Package 'Platypus'

October 12, 2022

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 (Yermanos A, et al (2021) <doi:10.1093/nargab/lqab023>).

Version 3.4.1

Date 2022-03-01

Maintainer Alexander Yermanos <ayermanos@gmail.com>

Depends R(>= 3.5.0),

Imports BiocGenerics, Biostrings, cowplot, dplyr, ggplot2, ggtree, jsonlite, Matrix(>= 1.3-3), doParallel, foreach, plyr, reshape2, seqinr, stringr, Seurat, SeuratObject, tibble, tidyr, utils, useful

Suggests AnnotationDbi, ape, bigstatsr, bio3d, Biobase, biomaRt, caret, circlize, colorspace, data.table, dichromat, e1071, edgeR, fda.usc, fgsea, ggalluvial, ggpubr, ggrepel, ggridges, ggseqlogo, graphkernels, graphlayouts, grid, gridExtra, harmony, igraph, iNEXT, IRanges, limma, keras, kmer, knitr, magrittr, Matrix.utils, methods, monocle3, msigdbr, naivebayes, org.Hs.eg.db, org.Mm.eg.db, Peptides, phangorn, pheatmap, phytools, pROC, ProjecTILs, protr, purrr, r3dmol, randomForest, RColorBrewer, readbitmap, readr, readxl, reticulate, rstudioapi, rlang, Rtsne, scales, scuttle, SeuratWrappers, SingleCellExperiment, slingshot, ssh, SummarizedExperiment, stringdist, tensorflow, tidygraph, tidytree, tidyselect, umap, vanddraabe, vegan, viridis, xgboost, yardstick, testthat (>= 3.0.0)

Additional_repositories https://vickreiner.github.io/drat/

License GPL-2

Encoding UTF-8

RoxygenNote 7.1.1

VignetteBuilder knitr

LazyData true

Config/testthat/edition 3

NeedsCompilation no

Author Alexander Yermanos [aut, cre],
Andreas Agrafiotis [ctb],
Raphael Kuhn [ctb],
Danielle Shlesinger [ctb],
Jiami Han [ctb],
Tudor-Stefan Cotet [ctb],
Victor Kreiner [ctb]

Repository CRAN

Date/Publication 2022-08-15 07:20:20 UTC

R topics documented:

`AbForests_AntibodyForest`

Infer and draw B cell evolutionary networks

Description

AntibodyForest takes the output of either ConvertStructure or CsvToDf or SubRepertoires or RemoveNets and outputs B cell phylogenetic networks in tree format. There is also the possibility to give the full-length list of clonal lineages, which contains both isotype and transcriptional cluster information, only when no prior data transformation is desired. Each network represents a clonal lineage, referring to the number of B cell receptor sequences originating from an independent V(D)J recombination event. Each vertex represents a unique recovered full-length variable heavy and light chain antibody sequence of a clonal family. Edges separating nodes are drawn given that clonal variants are similarly related according to their Levenshtein distance. Edge weights are extacted from the distance matrix apart from the special case of unmutated germline, in which the weights of outgoing edges from it are either set to 1 or to the difference between the corresponding distance from the matrix and the absolute value of the difference between the sequence lengths of germline and corresponding connected nodes. At tree building, starting from the reference ancestral germline, each node is connected to nodes that can be reached via the minimum distance based on the distance matrix calculation. Therefore, potential edges that go back to previous tree layers along with bidirectional circles are eliminated. Polytomies, displayed by B cell clones producing multiple distinct offsprings, are resolved in case of reaching nodes with equal minimum distance. Indeed, the algorithm removes edges either randomly from the recipient nodes,based on the node closest or farthest from the germline, considering the number of intermediate nodes or edge path length, or the highest/lowest counting of cells on the present node. Additional ties are settled by random edge selection. Consequently, parsimony holds, meaning that each daughter node has only one parent. Distinct tree topologies enable to visually investigate the trade-off between balance and evolution, and further quantify the amount of diversification of the subsequent detected clonal abundant clones during somatic hypermutation and class switching. The minimum decision-based criterion determines the amount of balance presented in the tree, while the maximum decision-based method the amount of evolution presented in the tree. Single color or color distribution on each node demonstrates the proportion of B cells with the specific isotype(s) or transcriptional cluster(s), while setting the size of vertices can be performed based on the number of unique sequences per clone, vertex betweenness and vertex closeness. Scaling of nodes by their relative clonal expansion assists in pinpointing identical antibody sequences across a multitude of B cells. Node labeling can depict clonal frequency.

Usage

```
AbForests_AntibodyForest(
  full_list,
  csv,
  files,
  distance_mat,
  clonal_frequency,
  scaleByClonalFreq,
  weight,
  tie_flag,
  scaleBybetweenness,
  scaleBycloclosenes_metr,
  opt,
  random.seed,
  alg_opt,
  cdr3
)
```

Arguments

full_list	a list of clone lineages, represented as data.frames
csv	an indicator variable. TRUE if full_list argument is a list of csv files, FALSE otherwise
files	a list of data.frames. Each data.frame contains 2 columns, one that describes the sequences and the other which type of information (isotype or cluster) is considered in the analysis. All these cases are determined by the user.
distance_mat	a custom integer distance matrix, or NULL for using the default distance matrix (calucated based on the levenshtein distance, which counts the number of mutations between sequences).
clonal_frequency	a logical variable, TRUE if labeling of vertices is based on clonal frequency and FALSE otherwise.
scaleByClonalFreq	logical variable with TRUE if vertex size is scaled by the number of unique sequences per clone and FALSE otherwise.
weight	logical variable. When its value is FALSE, then the weights of outgoing edges from Germline node are set to 1. When its value is TRUE, the weights are set to the difference between the number of mutations among sequences in germline and connected nodes(value in the corresponding distance matrix) and the absolute value of the difference between the sequence lengths of germline and corresponding connected nodes. In both cases, weights of remaining edges are extracted from the distance matrix. Outgoing edges from Germline represent the number of mutations of sequences having as common ancestor the Germline.
tie_flag	a string, with options 'rand', 'full', 'close_to_germ', 'far_from_germ', 'close_path_to_germ', 'far_path_from_germ','most_expanded' and 'least_expanded' for removing edges when equal distance (tie) in distance matrix. 'rand' means random pruning

in one of nodes, 'full' means keeping all nodes, close_to_germ means pruning of node(s) farthest from germline (based on number of intermediate nodes), 'far_from_germ' means pruning of node(s) closest to germline (based on number of intermediate nodes), 'close_path_to_germ' means pruning of node(s) farthest from germline (based on edge path length), 'far_path_from_germ' meams pruning of node(s) closest to germline (based on edge path length),'most_expanded' means pruning of node(s) with the lowest B cell count(clonal frequency) and least_expanded, which means pruning of node(s) with the hightest B cell count(clonal frequency). In cases of subsequent ties, a random node is selected.

`scaleBybetweenness`
: logical variable with TRUE if vertex size is scaled by the vertex betweenness centrality.

`scaleBycloclosenes_metr`
: logical variable with TRUE if vertex size is scaled by closeness centrality of vertices in graph.

`opt`
: a string with options "isotype" and "cluster". The option "isotype" is utilized when the user desires to do an isotype analysis, while the selection of "cluster" denotes that an analysis based on transcriptome is requested.

`random.seed`
: a random seed, specified by the user, when random sampling of sequences happens in each of the cases described in tie_flag argument.

`alg_opt`
: a string denoting the version of the edge selection algorithm used in the construction of networks. Possible choices: "naive", "two-step".

`cdr3`
: variable with values 0 if the user desires to select full length sequences (only when the input is a list of csv files), 1 for sequences in the CDR3 only (only when the input is a list of csv files) and NULL otherwise.

Value

graphs. A list of lists. E.g graphs[[1]][[1]] network: an igraph object, containing the first network in tree format. graphs[[1]][[2]] legend: contains the legend parameters of the first network. graphs[[1]][[3]] count.rand: contains the number of randomly considered nodes for the first network. graphs[[1]][[4]] adj.matrix: contains the adjacency matrix for the first network. graphs[[1]][[5]] distance.matrix: contains the distance matrix for the first network. graphs[[1]][[6]] cells.per.network: contains the number of cells for the first network. graphs[[1]][[7]] variants.per.network: contains the number of variants for the first network. graphs[[1]][[8]] variant.sequences: contains the sequences of the variants for the first network. graphs[[1]][[9]] cells.per.variant: contains the number of cells per variant (clonal frequency) for the first network. graphs[[1]][[10]] cell.indicies.per.variant: the indices of cells per variant for the first network. graphs[[1]][[11]] new.variant.names: contains the names of variants for the first network. graphs[[1]][[12]] germline.index: contains the index of germline sequence for the first network. graphs[[1]][[13]] isotype.per.variant: contains the isotypes corresponding to each variant for the first network. graphs[[1]][[14]] transcriptome.cluster.per.variant: contains the transcriptional clusters corresponding to each variant for the first network. graphs[[1]][[15]] isotype.per.cell: contains the isotype corresponding to each cell for the first network. graphs[[1]][[16]] transcriptome.cluster.per.cell: contains the transcriptional cluster corresponding to each cell for the first network.

