down.genes FOR HEATMAP Integer specifying the number of downregulated genes to be shown.

base The base with respect to which logarithms are computed. Default: 2

label.n.top.genes
FOR VOLCANO Interger. How many top genes to label either by Fold change (if logFC ==TRUE) or by p.value (if logFC == FALSE). More than 50 are not recommended. Also works in conjunction with genes.to.label

genes.to.label FOR VOLCANO Character vector of genes to label irregardless of their p value.

platypus.version
Function works with V2 and V3, no need to set this parameter

size.top.colorbar
Integer. Size of the top colorbar for heatmap plot.

Value

Returns a dataframe containing the output from the FindMarkers function, which contains information regarding the genes that are differentially regulated, statistics (p value and log fold change), and the percent of cells expressing the particular gene for both groups.

Examples

```
try({DEGs <- GEX_DEgenes(GEX = Platypus::small_vgm[[2]],min.pct = .25,
group1 = "s1",group2 = "s2", return.plot = "volcano")})
```

GEX_dottile_plot *GEX Dottile plots*

Description

Outputs a dotplot for gene expression, where the color of each dot is scaled by the gene expression level and the size is scaled by the % of cells positive for the gene

Usage

```
GEX_dottile_plot(GEX, genes, group.by, threshold.to.plot, platypus.version)
```

Arguments

GEX	GEX seurat object generated with VDJ_GEX_matrix
genes	Character vector. Genes of those in rownames(GEX) to plot. Can be any number, but more then 30 is discuraged because of cluttering
group.by	Character. Name of a column in GEX@meta.data to split the plot by. If set to \"none\", a plot with a single column will be produced.
threshold.to.plot	Integer 1-100. % of cells which must be expressing the feature to plot a point. If below, the field will be left empty
platypus.version	This is coded for \"v3\" only, but in practice any Seurat Object can be fed in

Value

Returns a ggplot object were the dot size indicates the percentage of expressing cells and the dot color indicates the expression level.

Examples

```
try({GEX_dottile_plot(GEX = Platypus::small_vgm[[2]], genes = c("CD19","CD83"),
group.by = "seurat_clusters", threshold.to.plot = 5)})
```

GEX_gene_visualization *Visualization of marker expression in a data set or of predefined genes (B cells, CD4 T cells and CD8 T cells).*

Description

Visualization of marker expression in a data set or of predefined genes (B cells, CD4 T cells and CD8 T cells).

Usage

```
GEX_gene_visualization(
  GEX,
  gene_set,
  predefined_genes = c("B_cell", "CD4_T_cell", "CD8_T_cell"),
  group.by
)
```

Arguments

GEX	GEX output of the VDJ_GEX_matrix function (VDJ_GEX_matrix[[2]])).
gene_set	Character vector containing the markers of interest given by the user.
predefined_genes	Character vector to chose between B_cell, CD4_T_cell, and CD8_T_cell.
group.by	Character. Column name of vgm to group plots by

Value

Return a list. Element[[1]] is the feature plot of markers of interest or predefined genes. Element[[2]] is the dottile plot of markers of interest or predefined genes. Element[[3]] is the violin plot of markers of interest or predefined genes.

Examples

```
GEX_gene_visualization(GEX = Platypus::small_vgm[[2]], predefined_genes = "B_cell")
```

GEX_GSEA *GEX Gene Set Enrichment Analysis and plotting*

Description

Conducts a Gene Set Enrichment Analysis (GSEA) on a set of genes submitted in a data frame with a metric each. Works with the output of GEX_genes_cluster or a custom data frame containing the gene symbols either in a column "symbols" or as rownames and a metric for each gene. The name of the column containing the metric has to be declared via the input metric.colname.

Usage

```
GEX_GSEA(
  GEX.cluster.genes.output,
  MT.Rb.filter,
  filter,
  path.to.pathways,
  metric.colname,
  pval.adj.cutoff,
  Enrichment.Plots,
  my.own.geneset,
  eps,
  platypus.version
)
```

Arguments

`GEX.cluster.genes.output`
: Data frame containing the list of gene symbols and a metric. Function works directly with GEX_cluster_genes output.

`MT.Rb.filter`
: Logical, should Mitotic and Ribosomal genes be filtered out of the geneset. True by default.

`filter`
: Character vector containing the identifying symbol sequence for the genes which should be filtered out, if MT.Rb.filter == TRUE. By default set to c("MT-", "RPL", "RPS").

`path.to.pathways`
: Either a path to gmt file containing the gene sets (can be downloaded from MSigDB) or vector where first element specifies species and second element specifies the MSigDB collection abbreviation. E.g.: c("Homo sapiens", "H"). Mouse C7 (immunologic signature) gene set will be used by default.

`metric.colname`
: Name of column which contains the metric used for the ranking of the submitted genelist. "avg_logFC" is used by default.

`pval.adj.cutoff`
: Only genes with a more significant adjusted pvalue are considered. Default: 0.001

`Enrichment.Plots`
: List of Gene-set names which should be plotted as Enrichment plots in addition to the top 10 Up and Downregulated Genesets.

`my.own.geneset`
: A list, where each element contains a gene list and is named with the corresponding pathway name. Default is set to FALSE, so that gene sets from MSigDB are used. Should not contain ".gmt" in name.

`eps`
: Numeric, specifying boundary for calculating the p value in the GSEA.

`platypus.version`
: Function works with V2 and V3, no need to set this parameter.

Value

Returns a list containing a tibble with the gene sets and their enrichment scores and Enrichment plots. List element [[1]]: Dataframe with Genesets and statistics. [[2]]: Enrichment plots of top10 Up regulated genesets. [[3]]: Enrichment plots of top10 Down regulated genesets. [[4]]: Enrichment plots of submited gene-sets in parameter Enrichment.Plot.

Examples

```
try({
df <- GEX_cluster_genes(Platypus::small_vgm[[2]])

output <- GEX_GSEA(GEX.cluster.genes.output =  df[[1]], MT.Rb.filter = TRUE
, path.to.pathways = "./c5.go.bp.v7.2.symbols.gmt")
cowplot::plot_grid(plotlist=output[[2]], ncol=2)

output <- GEX_GSEA(GEX.cluster.genes.output =  df[[1]], MT.Rb.filter = TRUE
, path.to.pathways = c("Mus musculus", "C7"))

output <- GEX_GSEA(GEX.cluster.genes.output =  df[[1]], MT.Rb.filter = TRUE
, my.own.geneset = my_geneset)
})
```

GEX_heatmap *Flexible GEX heatmap wrapper*

Description

Produces a heatmap containing gene expression information at the clonotype level. The rows correspond to different genes that can either be determined by pre-made sets of B or T cell markers, or can be customized by the user. The columns correspond to individual cells and the colors correspond to the different clonotype families.

Usage

```
GEX_heatmap(
  GEX,
  b.or.t,
  sample.index,
  clone.rank.threshold,
  custom.array,
  slot
)
```

Arguments

GEX	A single seurat object from clonotype_GEX function corresponding to all of the samples in a single VDJ_analyze object. This will likely be supplied as clonotype_GEX.output[[i]] if there were multiple, distinct transcriptomes.
b.or.t	Logical indicating if B or T cell gene panel should be used.
sample.index	Corresponds to which repertoire should be used in the case that the length of clonotype.list has a length greater than 1. The transcriptional profiles from only one repertoire can be plotted at a time.
clone.rank.threshold	A numeric that specifies the threshold clonal rank that specifies which clonotypes to extract transcriptome information from. For example, if 10 is supplied then the gene expression for the top ten clones included on the heatmap, separated by clonotype.
custom.array	Corresponds to which repertoire should be used in the case that the length of clonotype.list has a length greater than 1. The transcriptional profiles from only one repertoire can be plotted at a time.
slot	Seurat data slot from which to plot values. Can be "raw.data", "data" or "scale.data"

Value

Returns a heatmap via Seurat::DoHeatmap of gene expression per clonotype

See Also

VDJ_extract_sequences

Examples

```
small_vgm <- Platypus::small_vgm
small_vgm[[2]]$clone_rank <- c(1:nrow(small_vgm[[2]]@meta.data))
GEX_heatmap(GEX = small_vgm[[2]],b.or.t = "custom"
,clone.rank.threshold = 1,sample.index = "s1"
,custom.array = c("CD24A","CD83"), slot = "data")
```

GEX_lineage_trajectories *This is a function to infer single cell trajectories and identifying lineage structures on clustered cells. Using the slingshot library*

Description

This is a function to infer single cell trajectories and identifying lineage structures on clustered cells. Using the slingshot library

Usage

```
GEX_lineage_trajectories(GEX, grouping, cluster.num)
```

Arguments

GEX	GEX output of the VDJ_GEX_matrix function (VDJ_GEX_matrix[[2]]))
grouping	Determine by which identifier to group by. E.g. 'group_id' or default 'seurat_clusters' which are automatically generated in the clustering process.
cluster.num	A seurat cluster number for starting point of the lineage. Can be identified by using Seurat::DimPlot(VGM[[2]],group.by = "seurat_clusters"). Default is "0".

Value

Returns a list. Element [[1]] returns updated GEX object with the inferred pseudotime trajectories per lineage. [[2]] returns the UMAP with the grouped cells. [[3]] and [[4]] show the slingshot inferred trajectories in two different styles.

Examples

```
try({
lineage_trajectories <- GEX_lineage_trajectories(Platypus::small_vgm[[2]],
 grouping = 'group_id',
 cluster.num = "3")
})
```

GEX_pairwise_DEGs *Wrapper for calculating pairwise differentially expressed genes*

Description

Produces and saves a list of volcano plots with each showing differentially expressed genes between pairs groups. If e.g. seurat_clusters used as group.by, a plot will be generated for every pairwise comparison of clusters. For large numbers of this may take longer to run. Only available for platypus v3

Usage

```
GEX_pairwise_DEGs(
  GEX,
  group.by,
  min.pct,
  RP.MT.filter,
  label.n.top.genes,
  genes.to.label,
  save.plot,
  save.csv
)
```

Arguments

GEX	Output Seurat object of the VDJ_GEX_matrix function (VDJ_GEX_matrix.output[[2]])
group.by	Character. Defaults to "seurat_clusters" Column name of GEX@meta.data to use for pairwise comparisons. More than 20 groups are discuraged.
min.pct	Numeric. Defaults to 0.25 passed to Seurat::FindMarkers
RP.MT.filter	Boolean. Defaults to True. If True, mitochondrial and ribosomal genes are filtered out from the output of Seurat::FindMarkers
label.n.top.genes	Integer. Defaults to 50. Defines how many genes are labelled via geom_text_repel. Genes are ordered by adjusted p value and the first label.n.genes are labelled
genes.to.label	Character vector. Defaults to "none". Vector of gene names to plot independently of their p value. Can be used in combination with label.n.genes.
save.plot	Boolean. Defaults to False. Whether to save plots as appropriately named .png files
save.csv	Boolean. Defaults to False. Whether to save deg tables as appropriately named .csv files

Value

A nested list with out[[i]][[1]] being ggplot volcano plots and out[[i]][[2]] being source DEG dataframes.

Examples

```
GEX_pairwise_DEGs(GEX = Platypus::small_vgm[[2]],group.by = "sample_id"
,min.pct = 0.25,RP.MT.filter = TRUE,label.n.top.genes = 2,genes.to.label = c("CD24A")
,save.plot = FALSE, save.csv = FALSE)
```

GEX_phenotype *Assignment of cells to phenotypes based on selected markers*

Description

Adds a column to a VGM[[2]] Seurat object containing cell phenotype assignments. Defaults for T and B cells are available. Marker sets are customizable as below

Usage

```
GEX_phenotype(seurat.object, cell.state.names, cell.state.markers, default)
```

Arguments

seurat.object	A single seurat object / VDJ_GEX_matrix.output[[2]] object
cell.state.names	Character vector containing the cell state labels defined by the markers in cell.state.markers parameter. Example is c("NaiveCd4","MemoryCd4").
cell.state.markers	Character vector containing the gene names for each state. ; is used to use multiple markers within a single gene state. Different vector elements correspond to different states. Order must match cell.state.names containing the c("CD4+;CD44-","CD4+;IL7R+;CD44+").
default	Default is TRUE - will use predefined gene sets and cell states.

Value

Returns the input Seurat object with an additional column

Examples

```
vgm.phenotyped <- GEX_phenotype(seurat.object = Platypus::small_vgm[[2]]
, default = TRUE)
```

GEX_phenotype_per_clone

Plotting of GEX phenotype by VDJ clone

Description

Integrates VDJ and gene expression libraries by providing cluster membership seq_per_vdj object and the index of the cell in the Seurat RNA-seq object. ! For platypus.version == "v3" and VDJ_GEX_matrix output the function will iterate over entries in the sample_id column of the GEX by default.

Usage

```
GEX_phenotype_per_clone(
  GEX,
  clonotype.ids,
  global.clonotypes,
  GEX.group.by,
  GEX.clonotypes,
  platypus.version
)
```

Arguments

GEX	For platypus.version == "v3" the GEX object from the output of the VDJ_GEX_matrix function (VDJ_GEX_matrix.output \[\[2\]\]). For platypus.version == "v2" a single seurat object from automate_GEX function after labeling cell phenotypes using the GEX_phenotype function.
clonotype.ids	For platypus.version == "v2" Output from either VDJ_analyze or VDJ_clonotype functions. This list should correspond to a single GEX.list object, in which each list element in clonotype.list is found in the GEX.object. Furthermore, these repertoires should be found in the automate_GEX library.
global.clonotypes	Boolean. Defaults to FALSE. Set to True if clonotyping has been done across samples
GEX.group.by	For platypus.version == "v3". Character. Column name of the GEX@meta.data to group barplot by. Defaults to seurat_clusters
GEX.clonotypes	For platypus.version == "v3". Numeric vector with ids of clonotypes to plot e.g. c(1,2,3,4). Can also be set to "topclones"
platypus.version	Set to either "v2" or "v3" depending on wether suppyling GEX_automate or VDJ_GEX_matrix\[\[2\]\] objects. Defaults to "v3"

Value

Returns a stacked barplot that visualizes the seurat cluster membership for different cell phenotypes.

Examples

```
small_vgm_cl <- Platypus::small_vgm
small_vgm_cl[[2]]$clonotype_id_10x <- "clonotype1"
GEX_phenotype_per_clone(GEX = small_vgm_cl[[2]]
, GEX.clonotypes = c(1), GEX.group.by = "seurat_clusters", platypus.version = "v3")
```

GEX_proportions_barplot

Plots proportions of a group of cells within a secondary group of cells. E.g. The proportions of samples in seurat clusters, or the proportions of samples in defined cell subtypes

Description

Plots proportions of a group of cells within a secondary group of cells. E.g. The proportions of samples in seurat clusters, or the proportions of samples in defined cell subtypes

Usage

```
GEX_proportions_barplot(GEX, source.group, target.group, stacked.plot)
```

Arguments

GEX	GEX Seurat object generated with VDJ_GEX_matrix (VDJ_GEX_matrix.output[[2]])
source.group	Character. A column name of the GEX@meta.data with the group of which proportions should be plotted
target.group	Character. A column name of the GEX@meta.data with the group to calculate proportions within. If unsure, see examples for clarification
stacked.plot	Boolean. Defaults to FALSE. Whether to return a stacked barplot, with the y axis representing the % of cells of the target group. If set to FALSE a normal barplot (position = "dodge") will be returned with the y axis representing the % of cells of the source group

Value

Returns a ggplot barplot showing cell proportions by source and target group.

Examples

```
try({
GEX_proportions_barplot(GEX = Platypus::small_vgm[[2]], source.group = "sample_id"
, target.group = "seurat_clusters",stacked.plot = FALSE)
GEX_proportions_barplot(GEX = Platypus::small_vgm[[2]],
source.group = "seurat_clusters", target.group = "sample_id"
,stacked.plot = TRUE)
})
```

GEX_scatter_coexpression *Scatter plot for coexpression of two selected genes*

Description

Plots a composite figure showing single marker expression as histograms and coexpression as a scatterplot.

Usage

```
GEX_scatter_coexpression(GEX, gene.1, gene.2, color.theme)
```

Arguments

GEX	GEX seurat object generated with VDJ_GEX_matrix
gene.1	Character. Name of a gene in rownames(VDJ.matrix)
gene.2	Character. Name of a gene in rownames(VDJ.matrix)
color.theme	Character. A color to use for the composite plot

Value

Returns a gridplot showing coexpression scatterplot as well as histograms of gene.1 and gene.2

Examples

```
gene1 <- "CD24A"
gene2 <- "CD83"
GEX_scatter_coexpression(GEX = Platypus::small_vgm[[2]], gene1,gene2)
```

GEX_volcano *Flexible wrapper for GEX volcano plots*

Description

Plots a volcano plot from the output of the FindMarkers function from the Seurat package or the GEX_cluster_genes function alternatively.

Usage

```
GEX_volcano(
  DEGs.input,
  input.type,
  condition.1,
  condition.2,
  explicit.title,
  RP.MT.filter,
  color.p.threshold,
  color.log.threshold,
  label.p.threshold,
  label.logfc.threshold,
  n.label.up,
  n.label.down,
  by.logFC,
  maximum.overlaps,
  plot.adj.pvalue
)
```

Arguments

DEGs.input	Either output data frame from the FindMarkers function from the Seurat package or GEX_cluster_genes list output.
input.type	Character specifing the input type as either "findmarkers" or "cluster.genes". Defaults to "cluster.genes"
condition.1	either character or integer specifying ident.1 that was used in the FindMarkers function from the Seurat package. Should be left empty when using the GEX_cluster_genes output.
condition.2	either character or integer specifying ident.2 that was used in the FindMarkers function from the Seurat package. Should be left empty when using the GEX_cluster_genes output.
explicit.title	logical specifying whether the title should include logFC information for each condition.
RP.MT.filter	Boolean. Defaults to TRUE. Whether to exclude ribosomal and mitochondrial genes.
color.p.threshold	numeric specifying the adjusted p-value threshold for geom_points to be colored. Default is set to 0.01.
color.log.threshold	numeric specifying the absolute logFC threshold for geom_points to be colored. Default is set to 0.25.
label.p.threshold	numeric specifying the adjusted p-value threshold for genes to be labeled via geom_text_repel. Default is set to 0.001.
label.logfc.threshold	numeric specifying the absolute logFC threshold for genes to be labeled via geom_text_repel. Default is set to 0.75.

`n.label.up` numeric specifying the number of top upregulated genes to be labeled via geom_text_repel. Genes will be ordered by adjusted p-value. Overrides the "label.p.threshold" and "label.logfc.threshold" parameters.

`n.label.down` numeric specifying the number of top downregulated genes to be labeled via geom_text_repel. Genes will be ordered by adjusted p-value. Overrides the "label.p.threshold" and "label.logfc.threshold" parameters.

`by.logFC` logical. If set to TRUE n.label.up and n.label.down will label genes ordered by logFC instead of adjusted p-value.

`maximum.overlaps` integer specifying removal of labels with too many overlaps. Default is set to Inf.

`plot.adj.pvalue` logical specifying whether adjusted p-value should by plotted on the y-axis.

Value

Returns a volcano plot from the output of the FindMarkers function from the Seurat package, which is a ggplot object that can be modified or plotted. Infinite p-values are set defined value of the highest -log(p) + 100.

Examples

```
try({
GEX_volcano(findmarkers.output = FindMarkers.Output
, condition.1 = "cluster1", condition.2 = "cluster2"
, maximum.overlaps = 20)

GEX_volcano(findmarkers.output = FindMarkers.Output
, condition.1 = "cluster1", condition.2 = "cluster2"
, n.label.up = 50, n.label.down = 20)

GEX_volcano(findmarkers.output = GEX_cluster_genes.Output
, cluster.genes.output =TRUE)
})
```

PlatypusDB_AIRR_to_VGM

AIRR to Platypus V3 VGM compatibility function

Description

Loads in and converts input AIRR-compatible tsv file(s) into the Platypus VGM object format.All compulsory AIRR data columns are needed. Additionally, the following columns are required: v_call, cell_id, clone_id. If trim.and.align is set to TRUE additionally the following columns are needed: v_sequence_start, j_sequence_end. Note on TRUST4 input: TRUST4 (https://doi.org/10.1038/s41592-021-01142-n2) is a newly alignment tool for VDJ data by the Shirley lab. It is able to also extract

VDJ sequences from 10x GEX data. We are actively testing TRUST4 as an alternative to Cellranger and can not give recommendations as of now. This function does support the conversion of TRUST4 airr output data into the Platypus VGM format. In that case, an extra column will be added describing whether the full length VDJ sequence was extracted for any given cell and chain.

Usage

```
PlatypusDB_AIRR_to_VGM(
  AIRR.input,
  get.VDJ.stats,
  VDJ.combine,
  trim.and.align,
  filter.overlapping.barcodes.VDJ,
  group.id,
  verbose
)
```

Arguments

AIRR.input	Source of the AIRR table(s) as a list. There are 2 available input options: 1. 1. List with local paths to .tsv files / 3. List of AIRR tables loaded in as R objects within the current R environment.
get.VDJ.stats	Boolean. Defaults to TRUE. Whether to generate summary statistics on repertoires and output those as output_VGM[[3]]
VDJ.combine	Boolean. Defaults to TRUE. Whether to integrate repertoires. A sample identifier will be appended to each barcode both. Highy recommended for all later functions
trim.and.align	Boolean. defaults to FALSE. Whether to trim VJ/VDJ seqs and add information from alignment in AIRR dataframe columns. ! No alignment is done here, instead, columns containing alignment information in the AIRR dataframes are reformatted.
filter.overlapping.barcodes.VDJ	Boolean. defaults to TRUE. Whether to remove barcodes which are shared among samples in the GEX analysis. Shared barcodes normally appear at a very low rate.
group.id	vector with integers specifying the group membership. c(1,1,2,2) would specify the first two elements of the input AIRR list are in group 1 and the third/fourth input elements will be in group 2.
verbose	Writes runtime status to console. Defaults to FALSE

Value

A VDJ_GEX_Matrix object used in Platypus V3 as an input to most analysis and plotting functions

Examples

```
try({
VGM <- PlatypusDB_AIRR_to_VGM(AIRR.input =
```

```
list("~/pathto/s1/airr_rearrangement.tsv", "~pathto/s2/airr_rearrangement.tsv"),
VDJ.combine = TRUE, group.id = c(1,2), filter.overlapping.barcodes.VDJ = TRUE)
})
```

PlatypusDB_load_from_disk

PlatypusDB utility for import of local datasets

Description

Utility function for loading in local dataset as VDJ_GEX_matrix and PlatypusDB compatible R objects. Especially useful when wanting to integrate local and public datasets. This function only imports and does not make changes to format, row and column names. Exception: filtered_contig.fasta are appended to the filtered_contig_annotations.csv as a column for easy access

Usage

```
PlatypusDB_load_from_disk(
  VDJ.out.directory.list,
  GEX.out.directory.list,
  FB.out.directory.list,
  batches
)
```

Arguments

`VDJ.out.directory.list`
: List containing paths to VDJ output directories from cell ranger. This pipeline assumes that the output file names have not been changed from the default 10x settings in the /outs/ folder. This is compatible with B and T cell repertoires (both separately and simultaneously).

`GEX.out.directory.list`
: List containing paths the outs/ directory of each sample or directly the raw or filtered_feature_bc_matrix folder. Order of list items must be the same as for VDJ. This outs directory may also contain Feature Barcode (FB) information. Do not specify FB.out.directory in this case.

`FB.out.directory.list`
: List of paths pointing at the outs/ directory of output of the Cellranger counts function which contain Feature barcode counts. Any input will overwrite potential FB data loaded from the GEX input directories. Length must match VDJ and GEX directory inputs. (in case of a single FB output directory for multiple samples, please specifiy this directory as many times as needed)

`batches`
: Integer vector. Defaults to all 1, yielding all samples with batch number "b1". Give a batch number to each sample (each entry in the VDJ/GEX input lists). This will be saved as element 5 in the sample list output.

Value

Large nested list object containing all needed Cellranger outputs to run the VDJ_GEX_matrix function. Level 1 of the list are samples, level 2 are VDJ GEX and metadata information. (e.g. out[[1]][[1]] corresponds to VDJ data objects of sample 1)

Examples

```
try({
VDJ.in <- list()
VDJ.in[[1]] <- c("~/VDJ/S1/")
VDJ.in[[2]] <- c("~/VDJ/S2/")
GEX.in <- list()
GEX.in[[1]] <- c("~/GEX/S1/")
GEX.in[[2]] <- c("~/GEX/S2/")
PlatypusDB_load_from_disk(VDJ.out.directory.list = VDJ.in, GEX.out.directory.list = GEX.in)
})
```

PlatypusDB_VGM_to_AIRR

Platypus V3 VGM to AIRR compatibility function

Description

Exports AIRR compatible tables supplemented with VDJ and GEX information from the Platypus VGM object and the cellranger output airr_rearrangements.tsv

Usage

```
PlatypusDB_VGM_to_AIRR(
  VGM,
  VDJ.features.to.append,
  GEX.features.to.append,
  airr.rearrangements,
  airr.integrate
)
```

Arguments

VGM	Output object of the VDJ_GEX_matrix function generated with VDJ.combine = T, GEX.combine = T (to merge all samples) and integrate.VDJ.to.GEX = T (to integrate VDJ and GEX data)
VDJ.features.to.append	Character vector. Defaults to "none". Can be either "all" or column names of the VGM VDJ matrix (VGM[[1]]) to append to the AIRR compatible table.

GEX.features.to.append
: Character vector. Defaults to "none". Can be either "all" or GEX metadata column names or Gene names of the VGM GEX object (VGM[[2]])(passed to Seurat::FetchData()) to append to the AIRR compatible table. For a list of available features run: names(VGM[[2]]@meta.data) and rownames(VGM[[2]])

airr.rearrangements
: Source of the airr_rearrangements.tsv file as generated by Cellranger. There are 3 available input options: 1. R list object from Platypus_DB_load_from_disk or Platypus_DB_fetch / 2. List with local paths to airr_rearrangements.tsv / 3. List of airr_rearrangements.tsv loaded in as R objects within the current R enviroment. ! Order of input list must be identical to that of sample_ids in the VGM ! If not provided or set to "none" CIGAR strings in output will be empty.

airr.integrate
: Boolean. Defaults to TRUE, whether to integrate output AIRR tables

Value

A list of length of samples in VGM containing a AIRR-compatible dataframe for each sample if airr.integrate = F or a single dataframe if airr.integrate = T ! Cave the format: VGM object => 1 cell = 1 row; AIRR table 1 cell = as many rows as VDJ and VJ chains available for that cell. GEX cell-level information is attached to all rows containing a chain of that cell.

Examples

```
try({
airr.list.out <- PlatypusDB_VGM_to_AIRR(VGM = VGM
, VDJ.features.to.append = c("VDJ_cdr3s_aa")
, GEX.features.to.append = c("CTLA4", "TOX"), airr.rearrangements = Data.in)

airr.list.out <- PlatypusDB_VGM_to_AIRR(VGM = VGM
, VDJ.features.to.append = c("VDJ_cdr3s_aa")
, GEX.features.to.append = c("CTLA4", "TOX"),
airr.rearrangements =list("~/path_to/s1/airr.rearrangement.tsv"
,"~/path_to/s2/airr_rearrangement.tsv"))

airr.list.out <- PlatypusDB_VGM_to_AIRR(VGM = VGM
, VDJ.features.to.append = c("VDJ_cdr3s_aa")
, GEX.features.to.append = c("CTLA4", "TOX"),
airr.rearrangements = list(airr_rearrangements.s1, airr_rearrangements_2))

VDJ.out.directory.list <- list()
VDJ.out.directory.list[[1]] <- c("~/cellrangerVDJ/s1")
VDJ.out.directory.list[[2]] <- c("~/cellrangerVDJ/s2")

GEX.out.directory.list <- list()
GEX.out.directory.list[[1]] <- c("~/cellrangerGEX/s1")
GEX.out.directory.list[[2]] <- c("~/cellrangerGEX/s2")
VGM <- VDJ_GEX_matrix(VDJ.out.directory.list = VDJ.out.directory.list,
GEX.out.directory.list = GEX.out.directory.list,
GEX.integrate = TRUE, VDJ.combine = TRUE, integrate.GEX.to.VDJ = TRUE
, integrate.VDJ.to.GEX = TRUE,
get.VDJ.stats = FALSE, trim.and.align = FALSE)
```

```
airr.list.out <- PlatypusDB_VGM_to_AIRR(VGM = VGM,
VDJ.features.to.append = c("VDJ_sequence_nt_trimmed","VJ_sequence_nt_trimmed"),
GEX.features.to.append = c("UMAP_1","UMAP_2","CTLA4", "TOX"),
airr.rearrangements = c("~/cellrangerVDJ/s1/airr_rearrangement.tsv"
,"~/cellrangerVDJ/s2/airr_rearrangement.tsv"))
})
```

`small_vdj` *Small VDJ dataframe for function testing purposes*

Description

Small VDJ dataframe for function testing purposes

Usage

```
small_vdj
```

Format

An object of class `data.frame` with 3671 rows and 70 columns.

References

R package Platypus : https://doi.org/10.1093/nargab/lqab023

`small_vgm` *Small VDJ GEX matrix (VGM) for function testing purposes*

Description

Small VDJ GEX matrix (VGM) for function testing purposes

Usage

```
small_vgm
```

Format

An object of class `list` of length 5.

References

R package Platypus : https://doi.org/10.1093/nargab/lqab023

VDJ_abundances *Calculate abundances/counts of specific features for a VDJ dataframe*

Description

Calculate the absolute counts or proportions of a specific cell-level feature (column in the VDJ/VDJ.GEX.matrix[[1]] object), per an optional specific grouping factor (e.g., clonotype via 'clonotype_id') and an optional sample factor(e.g., 'sample_id'). Outputs either a count dataframe of the specific feature or a ggplot2 barplot.

Usage

```
VDJ_abundances(
  VDJ,
  feature.columns,
  proportions,
  specific.features,
  grouping.column,
  max.groups,
  specific.groups,
  sample.column,
  VDJ.VJ.1chain,
  treat.incomplete.groups,
  treat.incomplete.features,
  combine.features,
  treat.combined.features,
  treat.combined.groups,
  specific.feature.colors,
  output.format
)
```

Arguments

VDJ	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_build or VDJ_GEX_matrix function in Platypus.
feature.columns	vector of strings, denoting the columns of the VDJ/VDJ.GEX.matrix[[1]] object from which to extract the unique feature values (for which we will calculate the counts or proportions).
proportions	string, 'absolute' will return the absolute counts, 'group.level.proportions' will return the counts divided by the total number or elements/values in the specific groups (group level proportions), 'sample.level.proportions' will return the counts divided by the total number of elements in the sample.
specific.features	vector of specific feature values (or NULL) for which to calculate counts/proportions, from the specified feature.columns parameter (only works if a single feature column is specified in feature.columns).

`grouping.column`
: string, vector of strings, or 'none' - represents the column from the VDJ/VDJ.GEX.matrix[[1]] object by which to group counting process. This is usually the 'clonotype_id' column to calculate frequencies at the clonotype level. If 'none', no grouping will be done. To group by multiple columns, input the specific columns as a vector of strings. For example, if feature.columns='VDJ_cgene' and grouping.column='clonotype_id', we will obtain a count dataframe of the frequencies of each isotype per unique clonotype (per sample if sample.column='sample_id').

`max.groups`
: integer or NULL, the maximum number of groups for which to count features. If NULL, it will count for all groups.

`specific.groups`
: vector of strings (or 'none'), if the counting should be done only for specific groups (e.g., count the frequency of isotype only for clonotypes 1 and 2 if feature.columns='VDJ_cgene', grouping.column='clonotype_id' and specific.groups=c('clonotype1', 'clonotype2'))

`sample.column`
: string, represents the sample column if your VDJ/VDJ.GEX.matrix[[1]] object has multiple samples (usually 'sample_id')

`VDJ.VJ.1chain`
: boolean, if T will remove aberrant cells (more than 1 VDJ of VJ chain), if F it will keep them.

`treat.incomplete.groups`
: string, method of dealing with groups which are missing the features in the feature.columns parameter (e.g., a clonotype which does not have any transcriptomic clusters annotations if feature.columns='transcript_cluster').'exclude' - excludes groups with no cells for the specific features, 'unknown' - sets them as unknown

`treat.incomplete.features`
: string, method of dealing with missing feature values (e.g., a clonotype has several NA values for the 'VDJ_cgene' feature.column - cells with NA values). 'unknown' - counted as unknown, 'exclude' - excludes completely, 'max.global' - replaces value by max value of that feature across the repertoire, 'max.group' - replaced by the max feature value inside that group, 'proportional' - iteratively assigns the missing values to the known groups, keeping the same proportions.

`combine.features`
: boolean - if T and we have two columns in feature.columns, will combine the feature values for each cell in the VDJ object, counting them as a single feature when calculating proportions.

`treat.combined.features`
: string, method of dealing with combined features with missing values. 'exclude' will be treated similarly to excluding incomplete feature values (excluding them completely if a single value is missing from the combination), or 'include' and will be treated as a new feature value.

`treat.combined.groups`
: string, method of dealing with combined groups with missing values, in case the grouping.column parameter is a vector of strings. 'exclude' will exclude the combined group altogether if a group value is missing/NA. 'include' will include such groups in the analysis.

`specific.feature.colors`
: named list of specific colors to be used in the final barplots, for each unique feature value in the VDJ object's feature.columns values. For example, if we have a feature column of binders with unique values=c('yes', 'no'), specific.feature.colors=list('yes'='blue', 'no'='red') will color them accordingly.

`output.format`
: string, either 'plots' to obtain barplots, 'abundance.df' to obtain the count dataframe, or 'abundance.df.list' to obtain a list of count dataframes, for each sample.

Value

Either a count dataframe with the following columns: group(=unique group value, e.g., 'clonotype1' if grouping.column='clonotype_id'), sample, group_frequency, unique_feature_values, feature_value_counts, total_feature_names or a barplot of the counts/proportions per feature, per group.

Examples

```
VDJ_abundances(VDJ = Platypus::small_vdj,
feature.columns='VDJ_cgene', proportions='absolute',
grouping.column='clonotype_id', specific.groups='none',
output.format='plot')
```

VDJ_alpha_beta_Vgene_circos

Produces a Circos plot from the VDJ_analyze output. Connects the V-alpha with the corresponding V-beta gene for each clonotype.

Description

Produces a Circos plot from the VDJ_analyze output. Connects the V-alpha with the corresponding V-beta gene for each clonotype.

Usage

```
VDJ_alpha_beta_Vgene_circos(
  VGM,
  V.or.J,
  B.or.Tcells,
  label.threshold,
  c.threshold,
  cell.level,
  clonotype.per.gene.threshold,
  c.count.label,
  c.count.label.size,
  platypus.version,
  filter1H1L,
  gene.label,
  gene.label.size,
```

```
  arr.col,
  arr.direction,
  topX,
  platy.theme,
  clonotype.column
)
```

Arguments

VGM	The output of the VDJ_GEX_matrix function (VDJ_GEX_matrix.output[[1]]) has to be supplied. For Platypus v2: The output of the VDJ_GEX_integrate function (Platypus platypus.version v2). A list of data frames for each sample containing the clonotype information and cluster membership information.
V.or.J	Determines whether to plot the alpha beta gene pairing of the V or J genes. "V", "J" or "both" as possible inputs. Default: "both".
B.or.Tcells	Specify whether B or T cells are being analyzed ("B" or "T"). If not specified, function attempts to decide based on gene names.
label.threshold	Genes are only labeled if the count is larger then the label.threshold. By default all label.threshold = 0 (all genes are labeled).
c.threshold	Only clonotypes are considered with a frequency higher then c.threshold. Allows to filter for only highly expanded clonotypes.
cell.level	Logical, defines whether weight of connection should be based on number of clonotypes of number of cells. Default: number of clonotypes.
clonotype.per.gene.threshold	How many clonotypes are required to plot a sector for a gene. Filters the rows and colums of the final adjacency matrix.
c.count.label	Boolean, lets the user decide if the gene and count labels should be plotted or not. Default = T.
c.count.label.size	Determines the font size of the gene labels. By default the font size for count labels is 0.6.
platypus.version	Which platypus.version of platypus is being used. Default = v3. Set to v3 if VDJ_GEX_matrix.output[[1]] is used
filter1H1L	Whether to filter the input VGM in "v3" to only include cells with 1 VDJ and 1 VJ chain. Defaults to TRUE
gene.label	Boolean, lets the user decide if the gene labels should be plotted or not.
gene.label.size	Determines the font size of the gene labels. By default the labelsize is automatically adjusted to 0.7 for labels with two or less digits, 0.6 for labels between 2 and 6 digits, and 0.4 for all longer labels. A manually defined font size will be the same for all labels!
arr.col	Data.frame with three columns where the first two indicate the names of genes, clonotypes or clusters to be connected, and the third corresponds to the color of the arrow. Default set to data.frame(c("dummy.clonotype"), c("dummy.cluster"), c("dummy.color")), so no arrow is drawn.

arr.direction	Either 1 or -1 and determines the direction of the arrow. Default=1.
topX	Filters for the top X clonotypes and only plots the respective gene combinations or cluster memberships.
platy.theme	Allows plotting in the new "pretty" theme or the older "spiky" theme without group labels and radial arrangement of gene.labels. Default = "pretty".
clonotype.column	Which column in VGM contains the clonotyping information? Default="clonotype_id_10X".

Value

Returns a circos plot and a list object with the following elememts for N samples: [[1 to N]] The first N listelements corresponds to the recorded circos plots for N beeing the number or samples in the VGM. Since Circlize uses the R base plotting funciton, this is not a ggplot object but can still be replotted by calling the first list element. [[N+1]] Adjacency matrix forwarded to VDJ_circos(). This Matrix contains the counts and can be used for manual replotting using VDJ_circos directly. [[N+2]] Contains a named list with colors for each connection drawn and can be used for manual replotting using VDJ_circos directly. [[N+3]] Contains a named list with grouping information and can be used for manual replotting using VDJ_circos directly.

Examples

```
alpha_beta_VJgene <- VDJ_alpha_beta_Vgene_circos(Platypus::small_vgm[[1]])
alpha_beta_VJgene[[1]]
```

VDJ_antigen_integrate *Integrates antigen-specific information into the VDJ/VDJ.GEX.matrix[[1]] object*

Description

Integrate antigen-specific information from a list of antigen dataframes or antigen csv file paths. The antigen data should contain either the clonotypes, cell barcodes, or sequences with the specific column names of the VDJ/VDJ.GEX.matrix[[1]] object. These columns will be used to rematch the binder information at the cell, sequence, or clonotype level into the main VDJ.GEX.matrix[[1]].