See Also

ConvertStructure, CsvToDf, SubRepertoires, RemoveNets

Examples

```
## Not run:
AbForests_AntibodyForest(full_list = Platypus::new,csv=FALSE, files,clonal_frequency=TRUE,
scaleByClonalFreq=TRUE,weight=TRUE,tie_flag='close_to_germ',
scaleBybetweenness=FALSE,scaleBycloclosenes_metr=FALSE,
opt="cluster",alg_opt="0",cdr3=NULL)

## End(Not run)
```

`AbForests_CompareForests`

Comparison of distinct B cell repertoires

Description

CompareForests takes the output of AntibodyForest for 2 distinct repertoires and performs a comparison of these 2 repertoires.

Usage

```
AbForests_CompareForests(
  list1,
  list2,
  DAG,
  clonal_frequency,
  scaleByClonalFreq,
  weight,
  tie_flag,
  opt
)
```

Arguments

`list1`	a list of lists. Each sublist contains an igraph object with the networks of the evolved B clonal lineages in tree format, their legend and the number of randomly considered nodes per network for Repertoire of 1 (Output of AntibodyForest). E.g list1[[1]][[1]] is an igraph object, containing the first network of the evolved B clonal lineage in tree format. list1[[1]][[2]] contains the legend parameters of the first network of the evolved B clonal lineage. list1[[1]][[3]] is the number of randomly considered nodes for the first network of the evolved B clonal lineage.

`list2` a list of lists. Each sublist contains an igraph object with the networks of the evolved B clonal lineages in tree format, their legend and the number of randomly considered nodes per network for Repertoire of 2 (Output of AntibodyForest). E.g list2[[1]][[1]] is an igraph object, containing the first network of the evolved B clonal lineage in tree format. list2[[1]][[2]] contains the legend parameters of the first network of the evolved B clonal lineage. list2[[1]][[3]] is the number of randomly considered nodes for the first network of the evolved B clonal lineage.

`DAG` a logical variable, when TRUE a directed acyclic graph is produced.

`clonal_frequency`
a logical variable, TRUE if labeling of vertices is based on clonal frequency and FALSE otherwise.

`scaleByClonalFreq`
logical variable with TRUE if vertex size is scaled by the number of unique sequences per clone and FALSE otherwise.

`weight` logical variable. When its value is FALSE, then the weights of outgoing edges from Germline node are set to 1. When its value is TRUE, the weights are set to the difference between the number of mutations among sequences in germline and connected nodes(value in the corresponding distance matrix) and the absolute value of the difference between the sequence lengths of germline and corresponding connected nodes. In both cases, weights of remaining edges are extracted from the distance matrix. Outgoing edges from Germline represent the number of mutations of sequences having as common ancestor the Germline.

`tie_flag` a string, with options 'rand', 'full', 'close_to_germ', 'far_from_germ', 'close_path_to_germ', 'far_path_from_germ','most_expanded' and 'least_expanded' for removing edges when equal distance (tie) in distance matrix. 'rand' means random pruning in one of nodes, 'full' means keeping all nodes, close_to_germ means pruning of node(s) farthest from germline (based on number of intermediate nodes), 'far_from_germ' means pruning of node(s) closest to germline (based on number of intermediate nodes), 'close_path_to_germ' means pruning of node(s) farthest from germline (based on edge path length), 'far_path_from_germ' meams pruning of node(s) closest to germline (based on edge path length),'most_expanded' means pruning of node(s) with the lowest B cell count(clonal frequency) and least_expanded, which means pruning of node(s) with the hightest B cell count(clonal frequency). In cases of subsequent ties, a random node is selected.

`opt` a string with options "isotype" and "cluster". The option "isotype" is utilized when the user desires to do an isotype analysis, while the selection of "cluster" denotes that an analysis based on transcriptome is requested.

Value

combined_df. A data.frame that summarizes metrics for both repertoires. In particular, each row represents a single network and networks of both repertoires are combined row wise. Columns of combined_df are: Column1: Weighted.Longest.path.from.germline. Column2: Length.of.weighted.longest.shortest.path.fro
Column3: Unweighted.Longest.path.from.germline. Column4: Length.of.unweighted.longest.shortest.path.from.germline. Column5: Average.number.of.daughter.cells. Column6: Std.number.of.daughter.cells. Column7: Min.number.of.daughter.cells. Column8: Max.number.of.daughter.cells. Column9: Weighted.vertex.degree.

Column10: Average.number.of.clusters/isotypes. Column11: Isotypes/Clusters.info. Column12: vertex.betweenness.centrality. Column13: edge.betweenness.centrality. Column14: closeness.centrality.of.vertices. Column15: global.clustering.coefficient. Column16: average.clustering.coefficient. Column17: Mean.clonal.expansion. If the labeling or scaling of nodes in graph is based on clonal frequency (arguments: clonal_frequency==TRUE or scaleByClonalFreq==TRUE), then combined_df contains also: Column18: Ratio.Number.of.edges.from.germline.to.each.node.with.clonal.frequency. Column19: Mean.Ratio.Number.of.edges.from.germline.to.each.node.with.clonal.frequency. Column20: Mean.number.of.edges.from.germline. Column21: Ratio.Total.path.length.from.germline.to.each.node.with.clonal.f Column22: Mean.Ratio.Total.path.length.from.germline.to.each.node.with.clonal.frequency. Column23: Mean.Total.path.length.from.germline. Column24: Repertoire.id. Column25: Number.of.sequences.

isotype_info_rep1 A data.frame. It summarizes isotype/cluster info for repertoire 1.

isotype_info_rep2 A data.frame. It summarizes isotype/cluster info for repertoire 2.

See Also

AntibodyForest, ForestMetrics

Examples

```
## Not run:
CompareForests(list1,list2,DAG=TRUE,
clonal_frequency=TRUE,scaleByClonalFreq=TRUE,weight=TRUE,
tie_flag='close_to_germ',opt="cluster")

## End(Not run)
```

AbForests_ConvertStructure

Extract transcriptome/isotype information and B cell receptor sequences from single cell immune repertoire formatted as list of data.frames

---

Description

ConvertStructure alters a list of clone lineages, represented as data.frames and recovers the type of isotypes or transcriptional clusters and antibody sequences from these clone lineages. It can receive as input the original data or the output of SubRepertoiresByUniqueSeq or SubRepertoiresByCells. Then, the output list is used as input to the RemoveNets or AntibodyForest function.

Usage

```
AbForests_ConvertStructure(list, opt, cdr3)
```

Arguments

list	a list of data.frames. Each data.frame may contain information concerning full length heavy and light chain sequences, CDRH3 and CDRL3 sequences, the type of isotype and the transcriptional cluster that corresponds to each of these sequences.
opt	a string with options "isotype" and "cluster". The option "isotype" is utilized when the user desires to do an isotype analysis, while the selection of "cluster" denotes that an analysis based on transcriptome is requested.
cdr3	variable with values 0 if the user desires to select full length sequences (only when the input is a list of csv files), 1 for sequences in the CDR3 only (only when the input is a list of csv files) and NULL otherwise.

Value

list a list of data.frames. Each data.frame contains 2 columns, one that describes the sequences and the other which type of information (isotype or cluster) is considered in the analysis. All these cases are determined by the user.

See Also

RemoveNets, AntibodyForest

Examples

```
## Not run:
ConvertStructure(list,opt="cluster",cdr3=NULL)
ConvertStructure(list,opt="isotype",1)

## End(Not run)
```

AbForests_CsvToDf *Convert list of csvs, to nested list of data.frames*

Description

CsvToDf converts a list of csv files, which are clone lineages to a list of data.frames.

Usage

```
AbForests_CsvToDf(files)
```

Arguments

files	a list of csv files. Each csv file may contain information concerning full length heavy and light chain sequences, CDRH3 and CDRL3 sequences, the type of isotype and the transcriptional cluster that corresponds to each of these sequences.

Value

graphs a list of data.frames. Each data.frame contains 2 columns, one that describes the sequences and the other the type of information (isotype or cluster) is considered in the analysis. All these cases are determined by the user.

See Also

AntibodyForest

Examples

```
## Not run:
CsvToDf(files)

## End(Not run)
```

`AbForests_ForestMetrics` *Calculate metrics for networks*

Description

ForestMetrics takes the output of AntibodyForest and calculates metrics for each of the networks.

Usage

```
AbForests_ForestMetrics(
  graphs,
  DAG,
  clonal_frequency,
  scaleByClonalFreq,
  weight,
  tie_flag,
  opt
)
```

Arguments

graphs	A list of lists. Each sublist contains an igraph object with the networks of the evolved B clonal lineages in tree format, their legend and the number of randomly considered nodes per network(Output of AntibodyForest function). E.g graphs[[1]][[1]] is an igraph object, containing the first network of the evolved B clonal lineage in tree format. graphs[[1]][[2]] contains the legend parameters of the first network of the evolved B clonal lineage. graphs[[1]][[3]] is the number of randomly considered nodes for the first network of the evolved B clonal lineage.
DAG	a logical variable, when TRUE a directed acyclic graph is produced.

`clonal_frequency`
: a logical variable, TRUE if labeling of vertices is based on clonal frequency and FALSE otherwise.

`scaleByClonalFreq`
: logical variable with TRUE if vertex size is scaled by the number of unique sequences per clone and FALSE otherwise.

`weight`
: logical variable. When its value is FALSE, then the weights of outgoing edges from Germline node are set to 1. When its value is TRUE, the weights are set to the difference between the number of mutations among sequences in germline and connected nodes(value in the corresponding distance matrix) and the absolute value of the difference between the sequence lengths of germline and corresponding connected nodes. In both cases, weights of remaining edges are extracted from the distance matrix. Outgoing edges from Germline represent the number of mutations of sequences having as common ancestor the Germline.

`tie_flag`
: a string, with options 'rand', 'full', 'close_to_germ', 'far_from_germ', 'close_path_to_germ', 'far_path_from_germ','most_expanded' and 'least_expanded' for removing edges when equal distance (tie) in distance matrix. 'rand' means random pruning in one of nodes, 'full' means keeping all nodes, close_to_germ means pruning of node(s) farthest from germline (based on number of intermediate nodes), 'far_from_germ' means pruning of node(s) closest to germline (based on number of intermediate nodes), 'close_path_to_germ' means pruning of node(s) farthest from germline (based on edge path length), 'far_path_from_germ' meams pruning of node(s) closest to germline (based on edge path length),'most_expanded' means pruning of node(s) with the lowest B cell count(clonal frequency) and least_expanded, which means pruning of node(s) with the hightest B cell count(clonal frequency). In cases of subsequent ties, a random node is selected.

`opt`
: a string with options "isotype" and "cluster". The option "isotype" is utilized when the user desires to do an isotype analysis, while the selection of "cluster" denotes that an analysis based on transcriptome is requested.

Value

metrics. A list of lists. Each list contains various metrics for the quantification of networks. E.g metrics[[1]][[1]] is the weighted Longest path from germline for the first network. metrics[[1]][[2]] is the length of weighted longest shortest path from germline for the first network. metrics[[1]][[3]] is the unweighted Longest path from germline for the first network. metrics[[1]][[4]] is the length of unweighted longest shortest path from germline for the first network. metrics[[1]][[5]] is the weighted shortest path network for the first network. metrics[[1]][[6]] is the uweighted shortest path network for the first network. metrics[[1]][[7]] is the average number of daughter cells for the first network. metrics[[1]][[8]] is the std number of daughter cells for the first network. metrics[[1]][[9]] is the min number of daughter cells for the first network. metrics[[1]][[10]] is the max number of daughter cells for the first network. metrics[[1]][[11]] is a ggplot object that contains the plot of Degree Distribution of daughter cells for the first network. metrics[[1]][[12]] is the weighted vertex degree for the first network. metrics[[1]][[13]] is a ggplot object that contains the plot of unweighted Degree Distribution of daughter cells for the first network. metrics[[1]][[14]] is the average number of isotypes for the first network. metrics[[1]][[15]] is a ggplot object that contains the plot of Distribution of isotypes for the first network. metrics[[1]][[16]] is the Isotypes/Clusters info data.frame with columns Parent, Child and Parent-Child, which contains the

type of isotypes/clusters for each pair of nodes (Parent-Child relationship in tree) found in the first network. metrics[[1]][[17]] is a ggplot object that contains the plot of Isotype/Cluster Directionality for the first network. In particular, the frequency of all types of isotypes/clusters for each pair of nodes in the tree is depicted. metrics[[1]][[18]] is the vertex betweenness centrality for the first network. It is defined by the number of geodesics (shortest paths) going through a vertex according to igraph documentation. metrics[[1]][[19]] is the edge betweenness centrality for the first network. It is defined by the number of geodesics (shortest paths) going through an edge according to igraph documentation. metrics[[1]][[20]] is the closeness centrality of vertices for the first network. The closeness centrality of a vertex is defined by the inverse of the average length of the shortest paths to/from all the other vertices in the graph according to igraph documentation. metrics[[1]][[21]] is a ggplot object that contains the plot of Path length from Germline vs Node Degree for the first network. metrics[[1]][[22]] is a ggplot object that contains the plot of Number of edges from Germline vs Node Degree for the first network. metrics[[1]][[23]] is an igraph object that contains the Isotype/Cluster transition network for the first network. metrics[[1]][[24]] is the global clustering coefficient for the first network. metrics[[1]][[25]] is the average clustering coefficient for the first network. metrics[[1]][[26]] is the mean clonal expansion for the first network, calculated as the mean of clonal frequencies of all vertices in the network. If the labeling or scaling of nodes in graph is based on clonal frequency (arguments: clonal_frequency==TRUE or scaleByClonalFreq==TRUE), then metrics[[1]][[27]] is the ratio: Number of edges from germline to each node with clonal frequency for the first network. metrics[[1]][[28]] is the mean ratio: Number of edges from germline to each node with clonal frequency for the first network. metrics[[1]][[29]] is a ggplot object that contains the ratio of Number of edges from germline to each node with clonal frequency for the first network. metrics[[1]][[30]] is the mean number of edges from germline for the first network. metrics[[1]][[31]] is the ratio: Total path length from germline to each node with clonal frequency for the first network. metrics[[1]][[32]] is the mean ratio: Total path length from germline to each node with clonal frequency for the first network. metrics[[1]][[33]] is a ggplot object that contains the ratio of Total path length from germline to each node with clonal frequency for the first network. metrics[[1]][[34]] is the mean Total path length from germline for the first network. metrics[[2]][[1]] is the weighted Longest path from germline for the second network.

See Also

AntibodyForest

Examples

```
## Not run:
ForestMetrics(graphs,DAG=TRUE,clonal_frequency=TRUE,scaleByClonalFreq=TRUE,
weight=TRUE,tie_flag='close_to_germ',opt="cluster")

## End(Not run)
```

`AbForests_PlotGraphs` *Plot igraph and ggplot objects*

Description

PlotGraphs takes as input the output of AntibodyForest or ForestMetrics functions and plots all corresponding networks in the single cell immune repertoire or the corresponding ggplot object, the user specifies, from all clone lineages. The function gives the option in the user to store each tree or ggplot object within the repertoire in pdf format.

Usage

```
AbForests_PlotGraphs(graphs, no_arg, topdf, filename)
```

Arguments

graphs	a list of networks (Output of AntibodyForest function) or metrics (Output of ForestMetrics function).
no_arg	element of list the user desires to plot : integer value,if the user desires to plot a metric and NULL, if the user desires to plot the networks.
topdf	logical value, TRUE if user wants to store plots in pdf format (the no_arg element of each list is saved in a separate page of the pdf).
filename	name of saved pdf file based on the user's preferences.

Value

Empty, output plots are written to file as specified by the user with the parameter filename

See Also

AntibodyForest, ForestMetrics

Examples

```
## Not run:
PlotGraphs(graphs,no_arg=NULL,topdf=TRUE,filename)
PlotGraphs(graphs,no_arg5,topdf=TRUE,filename)

## End(Not run)
```

AbForests_PlyloToMatrix

Conversion of phylogenetic tree to distance matrix

Description

PlyloToMatrix converts a previously existing phylogenetic tree to a corresponding distance matrix using the cophenetic distance. Then, there is the option to utilize this custom distance matrix as an input distance matrix to AntibodyForest function. The user is responsible for specifying a correct and valid distance matrix. In particular, the size of distance matrix must match the number of sequences for each network in each repertoire.

Usage

```
AbForests_PlyloToMatrix(tree_name)
```

Arguments

tree_name	a plylogenetic tree (phylo object).

Value

dist_mat The corresponding distance matrix uses the cophenetic distance between two observations that have been clustered. This distance is defined to be the intergroup dissimilarity at which the two observations are first combined into a single cluster.

See Also

AntibodyForest

Examples

```
## Not run:
PlyloToMatrix(tree_name)

## End(Not run)
```

AbForests_RemoveNets *Filter sub-repertoires with less than N unique sequences or with less than C unique cells*

Description

RemoveNets takes the output of SubRepertoires and performs the filtering of the 5 sub-repertoires. In particular, from these 5 sub-repertoires, networks with number of nodes or number of unique sequences below a specified threshold are eliminated.

Usage

```
AbForests_RemoveNets(
  list,
  opt,
  distance_mat,
  tie_flag,
  weight,
  N,
  C,
  random.seed,
  alg_opt
)
```

Arguments

list	a list of 5 sub-lists of data.frames. Each sub-list corresponds to the set of networks, in which a majority isotype is specifyied. list[[1]] or list$list_IGHG contains the networks, in data.frame format, with more IGG isotypes, list[[2]] or list$list_IGHA contains the networks, in data.frame format, with more IGA isotypes, list[[3]] or list$list_IGHM contains the networks, in data.frame format, with more IGM isotypes, list[[4]] or list$list_IGAG contains the networks, in data.frame format, with a tie in IGA and IGG isotypes and list[[5]] or list$list_other contains the networks, in data.frame format, with other isotypes apart from the aforementioned combinations. In each sub-list, each data.frame represents a clone lineage and contains 2 columns, one that describes the antibody sequences and the other which type of information (isotype) is considered in the analysis. This list of data.frames has been generated by SubRepertoires function based on the initial data input and user's preferences.
opt	a string with options "isotype" and "cluster". The option "isotype" is utilized when the user desires to do an isotype analysis, while the selection of "cluster" denotes that an analysis based on transcriptome is requested.
distance_mat	a custom integer distance matrix, or NULL for using the default distance matrix (calucated based on the levenshtein distance, which counts the number of mutations between sequences). Given the phylogenetic tree, a custom-made distance matrix can be produced by PlyloToMatrix function.
tie_flag	a string, with options 'rand', 'full', 'close_to_germ', 'far_from_germ', 'close_path_to_germ', 'far_path_from_germ','most_expanded' and 'least_expanded' for removing edges when equal distance (tie) in distance matrix. 'rand' means random pruning in one of nodes, 'full' means keeping all nodes, close_to_germ means pruning of node(s) farthest from germline (based on number of intermediate nodes), 'far_from_germ' means pruning of node(s) closest to germline (based on number of intermediate nodes), 'close_path_to_germ' means pruning of node(s) farthest from germline (based on edge path length), 'far_path_from_germ' meams pruning of node(s) closest to germline (based on edge path length),'most_expanded' means pruning of node(s) with the lowest B cell count(clonal frequency) and least_expanded, which means pruning of node(s) with the hightest B cell count(clonal frequency). In cases of subsequent ties, a random node is selected.
weight	logical variable. When its value is FALSE, then the weights of outgoing edges from Germline node are set to 1. When its value is TRUE, the weights are set to the difference between the number of mutations among sequences in germline and connected nodes(value in the corresponding distance matrix) and the absolute value of the difference between the sequence lengths of germline and corresponding connected nodes. In both cases, weights of remaining edges are extracted from the distance matrix.
N	the threshold of unique sequences below which networks are removed.
C	the threshold of unique cells below which networks are removed.
random.seed	a random seed, specified by the user, when random sampling of sequences happens in each of the cases described in tie_flag argument.
alg_opt	a string denoting the version of the edge selection algorithm used in the construction of networks. "0" means the naive version and "1" the advanced one.