Usage

```
VDJ_antigen_integrate(
  VDJ,
  antigen.data.list,
  antigen.features,
  binder.threshold,
  VDJ.VJ.1chain,
  match.by,
  matching.type,
```

```
  distance.threshold,
  cores,
  sample.id,
  aberrant.chosen.sequences,
  output.format
)
```

Arguments

VDJ	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
antigen.data.list	list of antigen csv file paths or antigen dataframes for the specific antigen datasets. To ease matching, the column names by which we will match should be the same as the column names in the original VDJ/VDJ.GEX.matrix[[1]] object.
antigen.features	vector of columns of antigen features to be integrated from the antigen csv files into the VDJ/VDJ.GEX.matrix[[1]] object. The vector can also use unique, short-hand names of the columns to add (e.g., 'affinity' for 'octet.affinity.[nM]').
binder.threshold	list or nested list of threshold values and specific features by which to define binders in the VDJ. For example, if binder.threshold=list(list('affinity', 0.2), list('elisa', 0.8)), we will have two new binder columns: binders_affinity if the values are greater than 0.2, binders_elisa if they are greater than 0.8.
VDJ.VJ.1chain	boolean, if T will remove aberrant cells (more than 1 VDJ of VJ chain), if F it will keep them in the VDJ when matching antigen data.
match.by	string, represents the method by which to match the antigen data and integrate it into the VDJ/VDJ.GEX.matrix[[1]] object. 'clonotype' will match by 'clonotype_id' (needs to be present in the antigen data), 'clonotype.v3' will match by v3 cellranger clonotypes (you need a v3_clonotypes column in the VDJ/VDJ.GEX.matrix[[1]], 'cdr3.aa' by VDJ and VJ cdr3s amino acid sequences, 'cdrh3.aa' by VDJ cdr3s amino acid sequences, 'VDJ.VJ.aa' by full VDJ and VJ aa sequences, 'VDJ.VJ.nt' by trimmed nt VDJ and VJ sequences (must run VDJ_call_MIXCR first on the VDJ),'cdr3.nt' by VDJ and VJ cdr3s as nucleotides, 'cdrh3.nt.' by VDJ cdr3s as nucleotides, 'absolut' will match the VDJ_cdr3s_aa with the CDR3 column in Absolut! datasets.
matching.type	string, either 'exact' for exact sequence matching if the match.by parameter is a sequence type, or 'homology' for homology matching (matches if the Levehnstein distance is less than the distance.threshold parameter).
distance.threshold	integer, maximum string distance value by which to match sequences in the antigen data and sequences in the VDJ object (to further integrate the antigen data).
cores	Number of cores to use for parallel computations. Defaults to number of available cores. Setting this parameter is good practice on clusters.
sample.id	boolean, if T then will also match by the 'sample_id' column in the antigen dataframes.

`aberrant.chosen.sequences`
: boolean, if T will add a column of the chosen aberrant sequences (which matched a sequence in the antigen data) if matching by sequence (and VDJ.VJ.1chain=F).

`output.format`
: string, 'vgm' - returns the full VDJ object, 'dataframe.per.sample' - list of VDJ dataframes for each sample.

Value

Either the original VDJ dataframe with additional columns of the antigen features integrated, a list of VDJ dataframes per sample.

Examples

```
try({
VDJ_antigen_integrate(VDJ,antigen.directory.list=antigen.directory.list,
antigen.feature=c('elisa', 'affinity'),VDJ.VJ.1chain=T,
match.by='clonotype',sample.id=T, output.format='vgm')
})
```

VDJ_assemble_for_PnP *Ab sequence assembly for recombinant PnP expression*

Description

Assembles sequences from MIXCR output into inserts for expression in PnP cells. For detailes check https://doi.org/10.1038/ncomms12535 ! ALWAYS VALIDATE INDIVIDUAL SEQUENCE IN GENEIOUS OR OTHER SOFTWARE BEFORE ORDERING SEQUENCES FOR EXPRESSION ! Check notes on column content below ! Only cells with 1 VDJ and 1 VJ sequence are considered. Warnings are issued if sequences do not pass necessary checks

Usage

```
VDJ_assemble_for_PnP(
  VDJ.mixcr.matrix,
  id.column,
  species,
  manual_IgKC,
  manual_2A,
  manual_VDJLeader,
  write.to.disk,
  filename,
  verbose
)
```

Arguments

VDJ.mixcr.matrix	Output dataframe from the VDJ_call_MIXCR function or a dataframe generated using the VDJ_GEX_matrix function and supplemented with MIXCR information (Needed columns: All Framework and CDR sequences)
id.column	Character. Column name of VDJ.mixcr.matrix to use as ID for the assembled sequences. Defaults to "barcode"
species	Character. Which IgKC sequence to use. Can be "human" or "mouse". Defaults to "mouse"
manual_IgKC	Character. Manual overwrite for sequence used as IgKC.
manual_2A	Character. Manual overwrite for sequence used as Furine 2A site.
manual_VDJLeader	Character. Manual overwrite for sequence used as VDJ Leader and signal peptide.
write.to.disk	Boolean. Defaults to TRUE. Whether to save assembled sequences to working directory
filename	Character. Output file name for .fasta and .csv files if write.to.disk == TRUE. Defaults to PnP_assembled_seqs.fasta/.csv
verbose	Print runtime message to console. Defaults to FALSE

Value

Returns the input VGM matrix with one additional column containing the assembles sequences. If write.to.disk == TRUE writes a CSV containing key columns of the VGM as well as a .FASTA file to the current working director (getwd()) ! Important notes on column content: 1. The column "seq_length_check" contains either "passed" or "FAILED". If FAILED, this means that at least one of the sequences (e.g. FRL1) was shorter than 9NTs and therefore considered invalid. Please check for missing sequences if you find any warnings 2. The column "seq_codon_check" is deemed "passed" if all CDR and FR input sequences of a cell contain only full codons (i.e. are divisible by 3) 3. The column "PnP_assembled_seqs" contains the assembled sequences / inserts for PnP expression. These should be validated manually in Geneious or other software and can then be ordered to be synthesized. 4. The column "PnP_assembled_annotations" contains a string of annotations for the respective assembled sequence. The structure is | [Sequence element] -> [index (starting from 1) of last nucleotide of the sequence element] ... 5. The column "PnP_assembled_translations" contains the amino acid translation of the full contig that will result from the assembled insert in the backbone PnP vector. Please note: the sequences in the PnP_assembled_translation resulted from pasting the VJ leader sequence (contained in the PnP vector backbone), the PnP_assembled_seqs (The insert itself) and a surrogate stop codon ATAA. If correct, the translation should only contain one * (stop codon) at the very end. For reference: VJLeader sequence: ATGGATTTTCAGGTGCAGATTTTCAGCTTCCTGCTAATCAGCGCTTCAGTTATAATGTCCCGGGGG 6. The column "seq_VJCDR3_check" is deemed "passed" if the translated sequence of the input VJ CDR3 is found in the translated assembled sequence. If this test fails, there is likely an issue with the VJ segment 7. The column "seq_Fur2A_check" is deemed "passed" if correct AA sequence of the 2A site is found in the translated assembled sequence. If this test fails, and the seq_VJCDR3_test was passed, there is likely an issue at the border between VJ and IgKC/2A sequences 8. The column "seq_VDJCDR3_check" is deemed "passed" if the translated sequence of the input VDJ CDR3

is found in the translated assembled sequence. 9. The column "seq_splicesite_check" is deemed passed if the last 6 nucleotides of the assembled sequence are one of the following: "TCCTCA", "TCTTCA","TCGTCA","TCATCA".

Examples

```
try({
VGM_with_PnP_seq <- VDJ_assemble_for_PnP(VDJ.mixcr.matrix = VDJ_call_MIXCR.output
, id.column = "barcode",species = "mouse", manual_IgKC = "none", manual_2A = "none"
, manual_VDJLeader = "none", write.to.disk = FALSE, filename = "PnP_seq_example")
})
```

VDJ_build *Minimal Version of the VDJ Building Part from VDJ_GEX_matrix() Function*

Description

This function imports Cell Ranger output into an R dataframe for downstream analyses. It is a minimal version of the VDJ building part from the 'VDJ_GEX_matrix()' function of the Platypus package, adapted for Cell Ranger v7 and older versions. Seurat objects can be integrated by matching barcodes from the Seurat object's metadata with the barcodes in the 'barcode' column of the VDJ dataframe.

Usage

```
VDJ_build(
  VDJ.directory,
  VDJ.sample.list,
  remove.divergent.cells,
  complete.cells.only,
  trim.germlines,
  gap.opening.cost,
  parallel,
  num.cores
)
```

Arguments

`VDJ.directory` A string specifying the path to the parent directory containing the output folders (one folder for each sample) of Cell Ranger. This pipeline assumes that the output file names have not been changed from the default 10x settings in the '/outs/' folder. This is compatible with B and T cell repertoires. The following 5 files are necessary within this folder:

'filtered_contig_annotations.csv' Contains the filtered contig annotations.

'filtered_contig.fasta' Contains the filtered contig sequences in FASTA format.

'consensus_annotations.csv' Contains the consensus annotations.

'consensus.fasta' Contains the consensus sequences in FASTA format.

'concat_ref.fasta' Contains concatenated reference sequences.

`VDJ.sample.list`
: A list specifying the paths to the output folders (one folder for each sample) of Cell Ranger. This pipeline assumes that the output file names have not been changed from the default 10x settings in the '/outs/' folder and requires the same 5 files listed above.

`remove.divergent.cells`
: A logical value ('TRUE'/'FALSE'). If 'TRUE', cells with more than one VDJ transcript or more than one VJ transcript will be excluded. This could be due to multiple cells being trapped in one droplet or light chain dual expression (concerns ~2-5 percent of B cells, see DOI:10.1084/jem.181.3.1245). Defaults to 'FALSE'.

`complete.cells.only`
: A logical value ('TRUE'/'FALSE'). If 'TRUE', only cells with both a VDJ transcript and a VJ transcript are included in the VDJ dataframe. Keeping only cells with 1 VDJ and 1 VJ transcript could be preferable for downstream analysis. Defaults to 'FALSE'.

`trim.germlines`
: A logical value ('TRUE'/'FALSE'). If 'TRUE', the raw germline sequences of each clone will be trimmed using the consensus sequences of that clone as reference sequences (using 'Biostrings::pairwiseAlignment' with the option "global-local" and a gap opening cost specified by 'gap.opening.cost'). Defaults to 'FALSE'.

`gap.opening.cost`
: A numeric value representing the cost for opening a gap in 'Biostrings::pairwiseAlignment' when aligning and trimming germline sequences. Defaults to 10.

`parallel`
: A logical value ('TRUE'/'FALSE'). If 'TRUE', the per-sample VDJ building is executed in parallel (parallelized across samples). Defaults to 'FALSE'.

`num.cores`
: An integer specifying the number of cores to be used when 'parallel = TRUE'. Defaults to all available cores minus 1 or the number of sample folders in 'VDJ.directory' (whichever is smaller).

Details

The function extracts and processes VDJ data from Cell Ranger output folders, making it suitable for integration with downstream analysis workflows such as Seurat. It can handle both T and B cell repertoires and is optimized for Cell Ranger v7.

Value

A dataframe representing the VDJ / VGM[[1]] object. Each row in this dataframe represents one cell or one unique cell barcode.

Examples

```
try({
  VDJ <- VDJ_build(
```

```
    VDJ.directory = "path/to/VDJ_directory",
    remove.divergent.cells = TRUE,
    complete.cells.only = TRUE,
    trim.germlines = TRUE
  )
})
```

`VDJ_call_enclone` *(Re)clonotype a VDJ object using cellranger's enclone tool*

Description

Calls recon to clonotype a VDJ object given a VDJ.directory (with sample folders which should include the all_contig_annotations.json file) - outputs a new VDJ with updated clonotype_id, clonotype_id_10x, and clonotype_frequency columns

Usage

```
VDJ_call_enclone(
  VDJ,
  VDJ.directory,
  global.clonotype,
  samples.to.clonotype,
  samples.to.combine,
  same.origin,
  output.format,
  operating.system,
  parallel
)
```

Arguments

`VDJ`	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
`VDJ.directory`	string - directory for the VDJ data, should be the main folder which includes the individual sample folders (each with the all_contig_annotations.json file that is used by enclone)
`global.clonotype`	bool - if T, will use clonotype definitions irrespective of samples. Must also be T is you wish to merge clonotypes from two specific (which should be specified in the samples.to.combine parameter)
`samples.to.clonotype`	- vector - lists the samples names which should be clonotyped. The unspecified samples will keep their old clonotype defintions.

samples.to.combine
: - vector or list of vectors - lists the samples which you wish to have their clonotypes merged (e.g., c('s1','s2') to only merge the first 2 samples, or list(c('s1','s3'), c('s2', 's4')) to merge the first and third, second and fourth, respectively). global.clonotype must be set to T!

same.origin
: bool - if the merged samples come from the same donor, with the same or with different origins. If two datasets come from the same origin, enclone will filter to remove certain artifacts.

output.format
: string - 'vgm' to output a VGM-specific VDJ dataframe (all samples in the same dataframe).

operating.system
: string - operating system on which enclone will be run. 'Windows' for Windows, 'Linux' for Linux, 'Darwin' for MacOS.

parallel
: bool - if T, the program will be executed in parallel, on no. cores = max. available cores - 1.

Value

Reclonotyped VDJ object using the enclone software and 10x-specific clonotype deifinition.

Examples

```
try({
VDJ_call_enclone(vdj, VDJ.directory, samples.to.combine = c('s1', 's2', 's3'), global.clonotype = T)
})
```

VDJ_circos

Plots a Circos diagram from an adjacency matrix. Uses the Circlize chordDiagram function. Is called by VDJ_clonotype_clusters_circos(), VDJ_alpha_beta_Vgene_circos() and VDJ_VJ_usage_circos() functions or works on its own when supplied with an adjacency matrix.

Description

Plots a Circos diagram from an adjacency matrix. Uses the Circlize chordDiagram function. Is called by VDJ_clonotype_clusters_circos(), VDJ_alpha_beta_Vgene_circos() and VDJ_VJ_usage_circos() functions or works on its own when supplied with an adjacency matrix.

Usage

```
VDJ_circos(
  Adj_matrix,
  platy.theme,
  group,
  grid.col,
```

```
  label.threshold,
  axis,
  c.count.label,
  arr.col,
  arr.direction,
  gene.label.size,
  gene.label,
  c.count.label.size
)
```

Arguments

Adj_matrix	Adjacency matrix to be plotted. Rownames and Colnames correspond to genes to be matched and entries determine the weight of the connection between the genes (eg. number of clonotypes expressing these two genes).
platy.theme	Allows plotting in the new "pretty" theme or the older "spiky" theme without group labels and radial arrangement of gene.labels. Default = "pretty".
group	Named list of genes, with list elements corresponding to group-names, and element names being the gene-names. Is generated by VDJ_VJ_usage and VDJ_alpha_beta_Vgene_circos.
grid.col	Named list of genes, with list elements corresponding to color and element names being gene-names. If not supplied it is generated randomly within the function. Is also generated by VDJ_VJ_usage and VDJ_alpha_beta_Vgene_circos.
label.threshold	Genes are only labeled if the count is larger then the label.threshold. By default all label.threshold = 0 (all genes are labeled).
axis	Option to choose the count axis for each gene. "default", "percent" or "max" possible. Default: "max".
c.count.label	Boolean, lets the user decide if the gene and count labels should be plotted or not. Default = T.
arr.col	Data.frame with three columns where the first two indicate the names of genes, clonotypes or clusters to be connected, and the third corresponds to the color of the arrow. Default set to data.frame(c("dummy.clonotype"), c("dummy.cluster"), c("dummy.color")), so no arrow is drawn.
arr.direction	Either 1 or -1 and determines the direction of the arrow. Default=1.
gene.label.size	Determines the font size of the gene labels. By default the labelsize is automatically adjusted to 0.7 for labels with two or less digits, 0.6 for labels between 2 and 6 digits, and 0.4 for all longer labels. A manually defined font size will be the same for all labels!
gene.label	Boolean, lets the user decide if the gene labels should be plotted or not.
c.count.label.size	Determines the font size of the gene labels. By default the font size for count labels is 0.6.

Value

Returns a circos plot.

Examples

```
try({
VDJ_circos(Adj_matrix =  VDJ_alpha_beta_Vgene_circos_output[[2]][[1]],
  grid.col = VDJ_alpha_beta_Vgene_circos_output[[3]],
  group = VDJ_alpha_beta_Vgene_circos_output[[4]],
  c.count.label.size = 0.4,
  gene.label.size = 0.5,
  arr.col = data.frame(c("TRBV10"),c("TRBJ2-7"), c("black")),
  axis="percent")
})
```

VDJ_clonal_barplot *Function to create stacked barplots to visualize clonal expansion per repertoire directly from a VDJ matrix (either from the minimal_VDJ() or VDJ_GEX_matrix())*

Description

Function to create stacked barplots to visualize clonal expansion per repertoire directly from a VDJ matrix (either from the minimal_VDJ() or VDJ_GEX_matrix()).

Usage

```
VDJ_clonal_barplot(
  VDJ,
  counts.to.use,
  group.by,
  expanded.colors,
  non.expanded.color
)
```

Arguments

VDJ	VDJ matrix (either from the minimal_VDJ() or VDJ_GEX_matrix()
counts.to.use	The column name in the VDJ matrix of the clonotypes you want to use. Defaults to "clonotype_id".
group.by	The column name in the VDJ matrix on which you want to seperate the repertoire plots. If the entire VDJ matrix is one repertoire, this argument should be "none" or empty.
expanded.colors	Character vector. Colors to use for expanded clones. Should be more than 3 for better visibility. Defaults to a "darkorchid3"-based palette.
non.expanded.color	Character. Color to use for non expanded clones. Defaults to "black"

Value

Returns a list with a ggplot for each group.by element.

Examples

```
out <- VDJ_clonal_barplot(Platypus::small_vgm[[1]],
counts.to.use = "clonotype_id_10x",group.by = "sample_id")
```

`VDJ_clonal_donut` *Circular VDJ expansion plots*

Description

Generate circular plots of clonal expansion per repertoire directly from the VDJ matrix of the VDJ_GEX_matrix function

Usage

```
VDJ_clonal_donut(
  VDJ,
  counts.to.use,
  label.size,
  not.expanded.label.vjust,
  not.expanded.label.hjust,
  total.label.vjust,
  total.label.hjust,
  expanded.colors,
  non.expanded.color
)
```

Arguments

`VDJ`	VDJ dataframe generated using the VDJ_GEX_matrix function (VDJ_GEX_matrix.output[[1]]). Plots will be made by sample and using the clonal frequencies specified by counts.to.use
`counts.to.use`	How to count clonotypes and cells. A column name of the VDJ matrix containing clonotype IDs. This defaults to "clonotype_id_10x", which reflects clonotypes by Cellranger in an unaltered VGM. To use counts from the VDJ_clonotype_v3 function set this parameter to the relevant column e.g. "clonotype_id_cdr.aa" or "global_clonotype_id_cdr.aa" are two examples.
`label.size`	Size of text labels. All parameters below are purely for graphical purposes and optional. If necessary changes should be made in small (0.1) increments. ! It is recommended to optimize these ONLY once a format for saving the plot is set.
`not.expanded.label.vjust`	Numeric. Regulates the vertical position of the label for non expanded cells

not.expanded.label.hjust
: Numeric. Regulates the horizontal position of the label for non expanded cells

total.label.vjust
: Numeric. Regulates the vertical position of the center label

total.label.hjust
: Numeric. Regulates the horizontal position of the center label

expanded.colors
: Character vector. Colors to use for expanded clones. Should be more than 3 for better visibility. Defaults to a "darkorchid3"-based palette.

non.expanded.color
: Character. Color to use for non expanded clones. Defaults to "black"

Value

Returns a list of circular plots showing proportions of expanded clones and non-expanded clones. One plot is generated for each sample in the sample_id column

Examples

```
VDJ_clonal_donut(VDJ = Platypus::small_vgm[[1]])
```

VDJ_clonal_expansion *Flexible wrapper for clonal expansion barplots by isotype, GEX cluster etc.*

Description

Clonal frequency plot displaying clonal expansion for either T and B cells with Platypus VDJ_build() output. For Platypus v2 plotting of B cell clonotype expansion and isotypes please refer to VDJ_isotypes_per_clone.