Value

list a nested list of 5 sub-lists of data.frames. Each sub-list corresponds to the reduced set of networks according to threshold N, in which a majority isotype is specifyied. list[[1]] or list$list_IGHG contains the networks, in data.frame format, with more IGG isotypes, list[[2]] or list$list_IGHA contains the networks, in data.frame format, with more IGA isotypes, list[[3]] or list$list_IGHM contains the networks, in data.frame format, with more IGM isotypes, list[[4]] or list$list_IGAG contains the networks, in data.frame format, with a tie in IGA and IGG isotypes and list[[5]] or list$list_other contains the networks, in data.frame format, with other isotypes apart from the aforementioned combinations.

See Also

SubRepertoires, SubRepertoiresByUniqueSeq, PlyloToMatrix, AntibodyForest

Examples

```
## Not run:
RemoveNets(list,opt="cluster",distance_mat=NULL,
tie_flag='close_to_germ',weight=TRUE,N=4,C=NULL,random.seed=165)

## End(Not run)
```

AbForests_SubRepertoiresByCells

Split single cell immune repertoire into sub-repertoires by isotype based on number of B cells

---

Description

SubRepertoiresByCells separates the single cell immune repertoire into 5 sub-repertoires taking into account the number of cells. The goal is to determine the majority isotype per each network in the immune repertoire. Therefore, each sub-repertoire is dominated by isotype IGG, IGA, IGM, other and if there is an equal number of IGA and IGG isotypes in a network, IGA-IGG category exists respectively. In particular, in case there exists a tie in the number of IGA and IGM, the network is considered to contain IGA as majority isotype, while the same number of IGG and IGM in the network categorize this network as containing IGG as majority isotype. The function receives the output of CsvToDf or original data and can be given as input to ConvertStructure function.

Usage

```
AbForests_SubRepertoiresByCells(list)
```

Arguments

`list`	a list of data.frames. Each data.frame represents a clone lineage and separates initial input data into subsets of networks.

Value

list a nested list of 5 sub-lists of data.frames. Each sub-list corresponds to the set of networks, in which a majority isotype is specifyied. list[[1]] or list$list_IGHG contains the networks, in data.frame format, with more IGG isotypes, list[[2]] or list$list_IGHA contains the networks, in data.frame format, with more IGA isotypes, list[[3]] or list$list_IGHM contains the networks, in data.frame format, with more IGM isotypes, list[[4]] or list$list_IGAG contains the networks, in data.frame format, with a tie in IGA and IGG isotypes and list[[5]] or list$list_other contains the networks, in data.frame format, with other isotypes apart from the aforementioned combinations.

See Also

ConvertStructure, CsvToDf

Examples

```
## Not run:
SubRepertoiresByCells(list)

## End(Not run)
```

AbForests_SubRepertoiresByUniqueSeq

Split single cell immune repertoire into sub-repertoires by isotype based on number of unique sequences

---

Description

SubRepertoiresByUniqueSeq separates the single cell immune repertoire into 5 sub-repertoires taking into account only unique sequences. The goal is to determine the majority isotype per each network in the immune repertoire. Therefore, each sub-repertoire is dominated by isotype IGG, IGA, IGM, other and if there is an equal number of IGA and IGG isotypes in a network, IGA-IGG category exists respectively. In particular, in case there exists a tie in the number of IGA and IGM, the network is considered to contain IGA as majority isotype, while the same number of IGG and IGM in the network categorize this network as containing IGG as majority isotype.

Usage

```
AbForests_SubRepertoiresByUniqueSeq(
  list,
  opt,
  distance_mat,
  tie_flag,
  weight,
  random.seed,
  alg_opt,
  cdr3
)
```

Arguments

list	a list of data.frames. Each data.frame represents a clone lineage and contains 2 columns, one that describes the antibody sequences and the other which type of information (isotype) is considered in the analysis. This list of data.frames has been generated by ConvertStructure function based on the initial data input or the output of CsvToDf and user's preferences.
opt	a string with options "isotype" and "cluster". The option "isotype" is utilized when the user desires to do an isotype analysis, while the selection of "cluster" denotes that an analysis based on transcriptome is requested.
distance_mat	a custom integer distance matrix, or NULL for using the default distance matrix (calucated based on the levenshtein distance, which counts the number of mutations between sequences). Given the phylogenetic tree, a custom-made distance matrix can be produced by PlyloToMatrix function.
tie_flag	a string, with options 'rand', 'full', 'close_to_germ', 'far_from_germ', 'close_path_to_germ', 'far_path_from_germ','most_expanded' and 'least_expanded' for removing edges when equal distance (tie) in distance matrix. 'rand' means random pruning in one of nodes, 'full' means keeping all nodes, close_to_germ means pruning of node(s) farthest from germline (based on number of intermediate nodes), 'far_from_germ' means pruning of node(s) closest to germline (based on number of intermediate nodes), 'close_path_to_germ' means pruning of node(s) farthest from germline (based on edge path length), 'far_path_from_germ' meams pruning of node(s) closest to germline (based on edge path length),'most_expanded' means pruning of node(s) with the lowest B cell count(clonal frequency) and least_expanded, which means pruning of node(s) with the hightest B cell count(clonal frequency). In cases of subsequent ties, a random node is selected.
weight	logical variable. When its value is FALSE, then the weights of outgoing edges from Germline node are set to 1. When its value is TRUE, the weights are set to the difference between the number of mutations among sequences in germline and connected nodes(value in the corresponding distance matrix) and the absolute value of the difference between the sequence lengths of germline and corresponding connected nodes. In both cases, weights of remaining edges are extracted from the distance matrix.
random.seed	a random seed, specified by the user, when random sampling of sequences happens in each of the cases described in tie_flag argument.
alg_opt	a string denoting the version of the edge selection algorithm used in the construction of networks. "0" means the naive version and "1" the advanced one.
cdr3	variable with values 0 if the user desires to select full length sequences (only when the input is a list of csv files), 1 for sequences in the CDR3 only (only when the input is a list of csv files) and NULL otherwise.

Value

list a nested list of 5 sub-lists of data.frames. Each sub-list corresponds to the set of networks, in which a majority isotype is specifyied. list[[1]] or list$list_IGHG contains the networks, in data.frame format, with more IGG isotypes, list[[2]] or list$list_IGHA contains the networks, in data.frame format, with more IGA isotypes, list[[3]] or list$list_IGHM contains the networks, in

data.frame format, with more IGM isotypes, list[[4]] or list$list_IGAG contains the networks, in data.frame format, with a tie in IGA and IGG isotypes and list[[5]] or list$list_other contains the networks, in data.frame format, with other isotypes apart from the aforementioned combinations.

See Also

AntibodyForest, ConvertStructure, CsvToDf, PlyloToMatrix

Examples

```
## Not run:
SubRepertoiresByUniqueSeq(list,opt="isotype",distance_mat=NULL,
tie_flag='close_to_germ',weight=TRUE,random.seed=165,alg_opt="naive",cdr3=NULL)

## End(Not run)
```

AbForests_UniqueAntibodyVariants

Count the number of unique antibody variants per clonal lineage

---

Description

UniqueAntibodyVariants calculates the number of unique antibody sequences, as dictated by the different grouping sequences strategy,for each network in the immune repertoire.

Usage

```
AbForests_UniqueAntibodyVariants(
  list,
  opt,
  distance_mat,
  tie_flag,
  weight,
  random.seed,
  alg_opt,
  cdr3
)
```

Arguments

list	a list of data.frames. Each data.frame represents a clone lineage and contains information on the antibody sequences and on the isotype/transcriptional cluster is considered in the analysis based the user's preferences.
opt	a string with options "isotype" and "cluster". The option "isotype" is utilized when the user desires to do an isotype analysis, while the selection of "cluster" denotes that an analysis based on transcriptome is requested.

`distance_mat` a custom integer distance matrix, or NULL for using the default distance matrix (calucated based on the levenshtein distance, which counts the number of mutations between sequences). Given the phylogenetic tree, a custom-made distance matrix can be produced by PlyloToMatrix function.