Usage

```
VDJ_clonal_expansion(
  VDJ,
  celltype,
  clones,
  subtypes,
  isotypes.to.plot,
  species,
  treat.incomplete.clones,
  treat.incomplete.cells,
  group.by,
  color.by,
  variant.plot,
  text.size
)
```

Arguments

VDJ	VDJ dataframe generated using the VDJ_build function.
celltype	Character. Either "Tcells" or "Bcells". If set to Tcells bars will not be colored by default and the parameters treat_incomplete_cells, treat_incomplete_clones, subtypes and species are ignored. The color.by and group.by arguments work identically for both celltypes. If none provided it will detect this param from the celltype column.
clones	numeric value indicating the number of clones to be considered for the clonal expansion plot. Default value is 50. For a standard plot more than 50 is discouraged. When showing only one - possibly rare - isotype via isotypes.to.plot it may be useful to set this number higher (e.g. 100-200)
subtypes	Logical indicating whether to display isotype subtypes or not.
isotypes.to.plot	Character vector. Defaults to "all". This can be set to any number of specific Isotypes, that are to be shown exclusively. For example, to show only clones containing IgG, input "IGHG". If only wanting to check clones with IgA and IgD input c("IGHA","IGHD"). Works equally if subtypes are set to TRUE. Is ignored if color.by is not set to "isotype"
species	Character indicating whether the samples are from "Mouse" or "Human". Default is "Human".
treat.incomplete.clones	Character indicating how to proceed with clonotypes lacking a VDJC (in other words, no cell within the clonotype has a VDJC). "exclude" removes these clonotypes from the analysis. "include" keeps these clonotypes in the analysis. In the plot they will appear has having an unknown isotype.
treat.incomplete.cells	Character indicating how to proceed with cells assigned to a clonotype but missing a VDJC. "proportional" to fill in the VDJ isotype according to the proportions present in of clonotype (in case present proportions are not replicable in the total number of cells e.g. 1/3 in 10 cells, values are rounded to the next full integer and if the new counts exceed the total number of cells, 1 is subtracted from the isotype of highest frequency. If the number is below the number of cell, 1 is added to the isotype with lowest frequency to preserve diversity), "exclude" to exclude them from analysis and rank clonotypes only by the number of cells with a heavy chain. This ranking may deviate from the frequency column in the clonotype table. CAVE: if treat_incomplete_cells is set to "exclude", clonotypes lacking a VDJC entierly will be removed from the analysis. This results in a similar but not identical output as when treat_incomplete_clones is set to true. The two parameters are thereby non-redundant.
group.by	Character. Defaults to "sample_id". Column name of VDJ to split VDJ by. For each unique entry in that column a plot will be generated. Therefore plots can be generated by sample_id, group_id or any other metadata item.To get plots for the whole repertoire set to "none"
color.by	Character. Defaults to "isotype". If set to "isotype" bars are colored by the respective IgH chain or in grey for T cells. This can alternatively be set to any column name of the VDJ. This allows coloring clones by their V_gene usage or by GEX clusters

variant.plot	Logical indicating whether to plot the output showing the variants or not.
text.size	Numeric value indicating the size of the text in the plot. Default is 12.

Value

Returns a nested list. out[[1]] are plots out[[2]] are raw datatables containing also barcode and CDR3 information

Examples

```
clonal_out <- VDJ_clonal_expansion(VDJ = Platypus::small_vdj,
 celltype = "Bcells", clones = 30,subtypes = FALSE, species = "Mouse"
 ,treat.incomplete.clones = "exclude"
 ,treat.incomplete.cells = "proportional")
clonal_out[[1]] #list of plots
clonal_out[[2]] #list of source dataframes
```

VDJ_clonotype *Platypus V3 clonotyping wrapper*

Description

Updated clonotyping function based on implications for cells with different chain numbers than 1 VDJ 1 VJ chains.

This function offers two types of hierarchical clonotyping. The hierarchical option "single.chains" only merges cell with a single chain into clonotypes composed of cells with 1 VDJ 1 VJ chain. This is based on the assumption, that during mRNA capture and RT-PCR in GEMs, not all transcripts are captured and therefore cells may result missing a VDJ or VJ chain. The hierarchical option "double.and.single.chains" is based on the assumption, that cells with 1 VDJ and 2 VJ chains exist. For a review of the work concerning such cells as well as 2 VDJ 1 VJ cells please consult: https://doi.org/10.4049/jimmunol.1800904. The user may set a threshold of occurrence number above which cells with 1 VDJ 2 VJ chains are considered to be true and other cells with 1 VDJ 1 VJ, 1 VDJ 0 VJ and 0 VDJ 1 VDJ may be merged into the same clonotype by the strategy provided by the user. Cells with 2 VDJ chains are currently not considered in this process, as these are reported to be much rarer and, if appearing in the dataset are more likely to be doublets. We advice the user to carefully examine the output after hierarchical clonotyping before proceeding with further analysis. We thank Prof. Vijayanand as well as Vicente and Emmanuel from his lab for the discussions that have helped with improving the original Platypus clonotyping strategy.

Usage

```
VDJ_clonotype(
  VDJ,
  clone.strategy,
  homology.threshold,
  hierarchical,
  triple.chain.count.threshold,
```

```
  global.clonotype,
  VDJ.VJ.1chain,
  output.format,
  platypus.version
)
```

Arguments

VDJ	For platypus v2 output from VDJ_analyze function. This should be a list of clonotype dataframes, with each list element corresponding to a single VDJ repertoire. For platypus v3 VDJ output from the VDJ_GEX_matrix function (VDJ_GEX_matrix.output[[1]])
clone.strategy	(Updated keywords, previous format is also functional) String describing the clonotyping strategy. Possible options are 10x.default, cdr3.nt, cdr3.aa, VDJJ.VJJ, VDJJ.VJJ.cdr3length, VDJJ.VJJ.cdr3length.cdr3.homology, VDJJ.VJJ.cdr3length.VDJcdr3.homology, cdr3.homology, VDJcdr3.homology. cdr3.aa will convert the default cell ranger clonotyping to amino acid based. 'VDJJ.VJJ' groups B cells with identical germline genes (V and J segments for both heavy chain and light chain. Those arguments including 'cdr3length' will group all sequences with identical VDJ and VJ CDR3 sequence lengths. Those arguments including 'cdr3.homology' will additionally impose a homology requirement for CDRH3 and CDRL3 sequences.'CDR3.homology',or 'CDRH3.homology' will group sequences based on homology only (either of the whole CDR3 sequence or of the VDJ CDR3 sequence respectively). All homology calculations are performed on the amino acid level.
homology.threshold	Numeric value between 0 and 1 corresponding to the homology threshold forn the clone.strategy arguments that require a homology threshold. Default value is set to 70 percent sequence homology. For 70 percent homology, 0.3 should be supplied as input.
hierarchical	Character. Defaults to "none". This is an extention specifically for cells with aberrant numbers of chains (i.e. 0VDJ 1VJ, 1VDJ 0VJ, 0VDJ 2VJ, 2VDJ 0VJ). Cells with 2VDJ 2VJ are filtered out as these are most likely doublets. If set to "none" aberrant cells are assigned to their own clonotypes. If set to "single.chains" the function will proceed in two steps: 0. Prefiltering: cells with 2 VDJ 2 VJ chains as well as cells with 2 VDJ and any number of VJ chains are filtered out. 1. define clonotypes classically with all cells containing exactly 1VDJ 1VJ chains. 2. For cells with only a single chain (either VDJ or VJ), check if any clone exists, which matches the clonotyping criteria for this chain. If true, add this cell to that clone. If false, create a new clone containing that cell. In case that more than 1 existing clone matches the aberrant cell, the cell is assigned to the most frequent existing clone. Two reasons are behind this decision: 2.1. The aberrant cells is numerically more likely to be a part of the more frequent existing clone. 2.2 In case of a wrong assignment, the effect of the error is lower, if an already expanded clone is increase by one count, rather than a existing non-expanded clone being assigned a second entry and thereby resulting as expanded. Cells If set to "double.and.single.chains" the function will proceed as if set to "single.chains" but include two more steps 3. Check the

frequency of each cell 1 VDJ 2 VJ chain exact clone (by exact nucleotide CDR3 matching). Only if this count exceeds the triple.chain.count.threshold, the clone is used as a "hub clone". This protects from merging clonotypes on the basis of rare doublets. 4. Merge existing clonotypes into the 1 VDJ 2 VJ clonotypes as they match with the assumption that e.g. a cell with 1 VDJ 1 VJ is part of that same clonotype, but missing a VJ chain due to stochastical sampling

`triple.chain.count.threshold`
: Minimal occurrance frequency for any cell with more than 2 of either VDJ or VJ chain (e.g. 2 VDJ 1 VJ) for it to be considered as a trustworthy clone for hierarchical clonotyping ONLY when hierarchical is set to "double.and.single.chains". Defaults to 3, meaning that, an exact combination of three chains needs to appear in the dataset at least 3 times for it to be considered as a clone, into which other cells are merged. (For the counting of exact combination of chains CDR3 nucleotide string matching is used, even if clonotyping by homology)

`global.clonotype`
: Logical specifying whether clonotyping should occur across samples or only within a single sample (grouping via sample_id column).

`VDJ.VJ.1chain`
: Logical specifying whether cells other than once with 1 VDJ and 1 VJ chains should be considered.

`output.format`
: Parameter output.format is deprecated. If non VGM-style output is required please refer to the function VDJ_clonotype. Output is VGM style VDJ by cell dataframe

`platypus.version`
: Only "v3" available

Value

Returns a VGM[[1]]-type dataframe. The columns clonotype_id and clonotype_frequency are updated with the new clonotyping strategy. They represent the "active strategy" that downstream functions will use. Furthermore extra columns are added with clonotyping information.New columns are named by clonotyping strategy so to allow for multiple clonotyping identifiers to be present in the same VDJ dataframe and make comparisons between these straighforward.

Examples

```
reclonotyped_vgm <- VDJ_clonotype(VDJ=Platypus::small_vgm[[1]],
clone.strategy="cdr3.nt",
hierarchical = "none", global.clonotype = TRUE)
```

`VDJ_clonotype_v3_w_enclone`
: *Updated clonotyping function based on implications for cells with different chain numbers than 1 VDJ 1 VJ chains.*

Description

This function offers two types of hierarchical clonotyping. The hierarchical option "single.chains" only merges cell with a single chain into clonotypes composed of cells with 1 VDJ 1 VJ chain. This is based on the assumption, that during mRNA capture and RT-PCR in GEMs, not all transcripts are captured and therefore cells may result missing a VDJ or VJ chain. The hierarchical option "double.and.single.chains" is based on the assumption, that cells with 1 VDJ and 2 VJ chains exist. For a review of the work concerning such cells as well as 2 VDJ 1 VJ cells please consult: https://doi.org/10.4049/jimmunol.1800904. The user may set a threshold of occurrence number above which cells with 1 VDJ 2 VJ chains are considered to be true and other cells with 1 VDJ 1 VJ, 1 VDJ 0 VJ and 0 VDJ 1 VDJ may be merged into the same clonotype by the strategy provided by the user. Cells with 2 VDJ chains are currently not considered in this process, as these are reported to be much rarer and, if appearing in the dataset are more likely to be doublets. We advice the user to carefully examine the output after hierarchical clonotyping before proceeding with further analysis. We thank Prof. Vijayanand as well as Vicente and Emmanuel from his lab for the discussions that have helped with improving the original Platypus clonotyping strategy.

Usage

```
VDJ_clonotype_v3_w_enclone(
  VDJ,
  VDJ.directory,
  clone.strategy,
  samples.to.clonotype,
  samples.to.combine,
  homology.threshold,
  hierarchical,
  triple.chain.count.threshold,
  global.clonotype,
  VDJ.VJ.1chain,
  same.origin,
  platypus.version,
  operating.system
)
```

Arguments

VDJ	For platypus v2 output from VDJ_analyze function. This should be a list of clonotype dataframes, with each list element corresponding to a single VDJ repertoire. For platypus v3 VDJ output from the VDJ_build function
VDJ.directory	Cellranger output directory for VDJ files.
clone.strategy	(Updated keywords, previous format is also functional) String describing the clonotyping strategy. Possible options are 10x.default, cdr3.nt, cdr3.aa, VDJJ.VJJ, VDJJ.VJJ.cdr3length, VDJJ.VJJ.cdr3length.cdr3.homology, VDJJ.VJJ.cdr3length.VDJcdr3.homology, cdr3.homology, VDJcdr3.homology. cdr3.aa will convert the default cell ranger clonotyping to amino acid based. 'VDJJ.VJJ' groups B cells with identical germline genes (V and J segments for both heavy chain and light chain. Those arguments including 'cdr3length' will group all sequences with identical VDJ and VJ CDR3 sequence lengths. Those arguments including 'cdr3.homology'

will additionally impose a homology requirement for CDRH3 and CDRL3 sequences.'CDR3.homology',or 'CDRH3.homology' will group sequences based on homology only (either of the whole CDR3 sequence or of the VDJ CDR3 sequence respectively). All homology calculations are performed on the amino acid level.

`samples.to.clonotype`
: Vector - lists the samples names which should be clonotyped. The unspecified samples will keep their old clonotype defintions.

`samples.to.combine`
: Vector or list of vectors - lists the samples which you wish to have their clonotypes merged (e.g., c('s1','s2') to only merge the first 2 samples, or list(c('s1','s3'), c('s2', 's4')) to merge the first and third, second and fourth, respectively). global.clonotype must be set to T!

`homology.threshold`
: Numeric value between 0 and 1 corresponding to the homology threshold forn the clone.strategy arguments that require a homology threshold. Default value is set to 70 percent sequence homology. For 70 percent homology, 0.3 should be supplied as input.

`hierarchical`
: Character. Defaults to "none". This is an extention specifically for cells with aberrant numbers of chains (i.e. 0VDJ 1VJ, 1VDJ 0VJ, 0VDJ 2VJ, 2VDJ 0VJ). Cells with 2VDJ 2VJ are filtered out as these are most likely doublets. If set to "none" aberrant cells are assigned to their own clonotypes. If set to "single.chains" the function will proceed in two steps: 0. Prefiltering: cells with 2 VDJ 2 VJ chains as well as cells with 2 VDJ and any number of VJ chains are filtered out. 1. define clonotypes classically with all cells containing exactly 1VDJ 1VJ chains. 2. For cells with only a single chain (either VDJ or VJ), check if any clone exists, which matches the clonotyping criteria for this chain. If true, add this cell to that clone. If false, create a new clone containing that cell. In case that more than 1 existing clone matches the aberrant cell, the cell is assigned to the most frequent existing clone. Two reasons are behind this decision: 2.1. The aberrant cells is numerically more likely to be a part of the more frequent existing clone. 2.2 In case of a wrong assignment, the effect of the error is lower, if an already expanded clone is increase by one count, rather than a existing non-expanded clone being assigned a second entry and thereby resulting as expanded. Cells If set to "double.and.single.chains" the function will proceed as if set to "single.chains" but include two more steps 3. Check the frequency of each cell 1 VDJ 2 VJ chain exact clone (by exact nucleotide CDR3 matching). Only if this count exceeds the triple.chain.count.threshold, the clone is used as a "hub clone". This protects from merging clonotypes on the basis of rare doublets. 4. Merge existing clonotypes into the 1 VDJ 2 VJ clonotypes as they match with the assumption that e.g. a cell with 1 VDJ 1 VJ is part of that same clonotype, but missing a VJ chain due to stochastical sampling

`triple.chain.count.threshold`
: Minimal occurrance frequency for any cell with more than 2 of either VDJ or VJ chain (e.g. 2 VDJ 1 VJ) for it to be considered as a trustworthy clone for hierarchical clonotyping ONLY when hierarchical is set to "double.and.single.chains". Defaults to 3, meaning that, an exact combination of three chains needs to appear in the dataset at least 3 times for it to be considered as a clone, into which

other cells are merged. (For the counting of exact combination of chains CDR3 nucleotide string matching is used, even if clonotyping by homology)

`global.clonotype`
: Logical specifying whether clonotyping should occur across samples or only within a single sample (grouping via sample_id column).

`VDJ.VJ.1chain`
: Logical specifying whether cells other than once with 1 VDJ and 1 VJ chains should be considered.

`same.origin`
: Logical - if the merged samples come from the same donor, with the same or with different origins. If two datasets come from the same origin, enclone will filter to remove certain artifacts.

`platypus.version`
: Only "v3" available

`operating.system`
: Character - operating system on which enclone will be run. 'Windows' for Windows, 'Linux' for Linux, 'Darwin' for MacOS.

Value

Returns a VGM[[1]]-type dataframe. The columns clonotype_id and clonotype_frequency are updated with the new clonotyping strategy. They represent the "active strategy" that downstream functions will use. Furthermore extra columns are added with clonotyping information.New columns are named by clonotyping strategy so to allow for multiple clonotyping identifiers to be present in the same VDJ dataframe and make comparisons between these straighforward.

Examples

```
reclonotyped_vgm <- VDJ_clonotype(VDJ=Platypus::small_vgm[[1]],
clone.strategy="cdr3.nt",
hierarchical = "none", global.clonotype = TRUE)
```

`VDJ_contigs_to_vgm` *Formats "VDJ_contigs_annotations.csv" files from cell ranger to match the VDJ_GEX_matrix output using only cells with 1VDJ and 1VJ chain*

Description

Formats "VDJ_contigs_annotations.csv" files from cell ranger to match the VDJ_GEX_matrix output using only cells with 1VDJ and 1VJ chain

Usage

```
VDJ_contigs_to_vgm(directory, sample.names, celltype, FB, platypus.version)
```

Arguments

directory	list containing paths to the "filtered_contig_annotations.csv" files from cell ranger.
sample.names	vector specifying sample names.
celltype	Character. Either "Tcells" or "Bcells".
FB	Integer specifying whether VGM should contain Feature Barcode columns or not. Default set to FALSE.
platypus.version	Function based on VGM object from V3, no need to set this parameter.