`tie_flag` a string, with options 'rand', 'full', 'close_to_germ', 'far_from_germ', 'close_path_to_germ', 'far_path_from_germ','most_expanded' and 'least_expanded' for removing edges when equal distance (tie) in distance matrix. 'rand' means random pruning in one of nodes, 'full' means keeping all nodes, close_to_germ means pruning of node(s) farthest from germline (based on number of intermediate nodes), 'far_from_germ' means pruning of node(s) closest to germline (based on number of intermediate nodes), 'close_path_to_germ' means pruning of node(s) farthest from germline (based on edge path length), 'far_path_from_germ' meams pruning of node(s) closest to germline (based on edge path length),'most_expanded' means pruning of node(s) with the lowest B cell count(clonal frequency) and least_expanded, which means pruning of node(s) with the hightest B cell count(clonal frequency). In cases of subsequent ties, a random node is selected.

`weight` logical variable. When its value is FALSE, then the weights of outgoing edges from Germline node are set to 1. When its value is TRUE, the weights are set to the difference between the number of mutations among sequences in germline and connected nodes(value in the corresponding distance matrix) and the absolute value of the difference between the sequence lengths of germline and corresponding connected nodes. In both cases, weights of remaining edges are extracted from the distance matrix.

`random.seed` a random seed, specified by the user, when random sampling of sequences happens in each of the cases described in tie_flag argument.

`alg_opt` a string denoting the version of the edge selection algorithm used in the construction of networks. "0" means the naive version and "1" the advanced one.

`cdr3` variable with values 0 if the user desires to select full length sequences (only when the input is a list of csv files), 1 for sequences in the CDR3 only (only when the input is a list of csv files) and NULL otherwise.

Value

uni_seq a vector, same size as list, which contains the number of unique antibody variants for each clonal lineage.

Examples

```
## Not run:
UniqueAntibodyVariants(list,opt="cluster",
distance_mat=NULL,tie_flag=close_to_germ,weight=TRUE,random.seed=165,alg_opt="naive",cdr3=NULL)

## End(Not run)
```

`AlphaFold_prediction` *Structure prediction of Mixcr wrapper output with Alpha Fold*

Description

This function takes the output from the VDJ_call_MIXCR function as input in the VDJ.mixcr.out argument and predicts the structure with Alpha Fold. From the VDJ.mixcr.out object the full length VDJ & VJ sequence containing all the frameworks and CDR's is used to predict the structure of the variable part with Alpha Fold multi. If the user has no access to the Euler function, the fucntion just returns a fasta file with the VDJ and VJ sequence, that can be used for running Alpha Fold on a Cluster. For users that have a login to the Euler cluster, this function will automatically connect to Euler and start Alpha Fold for all the indicated sequences. After the prediction is finished the same function can be used to import the predicted structures as a pdb file and add it to the input as a list object

Usage

```
AlphaFold_prediction(
  VDJ.mixcr.out,
  cells.to.predict,
  max.template.date,
  dir.name,
  fasta.storage.path,
  euler.user.name,
  rm.local.fasta,
  import,
  import.local.path,
  import.local.dirnames,
  euler.dirname,
  euler.dirpath,
  n.ranked,
  rm.euler.files,
  rm.local.output,
  output.path,
  antigen.fasta.path,
  fasta.directory.path,
  platypus.version
)
```

Arguments

`VDJ.mixcr.out`	Contains the output from the VDJ_call_MIXCR function with VJ_aa_mixcr and VDJ_aa_mixcr columns containing the full length amino acid sequence of Framework 1 - 4.
`cells.to.predict`	Here you can specify 10x barcodes for the cells of the VDJ.mixcr.out that should be used for structure prediction. It can be set to "ALL" if the antibody structure of all cells shall be predicted.

`max.template.date`
: This is a parameter for running Alpha Fold and a date can be specified in the following format: "yyyy-mm-dd" This tells Aplhpa Fold which state of the databases it shall use.

`dir.name`
: By default the function creates a directory named AlphaFold_Fasta, where the FASTA files created for prediction. The name of this directory can be changed by specifying the dir.name argument.

`fasta.storage.path`
: Here you can specify where the function saves the fasta files needed as an Alpha Fold input. By default files an 'AlphaFold_Fasta' directory with all the fasta files is created in the same directory as the R script runs.

`euler.user.name`
: If running Alpha Fold on Euler is requested, the user name needs to be specified in this parameter. Make sure that you have access to GPU usage on the Cluster. You will be prompted to enter you password by the "ssh" package which handles your credentials in a safe manner.

`rm.local.fasta`
: Here you can specify if the local AlphaFold_Fasta directory shall be deleted from your local computer after uploaded to the scratch on Euler. By default it is set to TRUE, to keep your environment clean. If the function is not used in the Euler modus it is set to FALSE, so you will have the fasta files as an output.

`import`
: This argument is for telling the function to import predicted structures. It is by default set to FALSE, which will initiate prediction not import. There are two options for importing predicted structures: Import = "euler" will start a connection to Euler and import the pdb files from the "AlphaFold_Fasta/output" directory. Import = "local" will import the pdb files form a local directory.

`import.local.path`
: If import = "local" is used you can specify the path to the AlphaFold_Output directory here. By default it is expected in the same directory as the r script runs.

`import.local.dirnames`
: If import = "local" is used the function expects a directory named 'Output_AlphaFold' in the same directory as the script runs. In case you do not wanna import all the pdb files off all samples in the 'Output_AlphaFold directory you can specify a sub directories in the import.local.dirnames parameter. (import.local.dirnames = c(s4_AGCCTAATCCCTTGCA-1_ranked,s4_CCCATACCACGTTGGC-1_ranked,...))

`euler.dirname`
: If import = "euler" is used the name of the directory containing the Alpha Fold output directory can be specified in euler.dirname. It is set to "AlphaFold_Fasta" by default and is expected to be on your scratch.

`euler.dirpath`
: If import = "euler" is used the path to the directory containing the Alpha Fold output folder can be specified in euler.path. By default the function expects the output in the AlphaFold_Fasta directory on your scratch. In case you wanna import the data from a different location you can specify the path here. The function expects a sub directory named output which contains sub directories named after the specific barcodes. (../scratch/AlphaFold_Fasta/output/s4_AGCCTAATCCCTTGCA-1/)

`n.ranked`
: Alpha Fold returns 21 predictions for each sequence which are ranked for 0:20. The ranked_0 is the most accurate according to the model. Here you can specify

how many of the top ranked structures are added to the output object. By default only the most accurate structure 'ranked_0.pdb' is integrated.

`rm.euler.files` Here you can specify if the files on Euler shall be deleted after importing them. It is set to FASLE by default to reduce the risk of unintentionally deleting the predictions. However, make sure to keep you scratch environment clean.

`rm.local.output`
Here you an specify if the downloaded output folder from Euler shall be deleted after the import. It is set to true by default to keep you environment clean.

`output.path` If the data is downloaded from the cluster it is by default stored in a sub folder in the current directory. If the data should be downloaded at a different location this can be specified in the output.path.

`antigen.fasta.path`
It can be of interest to predict the antibody structure together with the antigen to see interaction. For this purpose a path to a FASTA file containing the amino acid sequence of the antigen can be specified in the antigen.fasta.path argument. This will add the antigen sequence to every antibody prediction.

`fasta.directory.path`
The prediction function can also be used to predict structure directly from amino acid FASTA files without specifying a the VDJ.mixcr.out argument. For this the path to a directory, congaing all the FASTA files of interest can be specified in the fasta.directory.path argument. The files just need to have the .fasta extension. If multiple FASTA files are in the directory, the function will predict all of them separately.

`platypus.version`
This function is not directly depended on other Platypus functions but was developed to be compatible with v3.

Value

This function returns a list with the VDJ.mixcr.out in the first element and a list of pdb files as a second element

Note

For running Alpha Fold on Euler, the user needs to have access to GPU usage. This is automatically activated if one is part of the Reddy Euler Group.

If running prediction on Euler, the function will create a "AlphaFold_Fasta" directory in sour scratch on the cluster where all the fasta files are uploaded. The output files will be saved as well in this directory.

Examples

```
## Not run:

ADD EXAMPLE CODE HERE


## End(Not run)
```

`AntibodyForests` *Infer B cell evolutionary networks and/or sequence similarity networks*

Description

Function to infer immune receptor evolutionary networks (trees) or complex/sequence-similarity networks from a Platypus VDJ object/VGM[[1]] - AntibodyForests objects will be created for each sample and each unique clonotype, if the default parameters are used. Networks can be created in a tree-building manner (minimum spanning tree algorithm with custom tie solving methods), by linking sequences with a minimal string distance between them iteratively (and solving distance ties in a hierarchical way, with multiple resolve.ties parameters/configurations). Nodes in the network represent unique sequences per clonotype, edges are determined by the string distance between the nodes/sequences. Sequence types are dictated by the sequence.tyoe parameter, allowing networks to be built for most of the sequence types available in a Platypus VDJ object. Networks can also be created by pruning edges from a fully connected network obtained from all sequences from a specific clonotype - complex similarity networks. Pruning can either be done via a distance threshold (prunes nodes too far apart), a node degree threshold (to prune nodes with a smaller degree/not well connected), or an expansion threshold (to prune nodes for sequences with low expansion/frequency). Lastly, networks can be created by converting a phylogenetic tree inferred via different methods (neighbour-joining, maximum likelihood, maximum parsimony) into an igraph object,

Usage

```
AntibodyForests(
  VDJ,
  sequence.type,
  include.germline,
  network.algorithm,
  directed,
  distance.calculation,
  resolve.ties,
  connect.germline.to,
  pruning.threshold,
  remove.singletons,
  keep.largest.cc,
  VDJ.VJ.1chain,
  node.features,
  filter.na.features,
  filter.specific.features,
  node.limits,
  cell.limits,
  weighted.edges,
  weighted.germline,
  expand.intermediates,
  specific.networks,
```

```
  network.level,
  forest.method,
  random.seed,
  parallel,
  as.igraph
)
```

Arguments

VDJ	VDJ or vgm[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
sequence.type	string denoting the sequence types to create the networks for. 'cdr3.aa' - networks for amino-acid CDR3 sequences, 'cdr3.nt' - networks for nucleotide CDR3 sequences, 'VDJ.VJ.nt.trimmed' - full, trimmed VDJ-VJ sequences, as obtained when setting trin.and.align = T for VDJ_GEX_matrix(), 'VDJ.VJ.nt.raw' - full, raw VDJ-VJ sequences, 'VDJ.VJ.aa.mixcr' and 'VDJ.VJ.nt.mixcr' for the VDJ and VJ chains (nt or aa) as inferred by MIXCR, 'VDJ.aa.mixcr' and 'VDJ.nt.mixcr' for the VDJ chain inferred by MIXCR, 'VJ.aa.mixcr' and 'VJ.nt.mixcr' for the VJ chain inferred by MIXCR, 'VDJ.nt.trimmed' for the trimmed VDJ chain as nucleotides, 'VDJ.nt.raw' for the untrimmed VDJ chain as nucleotides, similarly for the VJ chain ('VJ.nt.trimmed' and 'VJ.nt.raw'), 'VDJ.cdr3s.aa' for the CDRH3 region as amino-acids, VDJ.cdr3s.nt' for the CDRH3 region as nucleotides, similarly for the CDRL3 regions ('VJ.cdr3s.aa', 'VJ.cdr3s.nt'), 'VDJ.aa' and 'VJ.aa' for the full VDJ/VJ sequence as amino-acids. Defaults to 'VDJ.VJ.nt.trimmed'.
include.germline	string or vector of strings, denoting the germline column(s) to be used (in the c('VDJ_germline', 'VJ_germline') order). 'trimmed.ref' - the networks will include a germline node, obtained by pasting the VDJ_trimmed_ref and VJ_trimmed_ref sequences for each clonotype, obtained by calling VDJ_call_MIXCR on VDJ. As reconstructed germlines as usually available for full VDJ.VJ.nt sequences, use this with sequence.type=VDJ.VJ.nt.trimmed. NULL will not include a germline.
network.algorithm	string denoting the algorithm used in constructing the networks. 'tree' - will use a tree evolutionary inference algorithm: nodes denoting unique sequences per clonotype will be linked iteratively, as long as their string distance is the minimum. Use the resolve.ties parameter to further dictate the tree topology (when there are multiple ties in the minimum links). 'prune' will create networks by pruning nodes from a fully connected networks. It must always be followed by a pruning method. For example, 'prune.distance' will prune nodes with a larger string distance than the threshold specified in the pruning.threshold parameter. 'prune.degree' will remove nodes with a lower degree than the threshold specified in pruning.threshold. 'prune.expansion' will remove nodes with a lower sequence frequency/expansion than the threshold specified in pruning.threshold. Multiple pruning methods can be used simultaneously, as long as multiple pruning.threshold is a vector - a threshold for each method. For example, 'prune.distance.degree' with pruning.threshold=c(3,2) will remove edges of nodes with a string distance greater than 3, then will remove nodes with a degree smaller than 2. 'prune.distance.degree.expansion' with pruning.threshold=c(3,2,1) will also re-

move one-of sequences/nodes (with a single cell). 'phylogenetic.tree.nj' will create phylogenetic (binary) trees using the neighbour-joining algorithm via ape::nj(). 'phylogenetic.tree.ml' will create phylogenetic (binary) trees using a maximum-likelihood algorithm from the phangorn package (phangorn::ml()). 'phylogenetic.tree.mp' will create phylogenetic (binary) trees using a maximum-parsimony algorithm from the phangorn package (phangorn::mp()). 'mst' will create undirected trees using the minimum spanning tree algorithm from igraph (without specific tie solving mechanisms). 'global' is a custom option to easily create whole-repertoire/multi-repertoire similarity networks: it defaults to the 'prune.distance' option, while also changing some other parameters to ensure consistency (directed is set to F, include.germline is set to F, network.level is set to 'global')

`directed` boolean, whether networks obtained using network.algorithm='tree' should be directed (from the germline to the leaf nodes) or not. T - directed; F - undirected trees.

`distance.calculation` string or function, specifying the method for calculating string distances between the sequences. Must be compatible with the method parameter of the stringdist::stringdistmatrix() function. Will default to 'lv' for Levenshtein distances. Else, if a function of the form distance(seq1, seq2) must be specified, which should output a float = custom distance metric between the selected sequences.

`resolve.ties` vector of strings, denoting the manner in which ties should be resolved when assembling trees via network.algorithm='tree'. Ties are defined as edges with the same weights=string distances (as determined by the distance matrix for the fully connected network) between nodes already added in the tree and nodes to be added in the tree. There are multiple default and custom configurations for this parameter: 'first' will pick the first edge from a pool of edges of equal string distance (between the sequences) - these are ordered based on each node's expansion/cell count (therefore 'first' will try to add the most expanded node first); 'random' - resolve ties by picking random tied edges; 'close.germline.edges' and 'far.germline.edges' - will prefer the nodes closer or farther to/from the germline, as a number of edges (unweighted) to be next integrated into the network; 'close.germline.distance' and 'far.germline.distance' - picks nodes closer/farther to/from the germline, determined by the string distance; 'close.germline.weighted' and 'far.germline.weighted' - picks edges with nodes closer/farther to/from the germline, as a weighted path from the germline to the most recent integrated node; 'min.expansion' and 'max.expansion' - will pick the most/least expanded sequences; 'min.degree' and 'max.degree' - picks the nodes with the minimum or maximum degree - a distance threshold must be specified in the pruning.threshold parameter, otherwise all nodes will have the same degree; An additional custom configuration can be used: either min/max/specific feature value, tied to a specific feature column as defined in the node.features parameter, using '-'. For example, 'yes-OVA_binder' will select nodes that are OVA binders when resolving ties; for min and max, the node.feature column should be of numeric class. If a vector is provided, ties will be resolved in a hierarchical manner: for example, if resolve.ties=c('max.expansion', 'close.germline.distance'), it will first try to pick nodes with a max expansion that were not added in the network

(with edge ties to those already added), then those closer to the germline (minimum string distance). As these two options do not always fully converge, meaning that there could be also expansion ties and distance ties between the nodes to be added, not just edge ties, a 'first' options is always added at the end of the hierarchical tie resolving algorithm, which always converges/picks a specific edge and resolves a tie. Moreover, the 'first' option is also added when only selecting a single option (still in the form of a vector - for e.g., c('min.expansion') turns automatically into c('min.expansion', 'first')).

`connect.germline.to`

string defining how the germline should be connected for both the pruning and tree building algorithms. When network.algorithm='tree', two options are available: 'min.adjacent' - will first connect the nodes with the min string distance from the germline, then continue adding nodes and building the tree, and 'threshold.adjacent' - will connect nodes with a string distance value lower than the threshold defined in pruning.threshold. As the pruning algorithm starts by pruning all connections out of the specified boundaries, irrespective if they are germline ones or not, the germline needs to be added at the end if it is removed. If not, then this option is ignored. Thus, there are additional options for including the germline when building a network via the pruning algorithm: 'largest.connected.component' - connects the germline to the largest resulting connected component(s), 'all.connected.components' - to all connected components, 'all.connected.components.non.single' - does not connect the germline to single-node components; 'none' - germline is not connected; 'min.adjacent' - connects to the node(s) with the minimum string distance.

`pruning.threshold`

vector of max size=3, specifying the thresholds for the pruning algorithm when network.algorithm includes 'prune' (as seen. 'prune' can be followed by either 'expansion', 'degree' or 'distance', or a combination of them - 'prune.distance.degree'). If we have 'prune.degree', we need to first specify a distance threshold (as 'prune.degree'='prune.distance.degree', otherwise the degree is the same for all nodes in a fully connected network). See also network.algorithm. For a direct use, if network.algorithms is set to prune.distance, set pruning.threshold to a single integer denoting the distance between two nodes for which edges will be pruned (equal or more).

`remove.singletons`

integer - in the case of the pruning network algorithm or 'global' network algorithm, it denotes the minimum connected component node number threshold (for e.g., if remove.singletons = 3, it will remove all nodes from a graph that form a 3-node compnent or less: 2-node and singletons). If NULL, will keep all components (including singletons) in the complex similarity graph.

`keep.largest.cc`

boolean - if T, will only keep the largest connected component in the similarity network (pruning network algorithm or 'global' option for network.algorithm). If F, will keep all components (including singletons unless removed via remove.singletons).

`VDJ.VJ.1chain` boolean, T - excludes cells with an aberrant number of VDJ or VJ chains; F - will be kept in for network inference.