Value

data frame with column names that match the VDJ_GEX_matrix output. Can be appended to the VDJ_GEX_matrix output

Examples

```
try({
directory.list <- list()
directory.list[[1]] <- c("~/Dataset_1/filtered_contig_annotations.csv")
directory.list[[2]] <- c("~/Dataset_1/filtered_contig_annotations.csv")
filtered_contig_vgm <- VDJ_contigs_to_vgm(directory = directory.list,
sample.names = c(s3,s4), celltype = "Tcells")
})
```

VDJ_db_annotate *Wrapper function of VDJ_antigen_integrate function*

Description

Wraps the VDJ_antigen_integrate function and uses it to annotate a VDJ dataframe with antigen information. Needs to VDJ_db_load to be executed first, with preprocess=T and vgm.names=T to obtain the same column names as in the VDJ (to allow for sequence matching).

Usage

```
VDJ_db_annotate(VDJ, db.list, database.features, match, homology, lv.distance)
```

Arguments

VDJ	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
db.list	list of database dataframes or csv file paths, obtained from VDJ_db_load with .
database.features	list of features/column names to be integrated from the databases.

match	string - sequences by which to match and integrate the antigen information. Currently, only 'cdr3.aa' and 'cdrh3.aa' are supported, as all databases have these two sequence types ('VJ_cdr3s_aa','VDJ_cdr3s_aa').
homology	string - 'exact' for exact sequence matchings, 'homology' for homology matching.
lv.distance	integer - maximum Levehnstein distance threshold for the homology matchings.

Value

VDJ with new columns - antigen information integrated from the antigen databases.

Examples

```
try({
VDJ_db_annotate(VDJ=VDJ,db.list=db.list,database.features='Epitope',match='cdr3.aa',homology=FALSE)
})
```

VDJ_db_load *Load and preprocess a list of antigen-specific databases*

Description

Preprocessing function for several antigen databases for both TCRs (VDJdb, McPAS-TCR, TBAdb) and BCRs (TBAdb), saving them either at a specified path, or loading them as a database list for downstream integration/analyses.

Usage

```
VDJ_db_load(
  databases,
  file.paths,
  preprocess,
  species,
  filter.sequences,
  remove.na,
  vgm.names,
  keep.only.common,
  output.format,
  saving.path
)
```

Arguments

databases	list of databases to be processed and saved. Currently supported ones include: VDJdb(='vdjdb'), McPAS-TCR(='mcpas'), TBAdb(='tbdadb_tcr' or 'tbadb_bcr').

file.paths	list of file paths for the specified databases (in the database parameter). If NULL, will try to locally download the databases from the archived download links.
preprocess	boolean - if T, will preprocess each database individually.
species	string - either 'Human' or 'Mouse', the species for the processed database. Needs preprocess=T.
filter.sequences	string - 'VDJ' to remove rows with NA VDJ sequences, 'VJ' to remove rows with NA VJ sequences, 'VDJ.VJ' to remove rows with both VDJ and VJ sequences missing. Needs preprocess=T.
remove.na	string or NULL - 'all' will remove all rows with missing values from the database, 'common' will remove only rows with missing values for the shared columns among all databases ('VJ_cdr3s_aa','VDJ_cdr3s_aa','Species','Epitope','Antigen species'), 'vgm' will remove missing values for columns shared with the VDJ object (specific to each database). Needs preprocess=T.
vgm.names	boolean - if T, will change all column names of the shared columns (with VDJ) to match those from VDJ. Use this to integrate the antigen data into VDJ using VDJ_antigen_integrate or VDJ_db_annotate. Needs preprocess=T.
keep.only.common	boolean - if T, will only keep the columns shared between all databases ('VJ_cdr3s_aa','VDJ_cdr3s_aa','S species') for each processed database. Needs preprocess=T.
output.format	string - 'df.list' to save all databases as a list, 'save' to save them as csv files.
saving.path	string - directory where the processed databases should be locally saved if output.format='save'.

Value

Processed antigen-specific databases for both TCRs and BCRs.

Examples

```
try({
VDJ_db_load(databases=list('vdjdb'),file.paths=NULL,
preprocess=TRUE,species='Mouse',filter.sequences='VDJ.VJ',
remove.na='vgm', vgm.names=TRUE, keep.only.common=TRUE,
output.format='df.list', saving.path = tempdir())
})
```

VDJ_diversity *Calculates and plots common diversity and overlap measures for repertoires and alike. Requires the vegan package*

Description

Calculates and plots common diversity and overlap measures for repertoires and alike. Requires the vegan package

Usage

```
VDJ_diversity(
  VDJ,
  feature.columns,
  grouping.column,
  metric,
  VDJ.VJ.1chain,
  subsample.to.same.n
)
```

Arguments

VDJ	VDJ dataframe output from the VDJ_build function.
feature.columns	Character vector. One or more column names from the VDJ of which diversity or overlap metrics are calculated. if more than one column is provided (e.g. c("VDJ_cdr3s_aa","VJ_cdr3s_aa")) these columns will be pasted together before metric calculation.
grouping.column	Character. Column name of a column to group metrics by. This could be "sample_id" to calculate the metric for each sample. This column is required if metric = "simpson". If so, the simpson overlap index will be calculated pairwise for all combinations of elements in the grouping.column. Defaults to "none".
metric	Character. Diversity or overlap metric to calculate. Can be c("richness", "berger-parker", "simpson", "ginisimpson", "shannon", "shannonevenness", "jaccard"). Defaults to "shannon". If jaccard is selected, a heatmap with the pairwise comparisons between all groups is returned. If any of the others is selected, a dotplot is returned
VDJ.VJ.1chain	Boolean defaults to TRUE. Whether to filter out aberrant cells (more than 1 VDJ or VJ chain).
subsample.to.same.n	Boolean defaults to TRUE. Whether to subsample larger groups down to the size of the smallest group

Value

Returns a ggplot with the calculated metric for each group (if provided).

Examples

```
VDJ_diversity(VDJ = Platypus::small_vdj,
,feature.columns = c("VDJ_jgene"), grouping.column = "sample_id"
,metric = "jaccard")
```

VDJ_dynamics *Tracks a specific VDJ column across multiple samples/timepoints.*

Description

Track a VDJ column across multiple samples or timepoints. Tracking consists of creating a per sample/timepoint dataframe of unique values for the VDJ column and their respective counts inside that timepoints/repertoire. Also creates alluvial plots to show the temporal dynamics of the tracked elements.

Usage

```
VDJ_dynamics(
  VDJ,
  columns.to.track,
  starting.point.repertoire,
  track.all.elements,
  track.only.common,
  max.elements.to.track,
  specific.elements.to.track,
  additional.grouping.column,
  max.additional.groups,
  specific.additional.groups,
  timepoints.column,
  proportions.level,
  output.format,
  ignore.legend
)
```

Arguments

VDJ	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
columns.to.track	string or list of strings - VDJ column with values to track (e.g., 'VDJ_cgene' will track the changes in isotype counts/proportions across multiple timepoints, defined by the timepoints.column). If two columns are provided and tracked, then a new values will be created by combining the values from each column.
starting.point.repertoire	string or integer - the repertoire from which to start tracking (1 = will start at the first repertoire, 's3' will start at repertoire 's3').
track.all.elements	boolean - if T (and track.only.common=F), it will track all elements across all repertoires/timepoints.
track.only.common	boolean - if T (and track.all.elements=F), it will only track the common elements across all repertoires/timepoints.

`max.elements.to.track`
: integer or NULL - the maximum number of elements to track (elements are first sorted by frequency/abundance). If NULL, it will track all elements.

`specific.elements.to.track`
: vector of strings or NULL - specific elements we want tracked. If NULL, all elements will be tracked.

`additional.grouping.column`
: string or 'none' - VDJ column for calculating the frequency/counts of elements on a per-group level. If output.format='plot', each unique group will have its own bar plot of timepoints/repertoires (x axis) and feature counts (y axis). If NULL, no additional grouping will be done.

`max.additional.groups`
: integer or NULL - the maximum number of additional groups to consider (groups are first ordered by their frequency = total number of cells in that group in the VDJ matrix). If NULL, all groups will be considered.

`specific.additional.groups`
: vector of strings or NULL - specific grouping factors we want to consider. If NULL, all grouping factors will be considered.

`timepoints.column`
: string - VDJ column with either timepoints or repertoires across which we want to track our elements (usually 'sample_id').

`proportions.level`
: string - 'absolute.counts' for absolute counts, 'group' for per group proportions, 'repertoire' for per repertoire/timepoint proportions.

`output.format`
: string - 'plot' for alluvial barplots, 'df' for count/proportions dataframes of the tracked elements.

`ignore.legend`
: boolean - if T, the legend will not be included in the resulting ggplot object.

Value

Either a count dataframe of the tracked elements across multiple timepoints/repertoires, or alluvial barplot.

Examples

```
VDJ_dynamics(VDJ = Platypus::small_vgm[[1]],
columns.to.track='clonotype_id', starting.point.repertoire=1,
max.elements.to.track=10, timepoints.column='sample_id',
output.format='plot')
```

`VDJ_expand_aberrants` *Expand the aberrant cells in a VDJ dataframe by converting them into additional rows*

Description

Expand the aberrant cells in a VDJ dataframe by converting them into additional rows. Aberrant cells consist of cells with more than 1 VDJ or VJ chain.

Usage

```
VDJ_expand_aberrants(
  VDJ,
  chain.to.expand,
  add.barcode.prefix,
  additional.VDJ.features,
  additional.VJ.features,
  add.CDR3aa,
  add.expanded.number,
  recalculate.clonotype.frequency
)
```

Arguments

`VDJ`	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
`chain.to.expand`	string, 'VDJ' to expand VDJ aberrants, 'VJ' to expand VJ aberrants, 'VDJ.VJ' for both.
`add.barcode.prefix`	boolean - if T, a new barcode will be added for each expanded aberrant.
`additional.VDJ.features`	vector of strings - VDJ_expand_aberrants will only expand across the sequence columns of VDJ. If you have additional columns with aberrant cell features (e.g., both 'yes' and 'no' binders for a single sequence), where the aberrants are VDJ-specific, include them here.
`additional.VJ.features`	vector of strings - VDJ_expand_aberrants will only expand across the sequence columns of VDJ. If you have additional columns with aberrant cell features (e.g., both 'yes' and 'no' binders for a single sequence), where the aberrants are VJ-specific, include them here.
`add.CDR3aa`	boolean - if T, will create a new column 'CDR3aa' with pasted VDJ_cdr3s_aa and VJ_cdr3s_aa.
`add.expanded.number`	boolean - if T, will add the number of new cells resulting from an aberrant one.

recalculate.clonotype.frequency
: boolean - if T, will recalculate the clonotype frequencies for the resulting, expanded VDJ.

Value

Returns a VDJ format dataframe in which cells with more than one VDJ or VJ chain are split into multiple rows each containing only one VDJ VJ chain combination.

Examples

```
VDJ_expand_aberrants(VDJ = Platypus::small_vgm[[1]],
chain.to.expand='VDJ.VJ',
add.barcode.prefix=TRUE, recalculate.clonotype.frequency=FALSE)
```

VDJ_extract_germline_consensus_ref

Making the trimmed reference and concatenating fr1-fr4

Description

Function that takes the VDJ and the fr1-fr4 sequence per antibody Based on the ref argument, if TRUE it also returns the returns in the VDJ/VJ_ref.nt/aa the trimmed reference based on the alignement with the consensus.

Usage

```
VDJ_extract_germline_consensus_ref(
  VDJ,
  n_clones = NA,
  samples = NA,
  ref = TRUE,
  path_toData = "../Data/"
)
```

Arguments

VDJ	VDJ or vgm[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
n_clones	integer, denoting the top n clones to get the reference. If NA it is performed in all clones
samples	list of sample names, with the same order as they were accessed to make the VGM
ref	bool, denoting whether or not we trim the reference of the antibodies.
path_toData	str, denoting the folder containing the VDJ folder with VDJ information per sample

Value

$vdj: VDJ containing the VDJ/VJ_ref.nt/aa columns if ref = TRUE and the full_VDJ, full_VJ columns with the fr1-fr4. $clones: clone_ids for which a reference was made.

Examples

```
try({
samples = c('LCMV', 'TNFR')
vgm = read("VGM.RData")
n_clones = 20
result = VDJ_extract_germline_consensus_ref(vgm$VDJ, n_clones,
samples, ref = TRUE,
path_toData="../Data/")
VDJ = result[1]$vdj
clone_counts = result[2]$clones
})
```

VDJ_germline *Infer germline from the desired software/caller*

Description

Function to infer the germline from the tree

Usage

```
VDJ_germline(VDJ, germlines.from, VDJ.only)
```

Arguments

VDJ	VDJ dataframe obtained after calling VDJ_call_MIXCR or any other germline you want to use
germlines.from	MIXCR or any other germline caller - default: MIXCR
VDJ.only	boolean - if T, only Heavy Chain (VDJ) germline will be inferred

Value

VDJ with the updated germline

Examples

```
try({
VDJ_germline(VDJ, germlines.from='MIXCR',
VDJ.only=T)
})
```

VDJ_get_public *Function to get shared/public elements across multiple repertoires*

Description

Function to get shared elements across multiple repertoires, specified by the feature.columns parameter (a column of the VDJ matrix). If two columns are specified in feature.columns, the resulting shared features will combine the values from each column (at a per-cell level).

Usage

```
VDJ_get_public(
  VDJ,
  feature.columns,
  repertoire.column,
  specific.repertoires,
  find.public.all,
  find.public.percentage,
  treat.combined.features,
  output.format
)
```

Arguments

VDJ	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
feature.columns	Character or character vector columns of features to be assayed
repertoire.column	string - the repertoire-defining column (default to 'sample_id').
specific.repertoires	vector of strings or NULL - if only the shared elements from specific repertoires should be taken into account. If NULL, will output the shared/public elements across all repertoires.
find.public.all	boolean - if T, will look for the public elements across all repertoires
find.public.percentage	list - the first element denotes the percentage of repertoires to get shared elements for, the second element is the maximum number of repertoire combinations to consider (can be NULL to consider all).
treat.combined.features	string - 'exclude' will exclude combined features with one element missing, 'include' will include and considers them as a new feature value.
output.format	string - 'df' to get a shared element dataframe (with columns = Repertoire and Public), 'list' for a list of shared elements.

Value

Either a dataframe of public elements across multiple repertoires or a list.

Examples

```
VDJ_get_public(VDJ = Platypus::small_vgm[[1]],
feature.columns='VDJ_cdr3s_aa', find.public.all=TRUE,
output.format='df')
```

`VDJ_GEX_clonotype_clusters_circos`

Makes a Circos plot from the VDJ_GEX_integrate output. Connects the clonotypes with the corresponding clusters.

Description

Makes a Circos plot from the VDJ_GEX_integrate output. Connects the clonotypes with the corresponding clusters.

Usage

```
VDJ_GEX_clonotype_clusters_circos(
  VGM,
  topX,
  label.threshold,
  axis,
  c.threshold,
  c.count.label,
  c.count.label.size,
  n_cluster,
  platypus.version,
  gene.label,
  gene.label.size,
  arr.col,
  arr.direction,
  platy.theme,
  clonotype.column
)
```

Arguments

`VGM`	The output of the VDJ_GEX_matrix function (VDJ_GEX_matrix.output[[1]]) has to be supplied. For Platypus v2: The output of the VDJ_GEX_integrate function (Platypus platypus.version v2). A list of data frames for each sample containing the clonotype information and cluster membership information.

`topX` Filters for the top X clonotypes and only plots the respective gene combinations or cluster memberships.

`label.threshold` Genes are only labeled if the count is larger then the label.threshold. By default all label.threshold = 0 (all genes are labeled).

`axis` Character. Axis scaling. Defaults to "max". Passed to VDJ_circos

`c.threshold` Only clonotypes are considered with a frequency higher then c.threshold. Allows to filter for only highly expanded clonotypes.

`c.count.label` Boolean, lets the user decide if the gene and count labels should be plotted or not. Default = T.

`c.count.label.size` Determines the font size of the gene labels. By default the font size for count labels is 0.6.

`n_cluster` Integer. No default.

`platypus.version` Input version to use. Defaults to "v3" for VDJ_GEX_matrix input

`gene.label` Boolean, lets the user decide if the gene labels should be plotted or not.

`gene.label.size` Determines the font size of the gene labels. By default the labelsize is automatically adjusted to 0.7 for labels with two or less digits, 0.6 for labels between 2 and 6 digits, and 0.4 for all longer labels. A manually defined font size will be the same for all labels!

`arr.col` Data.frame with three columns where the first two indicate the names of genes, clonotypes or clusters to be connected, and the third corresponds to the color of the arrow. Default set to data.frame(c("dummy.clonotype"), c("dummy.cluster"), c("dummy.color")), so no arrow is drawn.

`arr.direction` Either 1 or -1 and determines the direction of the arrow. Default=1.

`platy.theme` Allows plotting in the new "pretty" theme or the older "spiky" theme without group labels and radial arrangement of gene.labels. Default = "pretty".

`clonotype.column` Which column in VGM contains the clonotyping information? Default="clonotype_id_10X".

Value

Returns a circos plot and a list object with the following elememts for N samples: [[1 to N]] The first N listelements corresponds to the recorded circos plots for N beeing the number or samples in the VGM. Since Circlize uses the R base plotting funciton, this is not a ggplot object but can still be replotted by calling the first list element. [[N+1]] Adjacency matrix forwarded to VDJ_circos(). This Matrix contains the counts and can be used for manual replotting using VDJ_circos directly. [[N+2]] Contains a named list with colors for each connection drawn and can be used for manual replotting using VDJ_circos directly. [[N+3]] Contains a named list with grouping information and can be used for manual replotting using VDJ_circos directly.

Examples

```
try({
 clonotype.clusters <- VDJ_GEX_clonotype_clusters_circos(Platypus::small_vgm[[1]],
 n_cluster=8, topX = 20)
 clonotype.clusters[[1]]
 })
```

VDJ_GEX_matrix *VDJ GEX processing and integration wrapper*

Description

This function is designed as a common input to the Platypus pipeline. Integration of datasets as well as VDJ and GEX information is done here. Please check the Platypus V3 vignette for a detailed walkthrough of the output structure. In short: output[[1]] = VDJ table, output[[2]] = GEX Seurat object and output[[3]] = statistics [FB] Feature barcode (FB) technology is getting increasingly popular, which is why Platypus V3 fully supports their use as sample delimiters. As of V3, Platpyus does not support Cite-seq data natively, also the VDJ_GEX_matrix function is technically capable of loading a Cite-seq matrix and integrating it with VDJ. For details on how to process sequencing data with FB data and how to supply this information to the VDJ_GEX_matrix function, please consult the dedicated vignette on FB data.