`node.features` string or vector of strings, denoting the column name(s) in the original VDJ/VGM[[1]] from which the node features to be extracted. This is done by first pooling the cell cell_barcodes per unique sequence in the VDJ (for each clonotype), then adding the features of those cells.

`filter.na.features` string or vector of strings, denoting the same column name(s) as specified in node.features. This will remove netowrks were ALL nodes values for the specific feature are equal to NA.

`filter.specific.features` list with two elements - first one denotes the column/feature which you wish to filter on, second denotes the specific feature which HAS TO BE INCLUDED IN THE NETWORK (e.g., list('OVA_binder', 'yes') will result in networks that have at least 1 binder for OVA).

`node.limits` list of integers or NULLs. node.limits[[1]] determines the least amount of unique sequences to create a network for, otherwise a network will not be created. If node.limits[[1]] is NULL, then there is no lower bound for the number of sequences in each network. node.limits[[2]] defines the upper bound for the number of sequences - networks with more sequences will have the extra ones removed, keeping only the most abundant sequences/largest sequence frequency.

`cell.limits` list of integers or NULLs. cell.limits[[1]] the minimum threshold of cells which should produce a unique sequence (sequences below this threshold are removed from the network). If node.limits[[1]] is NULL, then there is no lower bound for the number of cells per unique sequence. node.limits[[2]] defines the upper bound for the number of cells per sequence - sequence frequency.

`weighted.edges` boolean, T - edge weights will be equal to the string distance between a pair of nodes; F - edge weights = 1

`weighted.germline` boolean, T - adds weights to the edges connected to the germline, equal to the string distance between the germline and the specific connected nodes; F - edge weights = 1.

`expand.intermediates` boolean. T - will add inferred, intermediate nodes between nodes in the original network, determined by the string distance between a pair of nodes (for e.g., 2 nodes with an edge=string distance matrix of 3 will result in in 5 total nodes - 3 inferred nodes and 2 original ones, edges=1)

`specific.networks` #either an integer of max sorted clonotypes to be picked for network inference, 'all' for all clonotypes to be used, or list of specific clonotypes to create networks for.

`network.level` string determining the level at which networks should be built - 'intraclonal' will create intraclonal networks = networks for each sample and for each clonotype in the VDJ; 'global.clonotype' will create networks for each unique clonotype, irrespective of sample ids; 'global' will pool all clonotypes from all samples into a single global network; 'forest.per.sample' and 'forest.global' are tree-specific methods, used when obtaining networks via network.algorithm='tree': 'forest.per.sample' will join the intraclonal trees in each sample, 'forest.global' will join ALL trees. Joining is determined by the forest.method parameter.

forest.method	string determining how the trees should be joined if network.level='forest.per.sample' or 'forest.global'. 'single.germline' - trees will all be joined at a single germline (the one from the first clonotype), recalculating the string distances for the new adjacent nodes; 'multiple.germlines' - trees will all be joined in the same network, keeping the original germlines; 'multiple.germlines.joined' - same as 'multiple.germlines', but new edges will be added between the germlines.
random.seed	numeric, seed for the random tie resolving method of resolve.ties.
parallel	boolean with T - a parallelized mclapply will be used for each internal function, to accelerate computation; F - normal lapply will be used. Used best when having a large number of networks/clonotypes per sample.
as.igraph	boolean - if T, the resulting networks will be igraph objects. Otherwise, they are converted to tidygraph tibble objects.

Value

nested list of AntibodyForests objects for each sample and each clonotype. For example, output[[1]][[2]] denotes the AntibodyForests object of the first sample, second clonotype. If only a single clonotype and sample are available in the VDJ (or if the networks are joined via network.level = 'forest.global'), will output a single AntibodyForests object.

Examples

```
## Not run:
AntibodyForests(VDJ, sequence.type='VDJ.VJ.nt.trimmed',
include.germline=T, network.algorithm='tree',
resolve.ties=c('close.germline.distance', 'max.expansion'),
node.features='OVA_binder', expand.intermediates=T, network.level='intraclonal')

## End(Not run)
```

AntibodyForests_communities

Network clustering/community detection for the AntibodyForests similarity networks

---

Description

Performs community detection/clustering on the AntibodyForests sequence similarity networks. Annotates the resulting networks with a new igraph vertex attribute ('community') for downstream analysis or plotting. Can also add these annotations back to the VGM.

Usage

```
AntibodyForests_communities(
  trees,
  VGM,
  community.algorithm,
```

```
  graph.type,
  which.bipartite,
  features,
  count.level,
  additional.parameters
)
```

Arguments

trees	AntibodyForests object/list of AntibodyForests objects - the resulting sequence similarity or minimum spanning tree networks from the AntibodyForests function
VGM	VGM object - for annotating the VGM object with the resulting clusters/communities.
community.algorithm	string - denotes the community/clustering algorithm to be used. Several options are available: 'louvain', 'walktrap', 'edge_betweenness', 'fast_greedy', 'label_prop', 'leading_eigen', 'optimal', 'spinglass'.
graph.type	string - the graph type available in the AntibodyForests object which will be used as the function input. Currently supported network/analysis types: 'tree' (for the minimum spanning trees or sequence similarity networks obtained from the main AntibodyForests function), 'heterogeneous' for the bipartite graphs obtained via AntibodyForests_heterogeneous, 'dynamic' for the dynamic networks obtained from AntibodyForests_dynamics.
which.bipartite	string - whether to perform clustering on the cell layer of the bipartite/heterogeneous graph ('cells'), sequence layer ('sequences') or on both ('both').
features	vector of strings - features to be considered in the output bar plots (of feature counts per cluster). These features must be integrated when creating the initial AntibodyForests objects by using the node.features parameter.
count.level	string - whether to consider cells ('cells') or sequences ('sequences') when counting the unique feature values in the output bar plots. When counting by sequences/nodes, each unique node is assigned the feature value of the majority of its consituent cells.
additional.parameters	named list - additional parameters to be considered in the clustering algorithm, as mentioned in the igraph documentation for the respective algorithms (e.g., additional.parameters = list(resolution = 0.25)).

Value

a single AntibodyForests object or a nested list of AntibodyForests objects (depending on the input type) with community/cluster annotations as a vertex attribute. Additional bar plots of feature counts per resulting cluster are also displayed.

See Also

AntibodyForests, AntibodyForests_plot

Examples

```
## Not run:
AntibodyForests_communities(trees = AntibodyForests_object,
VGM = NULL, community.algorithm = 'louvain',
graph.type = 'tree', features = 'seurat_clusters',
count.level = 'cells', additional.parameters = list(resolution = 0.25))

## End(Not run)
```

AntibodyForests_dynamics

Create a nested list of longitudinal AntibodyForests objects

Description

Adds the dynamic slots to a nested list of AntibodyForests objects outputted from AntibodyForests function. Also inverts the nested list (per clonotype per sample instead of per sample per clonotype) - for tracking the evolution of a specific clonotype across multiple timepoints (samples). The timepoints order can be specified in the timepoint.order parameter. The new dynamic graphs contain all the unique nodes across the timepoints, but with edges created only for a single tree of a given timepoint. The new dynamic slots will be used for downstream analyses - AntibodyForests_metrics(graph.type = 'dynamic') and AntibodyForests_track_nodes. Before running this function, ensure your clonotypes are defined the same way across each timepoint before creating your networks using AntibodyForests (e.g., use the VDJ_call_enclone function or VDJ_clonotype with global.clonotype set to TRUE to ensure clonotype 1 is defined the same across each timepoint, otherwise clonotype1 in timepoint/sample 1 might not correspond to the same clonal definition as clonotype1 in timepoint/sample2).

Usage

```
AntibodyForests_dynamics(trees, graph.type, timepoints.order)
```

Arguments

`trees`	nested list of AntibodyForests objects, as obtained from the AntibodyForests function. Ensure the clonotype definition is consistent across each timepoint before running this function (and before running AntibodyForests to obtain your trees). Also ensure the timepoint ids are present in the sample_id column of your VDJ/VGM[[1]] object.
`graph.type`	string - 'tree' will use the usual output of the AntibodyForests function (tree graphs), 'heterogeneous' will use the output of the AntibodyForests_heterogeneous function (bipartite networks for both cells and sequences).
`timepoints.order`	vector of strings - order of the timepoints in the resulting nested list. For example, output[[1]][[1]] denotes the first clonotype, first timepoint/sample, output[[1]][[2]] denotes the first clonotype, second timepoint/sample,

Value

nested list of AntibodyForests objects for each clonotype and each sample/timepoint. For example, output[[1]][[2]] denotes the AntibodyForests object of the first clonotype, second timepoint.

Examples

```
## Not run:
AntibodyForests_dynamics(trees, graph.type = 'tree', timepoint.order = c('s1', 's2', 's3')

## End(Not run)
```

AntibodyForests_embeddings

Structural node embeddings for the AntibodyForests minimum spanning trees/ sequence similarity networks

---

Description

Structural node embeddings algorithms of the AntibodyForests networks. Supported algorithms include: node2vec (https://arxiv.org/abs/1607.00653) and spectral graph embedding on either the adjacency or the Laplacian matrix. Currently the node2vec model is supported as long as Rkeras is installed.