Usage

```
VDJ_GEX_matrix(
  VDJ.out.directory.list,
  GEX.out.directory.list,
  FB.out.directory.list,
  Data.in,
  Seurat.in,
  group.id,
  GEX.read.h5,
  VDJ.combine,
  GEX.integrate,
  integrate.GEX.to.VDJ,
  integrate.VDJ.to.GEX,
  exclude.GEX.not.in.VDJ,
  filter.overlapping.barcodes.GEX,
  filter.overlapping.barcodes.VDJ,
  get.VDJ.stats,
  append.raw.reference,
  select.excess.chains.by.umi.count,
  excess.chain.confidence.count.threshold,
  trim.and.align,
  parallel.processing,
```

```
  numcores,
  gap.opening.cost,
  gap.extension.cost,
  exclude.on.cell.state.markers,
  exclude.on.barcodes,
  integration.method,
  VDJ.gene.filter,
  mito.filter,
  norm.scale.factor,
  n.feature.rna,
  n.count.rna.min,
  n.count.rna.max,
  n.variable.features,
  cluster.resolution,
  neighbor.dim,
  mds.dim,
  FB.count.threshold,
  FB.ratio.threshold,
  FB.exclude.pattern,
  subsample.barcodes,
  verbose
)
```

Arguments

`VDJ.out.directory.list`
: List containing paths to VDJ output directories from cell ranger. This pipeline assumes that the output file names have not been changed from the default 10x settings in the /outs/ folder. This is compatible with B and T cell repertoires. ! Neccessary files within this folder: filtered_contig_annotations.csv, clonotypes.csv, concat_ref.fasta, all_contig_annotations.csv (only if trim.and.align == TRUE) and metrics_summary.csv (Optional, will be appended to stats table if get.VDJ.stats == TRUE)

`GEX.out.directory.list`
: List containing paths the outs/ directory of each sample or directly the raw or filtered_feature_bc_matrix folder. Order of list items must be the same as for VDJ. These may be paths to cellranger aggr or cellranger multi output directories. In that case, additional matrices found, will be loaded as either GEX or FB (Feature barcodes) depending on the number of features in the matrix.

`FB.out.directory.list`
: [FB] List of paths pointing at the outs/ directory of output from the Cellranger counts function which contain Feature barcode counts. ! Single list elements can be a path or "PLACEHOLDER", if the corresponding input in the VDJ or GEX path does not have any adjunct FB data. This is only the case when integrating two datasets of which only one has FB data. See examples for details. Any input will overwrite potential FB data loaded from the GEX input directories. This may be important, if wanting to input unfiltered FB data that will cover also cells in VDJ not present in GEX.

`Data.in` Input for R objects from either the PlatypusDB_load_from_disk or the PlatypusDB_fetch function. If provided, input directories should not be specified. If you wish to integrate local and downloaded data, please load them via load_from_disk and fetch and provide as a list (e.g. Data.in = list(load_from_disk.output, fetch.output))

`Seurat.in` Alternative to GEX.out.directory.list. A seurat object. VDJ.integrate has to be set to TRUE. In metadata the column of the seurat object, sample_id and group_id must be present. sample_id must contain ids in the format "s1", "s2" ... "sn" and must be matching the order of VDJ.out.directory.list. No processing (i.e. data normalisation and integration) will be performed on these objects. They will be returned as part of the VGM and with additional VDJ data if integrate.VDJ.to.GEX = T. Filtering parameters such as overlapping barcodes, exclude.GEX.not.in.VDJ and exclude.on.cell.state.markers will be applied to the Seurat.in GEX object(s).

`group.id` vector with integers specifying the group membership. c(1,1,2,2) would specify the first two elements of the input VDJ/GEX lists are in group 1 and the third/fourth input elements will be in group 2.

`GEX.read.h5` Boolean. defaults to FALSE. Whether to read GEX data from an H5 file. If set to true, please provide the each directory containing a cellranger H5 output file or a direct path to a filtered_feature_bc_matrix.h5 as one GEX.out.directory.list element.

`VDJ.combine` Boolean. Defaults to TRUE. Whether to integrate repertoires. A sample identifier will be appended to each barcode both in GEX as well as in VDJ. Recommended for all later functions

`GEX.integrate` Boolean. Defaults to TRUE. Whether to integrate GEX data. Default settings use the seurat scale.data option to integrate datasets. Sample identifiers will be appended to each barcode both in GEX and VDJ This is helpful when analysing different samples from the same organ or tissue, while it may be problematic when analysing different tissues.

`integrate.GEX.to.VDJ`
Boolean. defaults to TRUE. Whether to integrate GEX metadata (not raw counts) into the VDJ output dataframe ! Only possible, if GEX.integrate and VDJ.combine are either both FALSE or both TRUE

`integrate.VDJ.to.GEX`
Boolean. defaults to TRUE. Whether to integrate VDJ data into GEX seurat object as metadata. ! Only possible, if GEX.integrate and VDJ.combine are either both FALSE or both TRUE

`exclude.GEX.not.in.VDJ`
Boolean. defaults to FALSE. Whether to delete all GEX cell entries, for which no VDJ information is available. Dependent on data quality and sequencing depth this may reduce the GEX cell count by a significant number

`filter.overlapping.barcodes.GEX`
Boolean. defaults to TRUE. Whether to remove barcodes which are shared among samples in the GEX analysis. Shared barcodes normally appear at a very low rate.

`filter.overlapping.barcodes.VDJ`
Boolean. defaults to TRUE. Whether to remove barcodes which are shared among samples in the GEX analysis. Shared barcodes normally appear at a very low rate.

`get.VDJ.stats` Boolean. defaults to TRUE. Whether to generate general statistics table for VDJ repertoires. This is appended as element [[3]] of the output list.

`append.raw.reference`
Boolean. Defaults to TRUE. This appends the raw reference sequence for each contig even if trim.and.align is set to FALSE.

`select.excess.chains.by.umi.count`
Boolean. Defaults to FALSE. There are several methods of dealing with cells containing reads for more than 1VDJ and 1VJ chain. While many analyses just exclude such cells, the VGM is designed to keep these for downstream evaluation (e.g. in VDJ_clonotype). This option presents an evidenced-based way of selectively keeping or filtering only one of the present VDJ and VJ chains each. This works in conjunction with the parameter excess.chain.confidence.count.threshold (below) Idea source: Zhang W et al. Sci Adv. 2021 (10.1126/sciadv.abf5835)

`excess.chain.confidence.count.threshold`
Interger. Defaults to 1000. This sets a umi count threshold for keeping excessive chains in a cell (e.g. T cells with 2 VJ and 1 VDJ chain) and only has an effect if select.excess.chains.by.umi.count is set to TRUE. For a given cell with chains and their UMI counts: VDJ1 = 3, VDJ2 = 7, VJ1 = 6. If count.threshold is kept at default (1000), the VDJ chain with the most UMIs will be kept (VDJ2), while the other is filtered out (VDJ1), leaving the cell as VDJ2, VJ1. If the count.threshold is set to 3, both chains VDJ chains of this cell are kept as their UMI counts are equal or greater to the count.threshold and therefore deemed high confidence chains. In the case of UMI counts being equal for two chains AND below the count.threshold, the first contig entry is kept, while the second is filtered. To avoid filtering excess chains, set select.excess.chains.by.umi.count to FALSE. For further notes on the implication of these please refer to the documentation of the parameter hierarchical in the function VDJ_clonotype_v3.

`trim.and.align` Boolean. Defaults to FALSE. Whether to trim VJ/VDJ seqs, align them to the 10x reference and trim the reference. This is useful to get full sequences for antibody expression or numbers of somatic hypermutations. !Setting this to TRUE significantly increases computational time

`parallel.processing`
Character string. Can be "parlapply" for Windows system, "mclapply" for unix and Mac systems or "none" to use a simple for loop (slow!). Default is "none" for compatibility reasons. For the parlapply option the packages parallel, doParallel and the dependency foreach are required

`numcores` Number of cores used for parallel processing. Defaults to number of cores available. If you want to chek how many cores are available use the library Parallel and its command detectCores() (Not setting a limit here when running this function on a cluster may cause a crash)

`gap.opening.cost`
Argument passed to Biostrings::pairwiseAlignment during alignment to reference. Defaults to 10

`gap.extension.cost`
Argument passed to Biostrings::pairwiseAlignment during alignment to reference. Defaults to 4

`exclude.on.cell.state.markers`
: Character vector. If no input is provided or input is "none", no cells are excluded. Input format should follow: Character vector containing the gene names for each state. ; is used to use multiple markers within a single gene state. Different vector elements correspond to different states. Example: c("CD4+;CD44-","CD4+;IL7R+;CD44+"). All cells which match any of the given states (in the example case any of the 2) are excluded. This is useful in case different and non lymphocyte cells were co-sequenced. It should give the option to e.g. exclude B cells in the analysis of T cells in a dataset.

`exclude.on.barcodes`
: Character vector. Provide a list of 10x barcodes WITHOUT the terminal id (-1 , -2 etc.) to exclude from GEX and VDJ prior to processing.

`integration.method`
: String specifying which data normalization and integration pipeline should be used. Default is "scale.data", which correspondings to the ScaleData function internal to harmony package. 'anchors' scales data individually and then finds and align cells in similar states as described here: https://satijalab.org/seurat/articles/integration_introduction.h 'sct'specifies SCTransform from the Seurat package. "harmony" should be specificied to perform harmony integration. This method requires the harmony package from bioconductor.

`VDJ.gene.filter`
: Logical indicating if variable genes from the b cell receprot and t cell receptor should be removed from the analysis. True is highly recommended to avoid clonal families clustering together.

`mito.filter`
: Numeric specifying which percent of genes are allowed to be composed of mitochondrial genes. This value may require visual inspection and can be specific to each sequencing experiment. Users can visualize the percentage of genes corresponding to mitochondrial genes using the function "investigate_mitochondial_genes".

`norm.scale.factor`
: Scaling factor for the standard Seurat pipeline. Default is set to 10000 as reported in Seurat documentation.

`n.feature.rna`
: Numeric that specifies which cells should be filtered out due to low number of detected genes. Default is set to 0. Seurat standard pipeline uses 2000.

`n.count.rna.min`
: Numeric that specifies which cells should be filtered out due to low RNA count.Default is set to 0. Seurat standard pipeline without VDJ information uses 200.

`n.count.rna.max`
: Numeric that specifies which cells should be filtered out due to high RNA count.Default is set to infinity. Seurat standard pipeline without VDJ information uses 2500.

`n.variable.features`
: Numeric specifying the number of variable features. Default set to 2000 as specified in Seurat standard pipeline.

`cluster.resolution`
: Numeric specifying the resolution that will be supplied to Seurat's FindClusters function. Default is set to 0.5. Increasing this number will increase the number of distinct Seurat clusters. Suggested to examine multiple parameters to ensure gene signatures differentiating clusters remains constant.

`neighbor.dim` Numeric vector specifying which dimensions should be supplied in the FindNeighbors function from Seurat. Default input is '1:10'.

`mds.dim` Numeric vector specifying which dimensions should be supplied into dimensional reduction techniques in Seurat and Harmony. Default input is '1:10'.

`FB.count.threshold`
Numeric. Defaults to 10. For description of Feature Barcode assignment see parameter FB.ratio.threshold above

`FB.ratio.threshold`
Numeric. Defaults to 2 Threshold for assignment of feature barcodes by counts. A feature barcode is assigned to a cell if its counts are >FB.count.threshold and if its counts are FB.ratio.threshold-times higher than the counts of the feature barcode with second most counts.

`FB.exclude.pattern`
Character (regex compatible). If a feature barcode matches this pattern it will be excluded from the hashing sample assignments. This may be neccessary if CITE-seq barcodes and hashing barcodes are sequenced in the same run.

`subsample.barcodes`
For development purposes only. If set to TRUE the function will run on 100 cells only to increase speeds of debugging

`verbose` if TRUE prints runtime info to console. Defaults to TRUE

Value

Single cell matrix including VDJ and GEX info. Format is a list with out[[1]] = a VDJ dataframe (or list of dataframes if VDJ.combine == FALSE, not recommended) containing also selected GEX information of integrate.GEX.to.VDJ = T. out[[2]] = GEX Seurat object with the metadata also containing GEX information if integrate.VDJ.to.GEX = T. out[[3]] = Dataframe with statistics on GEX and VDJ. out[[4]] = runtime parameters. out[[5]] = session info

Examples

```
try({
VDJ.out.directory.list <- list()
VDJ.out.directory.list[[1]] <- c("~/VDJ/S1/")
VDJ.out.directory.list[[2]] <- c("~/VDJ/S2/")
GEX.out.directory.list <- list()
GEX.out.directory.list[[1]] <- c("~/GEX/S1/")
GEX.out.directory.list[[2]] <- c("~/GEX/S2/")
VGM <- VDJ_GEX_matrix(
VDJ.out.directory.list = VDJ.out.directory.list
,GEX.out.directory.list = GEX.out.directory.list
,GEX.integrate = T
,VDJ.combine = T
,integrate.GEX.to.VDJ = T
,integrate.VDJ.to.GEX = T
,exclude.GEX.not.in.VDJ = F
,filter.overlapping.barcodes.GEX = F
,filter.overlapping.barcodes.VDJ = F
,get.VDJ.stats = T
```

```
,parallel.processing = "none"
,subsample.barcodes = F
,trim.and.align = F
,group.id = c(1,2))
})
```

VDJ_GEX_overlay_clones

Overlay clones on GEX projection

Description

Highlights the cells belonging to any number of top clonotypes or of specifically selected clonotypes from one or more samples or groups in a GEX dimensional reduction.

Usage

```
VDJ_GEX_overlay_clones(
  GEX,
  reduction,
  n.clones,
  clones.to.plot,
  by.sample,
  by.other.group,
  ncol.facet,
  pt.size,
  clone.colors,
  others.color,
  split.plot.and.legend,
  platypus.version
)
```

Arguments

GEX	A single seurat object from VDJ_GEX_matrix, which also includes VDJ information in the metadata (set integrate.VDJ.to.GEX to TRUE in the VDJ_GEX_matrix function) (VDJ_GEX_matrix.output[[2]]) ! Clone ids and frequencies are drawn from the columns "clonotype_id" and "clonotype_frequency"
reduction	Character. Defaults to "umap". Name of the reduction to overlay clones on. Can be "pca", "umap", "tsne"
n.clones	Integer. Defaults to 5. To PLOT TOP N CLONES. Number of Top clones to plot. If either by.sample or by.group is TRUE, n.clones clones from each sample or group will be overlayed

`clones.to.plot` Character. Alternative to n.clones. TO PLOT SPECIFIC CLONES. Must reference a column in the GEX@meta.data filled with TRUE and FALSE. Entries with TRUE label are plotted. Such a column may be generated using GEX@metadata$clones_to_plot_column <- GEX@metadata$Some_cell_identifier == "Interesting"

`by.sample` Boolean. Defaults to FALSE. Whether to overlay clones by sample. If set to TRUE this will generate a facet_wrap plot with as many facets as samples.

`by.other.group` Character string. Defaults to "none". Must be a valid column name of the metadata of the input seurat object. If so, this will generate a facet_wrap plot with as many facets unique entries in the specified column. This may be useful to plot cell type specific clones

`ncol.facet` Integer. Defaults to 2. Number of columns in the facet_wrap plot if by.sample or by.group is TRUE

`pt.size` Numeric. Defaults to 1. Size of points in DimPlot. Passed to Seurat::DimPlot

`clone.colors` Character vector. Defaults to rainbow(n.clones). Colors to use for individual clones. One can provide either a vector of length n.clones or a of length Nr. of samples/groups * n.clones. In case that a vector of length n.clones is provided and by.group or by.sample is TRUE, colors are repeated for each sample/group

`others.color` Character. Color for cells that are not selected i.e. not part of the overlayed clonotypes. Defaults to "grey80". To hide the rest of the umap set to "white"

`split.plot.and.legend` Boolean. Defaults to FALSE. Whether to return the plot and the legend separately as a list. This can be useful if legends get large and distort the actual plots. The packages gridExtra and cowplot are required for this. If set to TRUE a list is returned where out[[1]] is the plot which can be printed just by executing out[[1]]; out[[2]] is the legend, which can be printed either using plot(out[[2]]) or grid.arrange(out[[2]])

`platypus.version` Character. At the moment this function runs only on the output of the VDJ_GEX_matrix function meaning that it is exclusively part of Platypus "v3". With further updates the functionality will be extended.

Value

A ggplot object or a list of a ggplot and a gtable legend (if split.plot.and.legend \=\= TRUE). Theme, colors etc. may be changed directly by adding new elements to this output (e.g. out \+ theme_minimal())

Examples

```
overlay_clones_plot <- VDJ_GEX_overlay_clones(
GEX = Platypus::small_vgm[[2]], reduction = "umap"
,n.clones = 5, by.sample = FALSE
,by.other.group = "none", pt.size = 1,split.plot.and.legend = FALSE)

overlay_clones_plot <- VDJ_GEX_overlay_clones(
GEX = Platypus::small_vgm[[2]], reduction = "umap"
```

```
,n.clones = 5, by.sample = TRUE, by.other.group = "none"
,pt.size = 1,ncol.facet = 2, split.plot.and.legend = FALSE)

overlay_clones_plot <- VDJ_GEX_overlay_clones(
GEX = Platypus::small_vgm[[2]], reduction = "umap"
,n.clones = 5, by.sample = TRUE, by.other.group = "group_id", pt.size = 1
,ncol.facet = 2, split.plot.and.legend = TRUE)
```

`VDJ_GEX_stats` *Standalone VDJ and GEX statistics.*

Description

Gives stats on number and quality of reads. This function is integrated into the VDJ_GEX_matrix. Before running, please check list element [[3]] of VDJ_GEX_matrix output for already generated statistics.

Usage

```
VDJ_GEX_stats(
  VDJ.out.directory,
  GEX.out.directory,
  sample.names,
  metrics10x,
  save.csv,
  filename
)
```

Arguments

`VDJ.out.directory`	List of paths with each element containing the path to the output of cellranger VDJ runs. This pipeline assumes that the output file names have not been changed from the default 10x settings in the /outs/ folder. This is compatible with B and T cell repertoires (both separately and simultaneously).
`GEX.out.directory`	OPTIONAL list of paths with each element containing the path to the output of cellranger GEX runs. This pipeline assumes that the output file names have not been changed from the default 10x settings in the /outs/ folder. This is compatible with B and T cell repertoires (both separately and simultaneously).
`sample.names`	OPTIONAL: an array of the same length as the input VDJ.out.directory list with custom names for each sample. If not provided samples will be numbered by processing order
`metrics10x`	Whether to append metrics_summary.csv information provided by Cellranger for both VDJ and GEX. Defaults to T
`save.csv`	Boolean. Defaults to TRUE. Whether to directly save the resuts as a comma delimited .csv file in the current working directory.
`filename`	Character ending in .csv. Filename to save .csv as.

Value

returns a single matrix where the rows are individual cells and the columns are repertoire features.

Examples

```
try({
stats <- VDJ_GEX_stats(VDJ.out.directory = VDJ.out.directory.list
,GEX.out.directory = GEX.out.directory.list,sample.names = c(1:4)
,metrics10x = TRUE,save.csv = FALSE)
})
```

VDJ_kmers *Calculates and plots kmers distributions and frequencies.*

Description

Calculates and plots kmers distributions and frequencies.

Usage

```
VDJ_kmers(
  VDJ,
  sequence.column,
  grouping.column,
  pool.per.group,
  kmer.k,
  max.kmers,
  specific.kmers,
  plot.format,
  as.proportions
)
```

Arguments

VDJ	VDJ dataframe output from the VDJ_GEX_matrix function.
sequence.column	Character vector. One or more sequence column names from the VDJ for kmer counting. if more than one column is provided (e.g. c("VDJ_cdr3s_aa","VJ_cdr3s_aa")) these columns will be pasted together before counting the kmers.
grouping.column	Character. Column name of a column to group kmer counting by. This could be "sample_id" to group each kmer by the sample.
pool.per.group	Boolean. If TRUE, will sum the kmer counts of each sequence per grouping factor (determined in grouping.column).
kmer.k	Integer. Length k of each kmer.

max.kmers	Integer. Maximum number of kmers to be plotted in the output barplots.
specific.kmers	Character vector. Specific kmers to be plotted in the output barplots.
plot.format	Character. The output plot format: 'barplot' for barplots of kmer frequency per group, 'pca' for group-level PCA reduction across the kmer vectors, 'density' for kmer count density plots.
as.proportions	Boolean. If TRUE, will return the kmer barplot as proportions instead of absolute counts.

Value

Returns a ggplot with the kmer analysis depedning on the plot.format parameter

Examples

```
try({
 VDJ_kmers(VDJ = Platypus::small_vgm[[1]],
 sequence.column = c("VDJ_cdr3s_aa"), grouping.column = "sample_id", kmer.k = 2, max.kmers = 5)
 })
```

VDJ_logoplot_vector *Flexible logoplot wrapper*

Description

Plots a logoplot of the CDR3 aminoacid region

Usage

```
VDJ_logoplot_vector(
  cdr3.vector,
  length_cdr3,
  seq_type,
  namespace,
  font,
  method
)
```

Arguments

cdr3.vector	A character vector of aa sequences. This is to increase flexibility of this function. Such a sequence vector may be retrieved from the VDJ_analyse function output on a clonotype level or from the VDJ_GEX_matrix function output on a per cell level. Additionally, any length of sequence may be used (e.g. HCDR3 only or H and LCDR3 pasted together)

length_cdr3	Integer or character. Defaults to "auto". Sets the length of the CDR3 regions that are selected to be plotted. If set to auto, the most frequently appearing length in the vector will be used
seq_type	passed to ggseqlogo. Can be set to "aa", "dna", "rna" or "other"
namespace	passed to ggseqlogo. Default "auto".
font	passed to ggseqlogo. Default "roboto_medium".
method	passed to ggseqlogo. Default "bits".

Value

Returns the logo plot.

Examples

```
VDJ_logoplot_vector(
cdr3.vector = Platypus::small_vgm[[1]]$VDJ_cdr3s_aa
,length_cdr3 = "auto",seq_type = "auto")
```

VDJ_network *Similarity networks based on CDR3 regions*

Description

Creates a similarity network where clones with similar CDR3s are connected.

Usage

```
VDJ_network(
  VDJ,
  distance.cutoff,
  per.sample,
  platypus.version,
  known.binders,
  hcdr3.only,
  is.bulk
)
```

Arguments

VDJ	Either (for platypus version "v2") output from VDJ_analyze function. This should be a list of clonotype dataframes, with each list element corresponding to a single VDJ repertoire, OR (for platypus version "v3") the the VDJ matrix output of the VDJ_GEX_matrix() function (VDJ.GEX.matrix.output[[1]])
distance.cutoff	The threshold Levenshtein distance for which two nodes will be connected on the similarity network.

per.sample	logical value indicating if a single networks should be produced for each mouse.
platypus.version	Character. Defaults to "v3". Can be "v2" or "v3" dependent on the input format
known.binders	Either a character vector with cdr3s of known binders or a data frame with cdr3s in the first and the corresponding specificity in the second column. If this parameter is defined, the output will be a network with only edges between known binders and the repertoire nodes and edges between the known binders that have at least one edge to a repertoire node
hcdr3.only	logical value indicating if the network is based on heavy chain cdr3s (hcdr3.only = T) or pasted heavy and light chain cdr3s (hcdr3.only = F), works for platypus.version 3 only
is.bulk	logical value indicating whether the VDJ input was generated from bulk-sequencing data using the bulk_to_vgm function. If is.bulk = T, the VDJ_network function is compatible for use with bulk data. Defaults to False (F).

Value

returns a list containing networks and network information. If per.sample is set to TRUE then the result will be a network for each repertoire. If per.sample ==FALSE, output[[1]] <- will contain the network, output[[2]] will contain the dataframe with information on each node, such as frequency, mouse origin etc. output[[3]] will contain the connected index - these numbers indicate that the nodes are connected to at least one other node. output[[4]] contains the paired graph - so the graph where only the connected nodes are drawn.

Examples

```
try({
network_out <- VDJ_network(VDJ = Platypus::small_vgm[[1]],per.sample = FALSE,distance.cutoff = 2)
})
```

`VDJ_ordination` *Performs ordination/dimensionality reduction for a species incidence matrix, depending on the species selected in the feature.columns parameter.*

Description

Performs ordination/dimensionality reduction for a species incidence matrix, depending on the species selected in the feature.columns parameter.

Usage

```
VDJ_ordination(
  VDJ,
  feature.columns,
  grouping.column,
```

```
  method,
  reduction.level,
  VDJ.VJ.1chain,
  umap.n.neighbours,
  tsne.perplexity
)
```

Arguments

VDJ	VDJ dataframe output from the VDJ_build function.
feature.columns	Character vector. One or more column names from the VDJ to indicate the unique species for the incidence/count matrix. if more than one column is provided (e.g. c("VDJ_cdr3_aa","VJ_cdr3_aa")) these columns will be pasted together before metric calculation.
grouping.column	Character. Column name of a column to group the ordination by. This could be "sample_id" to reduce across each sample. Indicative of 'sites' in a typical community data matrix/incidence matrix used in community ecology analyses (species by sites).
method	Character. The ordination method; choose from either: PCA - 'pca', t-SNE - 'tsne', UMAP - 'umap', PCOA/MDS - 'mds', DCA - 'dca'.
reduction.level	Character. Whether to reduce across groups ('groups'), features/sequences ('features'), or both ('both').
VDJ.VJ.1chain	Boolean defaults to TRUE. Whether to filter out aberrant cells (more than 1 VDJ or VJ chain).
umap.n.neighbours	Integer. Control the t-SNE perplexity when method = 'tsne'.
tsne.perplexity	Integrer. Defaults to 1

Value

Returns a ggplot with the ordination analysis performer across features, groups, or both

Examples

```
plot <- VDJ_ordination(VDJ = Platypus::small_vdj
,feature.columns = c("VDJ_cdr3_aa"), grouping.column = "sample_id"
,method = "pca", reduction.level = 'groups')
```

VDJ_overlap_heatmap *Wrapper to determine and plot overlap between VDJ features across groups*

Description

Yields overlap heatmap and datatable of features or combined features for different samples or groups

Usage

```
VDJ_overlap_heatmap(
  VDJ,
  feature.columns,
  grouping.column,
  jaccard,
  plot.type,
  pvalues.label.size,
  axis.label.size,
  add.barcode.table
)
```

Arguments

VDJ	VDJ output of the VDJ_GEX_matrix function (VDJ_GEX_matrix.output[[1]])
feature.columns	A character array of column names of which the overlap should be displayed. The content of these columns is pasted together (separated by "/"). E.g. if the overlap in cells germline gene usage is desired, the input could be c("VDJ_jgene","VDJ_dgene","VDJ_vg These columns would be pasted and compared across the grouping variable.
grouping.column	A column which acts as a grouping variable. If repertoires are to be compared use the sample_id column.
jaccard	Boolean. Defaults to FALSE. If set to TRUE, the overlap will be reported as jaccard index. If set to FALSE the overlap will be repored as absolute counts
plot.type	Character. Either "ggplot" or "pheatmap". Defaults to Pheatmap
pvalues.label.size	Numeric. Defaults to 4. Is passed on to ggplot theme
axis.label.size	Numeric. Defaults to 4. Is passed on to ggplot theme
add.barcode.table	Boolean. Defaults to T. Whether to generate a dataframe with frequencies and barcodes of cells with overlapping features. This is useful to e.g. analyze deferentially expressed genes between cells of two samples or groups expressing the same VDJ or VJ chain

Value

A list of a ggplot (out[[1]]), the source table or matrix for the plot out[[2]] and a table containing additional information in case that add.barcode.table was set to TRUE (out[[3]])

Examples

```
overlap <- VDJ_overlap_heatmap(VDJ = Platypus::small_vgm[[1]]
,feature.columns = c("VDJ_cdr3s_aa"),
grouping.column = "sample_id", axis.label.size = 15
,plot.type = "ggplot")
```

VDJ_phylogenetic_trees

Creates phylogenetic trees from a VDJ dataframe

Description

Creates phylogenetic trees as tidytree dataframes from an input VDJ dataframe. The resulting phylogenetic trees can be plotted using VDJ_phylogenetic_trees_plot. Both of these functions require the tidytree and ggtree packages.

Usage

```
VDJ_phylogenetic_trees(
  VDJ,
  sequence.type,
  as.nucleotide,
  trimmed,
  include.germline,
  global.clonotype,
  VDJ.VJ.1chain,
  additional.feature.columns,
  filter.na.columns,
  maximum.lineages,
  minimum.sequences,
  maximum.sequences,
  tree.algorithm,
  tree.level,
  n.trees.combined,
  germline.scale.factor,
  output.format,
  parallel
)
```

Arguments

VDJ	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
sequence.type	string - sequences which will be used when creating the phylogenetic trees. 'cdr3' for CDR3s of both VDJs and VJs, 'cdrh3' for VDJ CDR3s, 'VDJ.VJ' for pasted full sequences of both VDJ and VJ, 'VDJ' for full VDJ sequences, 'VJ' for full VJ.
as.nucleotide	boolean - if T, will only consider the DNA sequences specified by sequence.type, else it will consider the amino acid ones.
trimmed	boolean - in the case of full VDJ or VJ nt sequences, if the trimmed sequences should be consider (trimmed=T), or raw ones. You need to call MIXCR first on the VDJ dataframe using VDJ_call_MIXCR().
include.germline	boolean - if T, a germline sequence will be included in the trees (root), obtained by pasting the VDJ_trimmed_ref and VJ_trimmed_ref sequences. You need to call MIXCR first on the VDJ dataframe using VDJ_call_MIXCR().
global.clonotype	boolean - if T, will ignore samples from the sample_id column, creating global clonotypes.
VDJ.VJ.1chain	boolean - if T, will remove aberrant cells from the VDJ matrix.
additional.feature.columns	list of strings or NULL - VDJ column names which will comprise the per-sequence features to be included in the tidytree dataframe, which will be used to label nodes/ determines their color/ size etc. See also the VDJ_phylogenetic_trees_plot function.
filter.na.columns	list of strings - VDJ columns names: if a phylogenetic tree/tidytree dataframe has all elements = NA in that feature, that tree will be completely removed.
maximum.lineages	integer or 'all' - maximum number of clonotypes to create trees for. If 'all', will create trees for all clonotypes.
minimum.sequences	integer - lower bound of sequences for a tree. Defaults to 3. Trees with a lower number will be automatically removed.
maximum.sequences	integer - upper bound of sequences for a tree. Additional sequences will be removed, after being ordered by their total frequency.
tree.algorithm	string - the algorithm used when constructing the phylogenetic trees. 'nj' for Neighbour-Joining, 'bionj', 'fastme.bal', and 'fastme.ols'
tree.level	string - level at which to build phylogenetic trees. 'intraclonal' - tree per clonotype, per sample, 'global.clonotype' - global clonotype trees (include.germline must be F), irrespective of sample, 'combine.first.trees' will combine the trees for the most expanded clonotypes, per sample (include.germline must be F).
n.trees.combined	integer - number of trees to combine if tree.level='combine.first.trees'.

germline.scale.factor
: numeric - as germlines are incredibly distant from their closest neighbours (in the tree), this controls the scale factor for the germline tree branch length for more intelligible downstream plotting.

output.format
: string - 'tree.df.list' returns a nested list of tidytree dataframes, per clonotype and per sample; 'lineage.df.list' returns a list of lineage dataframes - unique sequences per clonotype,

parallel
: string - parallelization method to be used to accelerate computations, 'none', 'mclapply', or 'parlapply'.

Value

Nested list of tidytree dataframes or lineage dataframes.

Examples

```
try({
VDJ_phylogenetic_trees(VDJ=Platypus::small_vgm[[1]], sequence.type='VDJ.VJ',
trimmed=TRUE, as.nucleotide=TRUE, include.germline=TRUE,
additional.feature.columns=NULL, tree.level='intraclonal',
output.format='tree.df.list')
})
```

VDJ_phylogenetic_trees_plot

Phylogenetic tree plotting

Description

Function to plot phylogenetic trees obtained from VDJ_phylogenetic_trees

!Requires the ggtree package to be loaded! Plots trees from function VDJ_phylogenetic_trees

Usage

```
VDJ_phylogenetic_trees_plot(
  tree.dfs,
  color.by,
  size.by,
  shape.by,
  specific.leaf.colors,
  specific.leaf.shapes
)
```

Arguments

tree.dfs	nested list of tidytree dataframes obtained from VDJ_phylogenetic_trees with output.format='tree.df.list'. tree.dfs[[1]][[2]] represent a tree dataframe for the first sample, second clonotype.
color.by	string - VDJ or tree df column name which will be used to color the tree nodes.
size.by	string or NULL - VDJ or tree df column name which determines the node size. If NULL, node sizes will be equal.
shape.by	string or NULL - VDJ or tree df column name which determines the node shape. If NULL, node sizes will be equal.
specific.leaf.colors	named list or NULL - if NULL, colors will be automatically selected for each node according to its color.by value.
specific.leaf.shapes	named list or NULL - if NULL, shapes will be automatically selected for each node according to its shape.by value.

Value

nested list of ggtree plot objects for each sample and each clonotype.

Examples

```
try({
tree.dfs <- VDJ_phylogenetic_trees(VDJ=Platypus::small_vgm[[1]], sequence.type='VDJ.VJ',
trimmed=TRUE, as.nucleotide=TRUE, include.germline=TRUE,
additional.feature.columns=NULL, tree.level='intraclonal',
output.format='tree.df.list')
VDJ_phylogenetic_trees_plot(tree.dfs,color.by='clonotype_id', size.by='sequence_frequency')
})
```

VDJ_plot_SHM *Plotting of somatic hypermutation counts*

Description

Plots for SHM based on MIXCR output generated using the VDJ_call_MIXCR function and appended to the VDJ.GEX.matrix.output

Usage

```
VDJ_plot_SHM(
  VDJ.mixcr.matrix,
  group.by,
  quantile.label,
  point.size,
```

```
  mean.line.color,
  stats.to.console,
  platypus.version
)
```

Arguments

VDJ.mixcr.matrix	Output dataframe from the VDJ_call_MIXCR function or a dataframe generated using the VDJ_GEX_matrix function and supplemented with MIXCR information
group.by	Character. Defaults to "sample_id". Column name of VDJ.matrix to split VDJ.matrix by. For each unique entry in that column a set of plots will be generated. This can be useful to plot SHM by expansion or by transcriptomics-derived clusters
quantile.label	Numeric. Defaults to 0.9. Which points to label in the SHM scatterplot. If set to 0.9, the top 10% of cells by SHM number will be labelled. If ggrepel throws a warning, concerning overlap it is recommended to attempt to lable less points to avoid cluttering
point.size	Size of points in plots. Passed to geom_jitter()
mean.line.color	Color of mean bar in dotplots. Passed to geom_errorbar()
stats.to.console	Boolean. Defaults to FALSE. Prints basic statistics (AOV \+ post hoc test) to console
platypus.version	Character. Only "v3" available.

Value

Returns a list of ggplot objects. out\[\[1\]\] is a boxplot comparing SHM by group.by. out\[\[2\]\] to out\[\[n\]\] are plots for each group that visualize VDJ and VJ SHM distribution for each group. Data for any plot can be accessed via out \[\[any\]\]$data

Examples

```
small_vgm <- Platypus::small_vgm
small_vgm[[1]]$VDJ_SHM <- as.integer(rnorm(nrow(small_vgm[[1]]), mean = 5, sd = 3))
small_vgm[[1]]$VJ_SHM <- as.integer(rnorm(nrow(small_vgm[[1]]), mean = 5, sd = 3))

SHM_plots <- VDJ_plot_SHM(VDJ = small_vgm[[1]]
, group.by = "sample_id", quantile.label = 0.9)

SHM_plots <- VDJ_plot_SHM(VDJ = small_vgm[[1]]
, group.by = "seurat_clusters", quantile.label = 0.99)
```

VDJ_public *Function to get shared/public elements across multiple repertoires*

Description

Function to get shared elements across multiple repertoires, specified by the feature.columns parameter (a column of the VDJ matrix). If two columns are specified in feature.columns, the resulting shared features will combine the values from each column (at a per-cell level).

Usage

```
VDJ_public(
  VDJ,
  feature.columns,
  grouping.column,
  specific.groups,
  find.public.all,
  find.public.percentage,
  treat.combined.features,
  output.format
)
```

Arguments

VDJ	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
feature.columns	Character or character vector columns of features to be assayed
grouping.column	string - the repertoire/group-defining column (default to 'sample_id').
specific.groups	vector of strings or NULL - if only the shared elements from specific repertoires should be taken into account. If NULL, will output the shared/public elements across all repertoires.
find.public.all	boolean - if T, will look for the public elements across all repertoires
find.public.percentage	list - the first element denotes the percentage of repertoires to get shared elements for, the second element is the maximum number of repertoire combinations to consider (can be NULL to consider all).
treat.combined.features	string - 'exclude' will exclude combined features with one element missing, 'include' will include and considers them as a new feature value.
output.format	string - 'df' to get a shared element dataframe (with columns = Repertoire and Public), 'list' for a list of shared elements.

Value

Either a dataframe of public elements across multiple repertoires or a list.

Examples

```
VDJ_get_public(VDJ = small_vgm[[1]],
feature.columns='VDJ_cdr3s_aa', find.public.all=TRUE,
output.format='df')
```

`VDJ_rarefaction` *Plots rarefaction curves for species denoted in the feature.columns parameter across groups determined by grouping.columns*

Description

Plots rarefaction curves for species denoted in the feature.columns parameter across groups determined by grouping.columns

Usage

```
VDJ_rarefaction(
  VDJ,
  feature.columns,
  grouping.column,
  VDJ.VJ.1chain,
  rarefaction.type,
  hill.numbers,
  number.resamples,
  sample.sizes,
  endpoint
)
```

Arguments

`VDJ`	VDJ dataframe output from the VDJ_GEX_matrix function.
`feature.columns`	Character vector. One or more column names from the VDJ to indicate the unique species for the rarefaction (to rarefy across). If more than one column is provided (e.g. c("VDJ_cdr3s_aa","VJ_cdr3s_aa")) these columns will be pasted together.
`grouping.column`	Character. Column name of a column to group the rarefaction by. This could be "sample_id" for rarefaction curves for each sample.
`VDJ.VJ.1chain`	Boolean defaults to TRUE. Whether to filter out aberrant cells (more than 1 VDJ or VJ chain).

rarefaction.type
: Character. Options for the iNEXT rarefaction - 'sample.size','coverage.based', or 'sample.completeness'.

hill.numbers
: Integer/ vector of integers. The Hill numbers to be plotted out (0 - species richness, 1 - Shannon diversity, 2 - Simpson diversity)

number.resamples
: Integer. Number of bootstrap replications.

sample.sizes
: Vector if integers. The sample size points at which rarefaction should be performed. Defaults to NULL

endpoint
: Integer. The maximum sample size for rarefaction extrapolation. Defaults to NULL = 2 times the sample size for each sample.

Value

Returns a ggplot with the ordination analysis performer across features, groups, or both

Examples

```
try({
plot <- VDJ_diversity(VDJ = Platypus::small_vgm[[1]],
,feature.columns = c("VDJ_cdr3s_aa"), grouping.column = "sample_id")
})
```

VDJ_variants_per_clone

Wrapper for variant analysis by clone

Description

Returns statistics and plots to examine diversity of any sequence or metadata item within clones on a by sample level or global level

Usage

```
VDJ_variants_per_clone(
  VDJ,
  variants.of,
  clonotypes.col,
  stringDist.method,
  split.by,
  platypus.version
)
```

Arguments

VDJ	VDJ output of the VDJ_GEX_matrix (VDJ_GEX_matrix.output[[1]]). VDJ matrix supplemented with with MIXCR information is also valid
variants.of	Character vector. Defaults to c("VDJ_cdr3s_aa", "VJ_cdr3s_aa"). Column name(s) of VDJ to examine variants of. If more than one name is given, these columns will be pasted together. The default will therefore return statistics on the number of variants of VDJ and VJ cdr3s in every clone
clonotypes.col	Column name of the VDJ column containing clonotype information. Defaults to "clonotype_id_10x". This is useful if alternative clonotyping strategies have been used and are stored in other columns
stringDist.method	Character. Passed to Biostrings::strinDist. Method to calculate distance between variants of a clone. Defaults to "levenshtein". Other options are "hamming", "quality". If "hamming" variants of a clone will be shortened from the end to the shortest variant to make all input sequences the same length.
split.by	Character. Defaults to "sample_id". Column name of VDJ to split the analysis by. This is necessary, if clonotyping was done on a per sample level (e.g. "clonotype1" in sample 1 is not the same sequence as "clonotype1" in sample 2). If clonotyping was done across samples and no splitting is necessary input "none"
platypus.version	Character. Only "v3" available.

Value

Returns a list of dataframes. Each dataframe contains the statistics of one split.by element (by default: one sample)

Examples

```
variants_per_clone <- VDJ_variants_per_clone(VDJ = Platypus::small_vgm[[1]]
,variants.of = c("VDJ_cdr3s_aa", "VJ_cdr3s_aa"),
stringDist.method = "levenshtein", split.by = "sample_id")
```

VDJ_Vgene_usage *V(D)J gene usage stacked barplots*

Description

Produces a matrix counting the number of occurences for each VDJ and VJ Vgene combinations for each list enty in VDJ.clonotype.output or for each sample_id in VDJ.matrix

Usage

```
VDJ_Vgene_usage(VDJ, group.by, platypus.version)
```

Arguments

VDJ	For platypus.version = "v2" output from VDJ_analyze function. This should be a list of clonotype dataframes, with each list element corresponding to a single VDJ repertoire. For platypus.version = "v3" output VDJ dataframe from VDJ_GEX_matrix function (VDJ_GEX_matrix.output[[1]])
group.by	Character. Defaults to "sample_id". Column name of VDJ to group plot by.
platypus.version	Character. Defaults to "v3". Can be "v2" or "v3" dependent on the input format

Value

Returns a list of matrices containing the number of Vgene heavy/light chain combinations per repertoire.

Examples

```
example.vdj.vgene_usage <- VDJ_Vgene_usage(VDJ =
Platypus::small_vgm[[1]], platypus.version = "v3")
```

VDJ_Vgene_usage_barplot

V(D)J gene usage barplots

Description

Produces a barplot with the most frequently used IgH and IgK/L Vgenes.

Usage

```
VDJ_Vgene_usage_barplot(
  VDJ,
  group.by,
  HC.gene.number,
  LC.Vgene,
  LC.gene.number,
  platypus.version,
  is.bulk
)
```

Arguments

VDJ	Either (for platypus version "v2") output from VDJ_analyze function. This should be a list of clonotype dataframes, with each list element corresponding to a single VDJ repertoire, OR (for platypus version "v3") the the VDJ matrix output of the VDJ_build() function.

group.by	Character. Defaults to "sample_id". Column name of VDJ to group plot by.
HC.gene.number	Numeric value indicating the top genes to be dispayed. If this number is higher than the total number of unique HC V genes in the VDJ repertoire, then this number is equal to the number of unique HC V genes.
LC.Vgene	Logical indicating whether to make a barplot of the LC V genes distribution. Default is set to FALSE.
LC.gene.number	Numeric value indicating the top genes to be dispayed. If this number is higher than the total number of unique LC V genes in the VDJ repertoire, then this number is equal to the number of unique LC V genes.
platypus.version	Character. Defaults to "v3". Can be "v2" or "v3" dependent on the input format
is.bulk	logical value indicating whether the VDJ input was generated from bulk-sequencing data using the bulk_to_vgm function. If is.bulk = T, the VDJ_Vgene_usage_barplot function is compatible for use with bulk data. Defaults to False (F).

Value

Returns a list of ggplot objects which show the distribution of IgH and IgK/L V genes for the most used V genes.

Examples

```
VDJ_Vgene_usage_barplot(VDJ = Platypus::small_vdj,
HC.gene.number = 2, platypus.version = "v3")
```

VDJ_Vgene_usage_stacked_barplot

V(D)J gene usage stacked barplots

Description

Produces a stacked barplot with the fraction of the most frequently used IgH and IgK/L Vgenes. This function can be used in combination with the VDJ_Vgene_usage_barplot to vizualize V gene usage per sample and among samples.

Usage

```
VDJ_Vgene_usage_stacked_barplot(
  VDJ,
  group.by,
  HC.gene.number,
  Fraction.HC,
  LC.Vgene,
  LC.gene.number,
  Fraction.LC,
```

```
  platypus.version,
  is.bulk
)
```

Arguments

VDJ	Either (for platypus version "v2") output from VDJ_analyze function. This should be a list of clonotype dataframes, with each list element corresponding to a single VDJ repertoire, OR (for platypus version "v3") the the VDJ matrix output of the VDJ_GEX_matrix() function (normally VDJ.GEX.matrix.output[[1]])
group.by	Character. Defaults to "sample_id". Column name of VDJ to group plot by.
HC.gene.number	Numeric value indicating the top genes to be dispayed. If this number is higher than the total number of unique HC V genes in the VDJ repertoire, then this number is equal to the number of unique HC V genes.
Fraction.HC	Numeric value indicating the minimum fraction of clones expressing a particular HC V gene. If the usage of a particular gene is below this value, then this gene is excluded. If the usage of a particular gene is above this value even in one sample, then this gene is included in the analysis. Default value is set to 0, thus all genes are selected.
LC.Vgene	Logical indicating whether to make a barplot of the LC V gene distribution. Default is set to FALSE.
LC.gene.number	Numeric value indicating the top genes to be dispayed. If this number is higher than the total number of unique LC V genes in the VDJ repertoire, then this number is equal to the number of unique LC V genes.
Fraction.LC	Numeric value indicating the minimum fraction of clones expressing a particular LC V gene. If the usage of a particular gene is below this value, then this gene is excluded. If the usage of a particular gene is above this value even in one sample, then this gene is included in the analysis. Default value is set to 0, thus all genes are selected.
platypus.version	Set according to input format to either "v2" or "v3". Defaults to "v3"
is.bulk	logical value indicating whether the VDJ input was generated from bulk-sequencing data using the bulk_to_vgm function. If is.bulk = T, the VDJ_Vgene_usage_stacked_barplot function is compatible for use with bulk data. Defaults to False (F).

Value

Returns a list of ggplot objects which show the stacked distribution of IgH and IgK/L V genes for the most used V genes. Returns an empty plot if the Fraction.HC or Fraction.LC that were selected were too high, resulting in the exclusion of all the genes.

Examples

```
example.vdj.vgene_usage <- VDJ_Vgene_usage_stacked_barplot(
VDJ = Platypus::small_vdj, LC.Vgene = TRUE
,HC.gene.number = 15, Fraction.HC = 1, platypus.version = "v3")
```

VDJ_VJ_usage_circos *Makes a Circos plot from the VDJ_analyze output. Connects the V gene with the corresponding J gene for each clonotype.*

Description

Makes a Circos plot from the VDJ_analyze output. Connects the V gene with the corresponding J gene for each clonotype.

Usage

```
VDJ_VJ_usage_circos(
  VGM,
  VDJ.or.VJ,
  label.threshold,
  cell.level,
  c.threshold,
  clonotype.per.gene.threshold,
  c.count.label,
  c.count.label.size,
  platypus.version,
  filter1H1L,
  gene.label,
  gene.label.size,
  arr.col,
  arr.direction,
  topX,
  platy.theme,
  clonotype.column
)
```

Arguments

VGM	The output of the VDJ_build function has to be supplied. For Platypus v2: The output of the VDJ_GEX_integrate function (Platypus platypus.version v2). A list of data frames for each sample containing the clonotype information and cluster membership information.
VDJ.or.VJ	Determines whether to plot the V J gene pairing of the alpha or beta chain. "VDJ", "VJ" or "both" as possible inputs. Default: "both".
label.threshold	Genes are only labeled if the count is larger then the label.threshold. By default all label.threshold = 0 (all genes are labeled).
cell.level	Logical, defines whether weight of connection should be based on number of clonotypes of number of cells. Default: number of clonotypes.
c.threshold	Only clonotypes are considered with a frequency higher then c.threshold. Allows to filter for only highly expanded clonotypes.

`clonotype.per.gene.threshold`
: How many clonotypes are required to plot a sector for a gene. Filters the rows and colums of the final adjacency matrix.

`c.count.label`
: Boolean, lets the user decide if the gene and count labels should be plotted or not. Default = T.

`c.count.label.size`
: Determines the font size of the gene labels. By default the font size for count labels is 0.6.

`platypus.version`
: Which platypus.version of platypus is being used. Default = v3. Set to v3 if VDJ_GEX_matrix.output[[1]] is used

`filter1H1L`
: Whether to filter the input VGM in "v3" to only include cells with 1 VDJ and 1 VJ chain. Defaults to TRUE

`gene.label`
: Boolean, lets the user decide if the gene labels should be plotted or not.

`gene.label.size`
: Determines the font size of the gene labels. By default the labelsize is automatically adjusted to 0.7 for labels with two or less digits, 0.6 for labels between 2 and 6 digits, and 0.4 for all longer labels. A manually defined font size will be the same for all labels!

`arr.col`
: Data.frame with three columns where the first two indicate the names of genes, clonotypes or clusters to be connected, and the third corresponds to the color of the arrow. Default set to data.frame(c("dummy.clonotype"), c("dummy.cluster"), c("dummy.color")), so no arrow is drawn.

`arr.direction`
: Either 1 or -1 and determines the direction of the arrow. Default=1.

`topX`
: Filters for the top X clonotypes and only plots the respective gene combinations or cluster memberships.

`platy.theme`
: Allows plotting in the new "pretty" theme or the older "spiky" theme without group labels and radial arrangement of gene.labels. Default = "pretty".

`clonotype.column`
: Which column in VGM contains the clonotyping information? Default="clonotype_id_10X".

Value

Returns a circos plot and a list object with the following elememts for N samples: [[1 to N]] The first N listelements corresponds to the recorded circos plots for N beeing the number or samples in the VGM. Since Circlize uses the R base plotting funciton, this is not a ggplot object but can still be replotted by calling the first list element. [[N+1]] Adjacency matrix forwarded to VDJ_circos(). This Matrix contains the counts and can be used for manual replotting using VDJ_circos directly. [[N+2]] Contains a named list with colors for each connection drawn and can be used for manual replotting using VDJ_circos directly. [[N+3]] Contains a named list with grouping information and can be used for manual replotting using VDJ_circos directly.

Examples

```
try({usage_circos_VDJVJ <- VDJ_VJ_usage_circos(Platypus::small_vdj)
usage_circos_VDJVJ[[1]]
})
```

VGM_build *Function to obtain the VGM object by integrating the VDJ and GEX/Seurat objects*

Description

Takes as input a VDJ data frame (as obtained from the VDJ function in Platypus) and a Seurat object. Outputs an integrated VGM object (a list with the first element - the VDJ object; second element - the Seurat object). Integration involves matching by barcodes and adding all features from one object to the other and vice-versa. Authors: Tudor-Stefan Cotet, Victor Kreiner

Usage

```
VGM_build(VDJ, GEX, merge.by, additional.dataframes, columns.to.transfer)
```

Arguments

VDJ	VDJ data frame, obtained from the Platypus VDJ() function
GEX	Seurat object for the single-cell immune receptor repertoire analysis.
merge.by	string - the column name to match both objects/dataframes by. Should be present in both objects (Seurat object meta.data and VDJ dataframe). Defaults to "barcode."
additional.dataframes	vector of data frames - additional dataframes to be matched/merged to the VDJ and GEX. Will be matched by the column denoted in the merge.by parameter (should be present in the VDJ and all subsequent dataframes).
columns.to.transfer	string or vector of strings - columns that should be transferred/appended across all objects (matched by the merge.by parameter). Defaults to "all" - all unique columns from GEX not present in VDJ and vice-versa.

Value

An output VGM object: a list with the first element - the VDJ object; second element - the GEX/Seurat object. Additonal elements are appended to the list if additional.dataframes is not null.

Examples

```
try({
small_vgm <- VGM_build(
VDJ = small_vgm[[1]],
GEX = small_vgm[[2]],
columns.to.transfer = 'all') #transfer all new columns
})
```

VGM_expanded_clones *VDJ utility for T/F column for clonal expansion*

Description

Adds discrete columns containing TRUE / FALSE on whether a given cell is part of a expanded or not-expanded clonotype. Threshold frequency can be set.

Usage

```
VGM_expanded_clones(VGM, add.to.VDJ, add.to.GEX, expansion.threshold)
```

Arguments

VGM	Output object from the VDJ_GEX_matrix function (VDJ_GEX_matrix.output)
add.to.VDJ	Boolean. Whether to add expanded columns to VDJ matrix. Defaults to TRUE
add.to.GEX	Boolean. Whether to add expanded columns to GEX matrix. Defaults to TRUE
expansion.threshold	Integer. Defaults to 1. Cells in clonotypes above this threshold will be marked as expanded = TRUE.

Value

An output object from the VDJ_GEX_matrix function with added columns containing TRUE / FALSE values based on clonotype frequency.

Examples

```
VGM <- VGM_expanded_clones(
VGM = Platypus::small_vgm, add.to.VDJ = TRUE, add.to.GEX = TRUE,
expansion.threshold = 1)
```

VGM_expand_featurebarcodes *Utility for feature barcode assignment including clonal information*

Description

The VGM_expand_featurebarcodes function function can be used to trace back the cell origin of each sample after using cell hashing for single-cell sequencing. Replaces the original sample_id column of a vgm object with a pasted version of the original sample_id and the last digits of the feature barcode.

The original sample_id is stored in a new column called original_sample_id. Additionally, a second new column is created containing final barcode assignment information. Those barcodes match the origin FB_assignment if by.majority.barcodes is set to FALSE (default). However, if this input parameter is set to TRUE, the majority barcode assignment in stored in this colum.

Note: The majority barcode of a cell is the feature barcode which is most frequently assigned to the cells clonotype (10x default clonotype). The majority barcode assignment can be used under the assumption that all cells which are assigned to the same clonotype (within one sample), originate from the same donor organ or at least the same donor depending on the experimental setup.

For example: The original sample_id of a cell is "s1", the cell belongs to "clonotype1" and the feature barcode assigned to it is "i1-TotalSeq-C0953". If by.majority.barcodes default (FALSE) is used, the resulting new sample_id would be "s1_0953". However, if majority barcode assignment is used AND "i1-TotalSeq-C0953" is not the most frequently occurring barcode in "clonotype1" but rather barcode "i1-TotalSeq-C0951", the new sample_id would be "s1_0951". –> e.g., if 15 cells belong to clonotype1: 3 cells have no assigned barcode, 2 are assigned to "i1-TotalSeq-C0953" and 10 are assigned to "i1-TotalSeq-C0951" –> all 15 cells will have the new sample_id "s1_0951".

Usage

```
VGM_expand_featurebarcodes(
  vgm,
  by.majority.barcodes,
  integrate.in.gex,
  vdj.only,
  platypus.version
)
```

Arguments

vgm	VGM output of VDJ_GEX_matrix function (Platypus V3)
by.majority.barcodes	Logical. Default is FALSE. Indicated whether strict barcode assignment or majority barcode assignment should be used to create the new sample_id. If TRUE, for each clonotype the most frequent feature barcode will be chosen and assigned to each cell, even if that cell itself does not have this particular barcode assigned.
integrate.in.gex	Logical. Default is FALSE. If TRUE, the newly created sample_id's are integrated into gex component as well. Not recommended if no further gex analysis is done due to much longer computational time.
vdj.only	Logical. Defines if only vdj information is provided as input. Default is set to FALSE. If set to TRUE a vdj dataframe has to be provided as input (vgm = vdj).

Also, integrate.in.gex is automatically set to FALSE since no gex (vgm[[2]]) information is provided.

platypus.version
: This function works with "v3" only, there is no need to set this parameter.

Value

This function returns a vgm with new sample_id's in case vdj.only is set to FALSE (default). If vdj.only is set to true only the vdj dataframe with new sample_id's is returned. Note: If vdj.only is set to default (FALSE), VDJ information in the metadata of the GEX object is necessary. For this set integrate.VDJ.to.GEX to TRUE in the VDJ_GEX_matrix function

Examples

```
vgm_expanded_fb <- VGM_expand_featurebarcodes(
vgm = small_vgm[[1]],
by.majority.barcodes = FALSE,
integrate.in.gex=FALSE, vdj.only= TRUE)

vgm_expanded_fb <- VGM_expand_featurebarcodes(
vgm = small_vgm,
by.majority.barcodes = FALSE,
integrate.in.gex=TRUE, vdj.only= FALSE)

vgm_expanded_fb <- VGM_expand_featurebarcodes(vgm = small_vgm,
by.majority.barcodes = TRUE,
integrate.in.gex=TRUE, vdj.only= FALSE)
```

VGM_integrate *Utility for VDJ GEX matrix to integrated VDJ and GEX objects after addition of data to either*

Description

(Re)-intergrated VDJ and GEX of one or two separate VGM objects. This can be used as a simple "updating" utility function, if metadata has been added to the VDJ dataframe and is also needed in the GEX matrix or the reverse. Entries are integrated by barcode. If barcodes have been altered (barcode column in VDJ and cell names in GEX), the function will not yield results

Usage

```
VGM_integrate(VGM, columns.to.transfer, genes.to.VDJ, seurat.slot)
```

Arguments

VGM	Output object from the VDJ_GEX_matrix function (VDJ_GEX_matrix.output)
columns.to.transfer	Optional. Character Vector. Column names of either the VDJ matrix or GEX meta.data that should be transferred to the corresponding other matrix. if not provided all columns missing from one will be integrated into the other matrix
genes.to.VDJ	Character vector of gene names in GEX. In many cases it is useful to extract expression values for a gene to metadata. This is done via SeuratObject::FetchData(vars = genes,slot = seurat.slot) function. The VGM integrate takes gene ids, extracts these and adds them to the VDJ dataframe. If provided, no other columns are integrated between VDJ and GEX and columns.to.transfer is ignored.
seurat.slot	GEX object data slot to pull from. Can be 'counts', 'data', or 'scale.data'

Value

An output object from the VDJ_GEX_matrix function with added columns in VDJ or GEX

Examples

```
small_vgm[[1]] <- VDJ_clonotype(VDJ=Platypus::small_vgm[[1]],
clone.strategy="cdr3.nt",
hierarchical = "single.chains", global.clonotype = TRUE)

small_vgm <- VGM_integrate(
VGM = small_vgm,
columns.to.transfer = NULL) #transfer all new columns

small_vgm <- VGM_integrate(
VGM = small_vgm,
columns.to.transfer = c("global_clonotype_id_cdr3.nt"))

small_vgm <- VGM_integrate(
small_vgm, genes.to.VDJ = c("CD19","CD24A"),seurat.slot = "counts")
```

Index