Usage

```
AntibodyForests_embeddings(
  trees,
  graph.type,
  embedding.method,
  dim.reduction,
  color.by,
  num.walks,
  num.steps,
  p,
  q,
  window.size,
  num.negative.samples,
  embedding.dim,
  batch.size,
  epochs,
  tsne.perplexity,
  seed,
  parallel
)
```

Arguments

trees	AntibodyForests object/list of AntibodyForests objects - the resulting sequence similarity or minimum spanning tree networks from the AntibodyForests function
graph.type	string - the graph type available in the AntibodyForests object which will be used as the function input. Currently supported network/analysis types: 'tree' (for the minimum spanning trees or sequence similarity networks obtained from the main AntibodyForests function), 'heterogeneous' for the bipartite graphs obtained via AntibodyForests_heterogeneous, 'dynamic' for the dynamic networks obtained from AntibodyForests_dynamics.
embedding.method	string - the embeddings model/algorithm. 'node2vec' for an implementation of graph random walk and node2vec using R-keras (might be slow depending on graph size), 'spectral_adjacency' for spectral graph embeddings of the adjacency matrix (using igraph's embed_adjacency_matrix() function), 'spectral_laplacian' for embedding the Laplacian matrix (using igraph's embed_laplacian_matrix() function).
dim.reduction	string - dimensionality reduction algorithm for the resulting node2vec embeddings. Currently implemented methods include: 'umap', 'tsne' and 'pca'.
color.by	vector of strings - features to color the resulting scatter plots by. These features must be included as igraph vertex attributes when creating the AntibodyForests objects, by including them in the node.features parameter.
num.walks	integer - number of biased random walks to be performed for the node2vec training dataset.
num.steps	integer - number of steps per biased random walk.
p	numeric - probability of revisiting the same node already vistied in a random walk step (= return parameter).
q	numeric - probability of 'jumping' to a node closer or farther away from the node visited at step x (e.g., q > 1, random walk is biased to closer nodes, q < 1, random walk will 'jump' to farher nodes more frequently).
window.size	integer - size of sampling window in the skipgram model.
num.negative.samples	integer - number of negative samples to be considered in the skipgram model.
embedding.dim	integer - latent/embedding dimension of the node2vec output vectors.
batch.size	integer - training batch size of the node2vec model.
epochs	integer - number of training epochs for the node2vec model.
tsne.perplexity	numeric - T-SNE reduction perplexity.
seed	integer - random seed for the random walk steps of the node2vec model.
parallel	boolean - whether to execute the random walks in parallel or not.

Value

A scatterplot of reduced vector embeddings for each node in the graphs, colored by the features specified in color.by.

Examples

```
## Not run:
AntibodyForests_embeddings(output_networks,
graph.type = 'tree', embedding.method = 'node2vec',
dim.reduction = 'pca', num.walks = 10, num.steps = 10,
embedding.dim = 64, batch.size = 32, epochs = 50)

## End(Not run)
```

AntibodyForests_expand_intermediates

Infer intermediate nodes in the minimum spanning trees/ sequences similiarity networks created by the AntibodyForests function

---

Description

Intermediate nodes are expanded/inferred based on the edge weight between two existing nodes: for example, of node 1 and node 2 are connected by an edge of weight = 3, 2 new nodes are added in-between and all resulting edges have weight = 1.

Usage

```
AntibodyForests_expand_intermediates(trees, parallel)
```

Arguments

trees	AntibodyForests object/list of AntibodyForests objects - the resulting sequence similarity or minimum spanning tree networks from the AntibodyForests function.
parallel	boolean - whether to execute the main subroutine in parallel or not. Requires the 'parallel' R package.

Value

nested list of AntibodyForests objects or single AntibodyForests object, with the resulting networks having expanded/inferred intermediate nodes.

Examples

```
## Not run:
AntibodyForests_expand_intermediates(trees)

## End(Not run)
```

`AntibodyForests_heterogeneous` *Bipartite sequence-cell networks in AntibodyForests*

Description

Creates a bipartite network from a Seurat object and an already inferred AntibodyForests sequence similarity/ minimum spanning tree network.

Usage

```
AntibodyForests_heterogeneous(
  trees,
  GEX.object,
  node.features,
  cell.graph.type,
  recluster.cells,
  recluster.resolution,
  snn.threshold,
  keep.largest.cc,
  parallel
)
```

Arguments

`trees`	AntibodyForests object/list of AntibodyForests objects - the resulting sequence similarity or minimum spanning tree networks from the AntibodyForests function.
`GEX.object`	Seurat object/ VGM[[2]] for the inferred AntibodyForests networks (must include all cells available in the AntibodyForests object).
`node.features`	vector of strings - gene names in the Seurat object to be added as igraph vertex attributes in the resulting heterogeneous networks (will add gene expression per gene).
`cell.graph.type`	string - graph algorithm for building the cell-cell/ nearest-neighbour graphs, as done by the Seurat::FindNeighbors() function. 'knn' for the K nearest-neighbour graphs, 'snn' for the shared nearest-neighbour graphs.
`recluster.cells`	boolean - whether to recluster the resulting cell graphs or keep the already existing Seurat cluster definitions.
`recluster.resolution`	numeric - recluster resolution for the Louvain algorithm if recluster.cells is TRUE.
`snn.threshold`	numeric - SNN edge weight threshold to further prune edges in the cell graphs (increased value = sparser cell graphs). Defaults to 1/15 (as done in the Seurat::FindNeighbors function).

keep.largest.cc
: boolean - whether to keep only the largest connected component in the cell graphs or keep all singletons/doubletons/etc., as well.

parallel
: boolean - whether to execute the heterogeneous graph building algorithm in parallel or not. Requires the 'parallel' R package.

Value

nested list of AntibodyForests objects for each clonotype and each sample/timepoint or a single AntibodyForests object, with a new added object slot for the heterogeneous graph.

Examples

```
## Not run:
AntibodyForests_heterogeneous(trees, GEX.object = VGM[[2]], cell.graph.type = 'snn')

## End(Not run)
```

AntibodyForests_infer_ancestral

Creates phylogenetic trees, infers ancestral sequences, and converts the resulting trees into igraph objects.

Description

Phylogenetic trees and ancestral sequence reconstruction is performed using the IQ-TREE software. The IQ-TREE directory is required beforehand.

Usage

```
AntibodyForests_infer_ancestral(
  trees,
  alignment.method,
  iqtree.directory,
  collapse.trees,
  parallel
)
```

Arguments

trees
: AntibodyForests object/list of AntibodyForests objects - the resulting sequence similarity or minimum spanning tree networks from the AntibodyForests function.

alignment.method
: string - method/software to perform multiple sequence alignment before the ancestral sequence reconstruction step. Options include: 'mafft' (requires the MAFFT software to be locally installed beforehand), 'clustal', 'clustalomega', 'tcoffee', 'muscle', which all require the 'ape' R package.

`iqtree.directory`
: string - path to the IQ-TREE software directory.

`collapse.trees`
: boolean - if T, will collapse the resulting phylogenetic trees if an intermediate daughter sequence/node is the same as its parent.

`parallel`
: boolean - whether to execute the main subroutine in parallel or not. Requires the 'parallel' R package to be installed.

Value

nested list of AntibodyForests objects or single AntibodyForests object, with a modified tree slot including the phylogenetic tree converted into igraph objects and the reconstructed intermediate/ancestral sequences.

Examples

```
## Not run:
AntibodyForests_infer_ancestral(trees, alignment.method = 'mafft',
igtree.directoty = '/Users/.../Desktop/iqtree-1.6.12-MacOSX')

## End(Not run)
```

`AntibodyForests_join_trees` *Joins a list of trees/networks as AntibodyForests objects into a single AntibodyForests object*

Description

Joins a list of trees/networks as AntibodyForests into a single AntibodyForests object. The resulting network will include all joined networks as separate components. Useful for faster downstream analyses (e.g., node metrics via AntibodyForests_metrics on this object instead on each separate object, plotting multiple trees in the same plot, etc.,)

Usage

```
AntibodyForests_join_trees(tree.list, join.per, join.method)
```

Arguments

`tree.list`
: (nested) list of AntibodyForests objects, as obtained from the AntibodyForests function.

`join.per`
: string - 'sample' joins the objects per sample if the input is a nested list of AntibodyForests objects, resulting in a single joined graph per sample, 'global' joins all graphs in the nested list into a single object.

`join.method`
: string - networks, especially minimum spanning trees, can be joined into a single connected graph if join.method = 'single.germline' (will pick a single germline from all available germlines and will recalculate the string distance) or 'multiple.germlines.joined' (will add inter-germline edges). Will create a single object with disconnected subgraph if join.method = 'multiple.germlines'.

Value

single AntibodyForests object consisting of the joined graphs/trees. Resulting graph can be either a single connected component (if join.method = 'single.germline' or 'multiple.germlines.joined') or multiple disconnected subgraphs (join.method = 'multiple.germlines').

Examples

```
## Not run:
AntibodyForests_join_trees(tree.list, join.per = 'sample', join.method = 'multiple.germlines')

## End(Not run)
```

AntibodyForests_kernels

Graph kernel methods for graph structure/topology comparisons

---

Description

Performs graph structural comparisons using graph kernel-based method. Currently available kernel methods include: the Weisfeiler-Lehman kernel, the graphlet kernel, and the random walk kernel.

Usage

```
AntibodyForests_kernels(
  trees,
  graph.type,
  kernel.method,
  additional.param,
  max.networks
)
```

Arguments

`trees`	(nested) list of AntibodyForests objects, as obtained from the AntibodyForests function, to be compared.
`graph.type`	string - 'tree' will use the usual output of the AntibodyForests function (tree graphs), 'heterogeneous' will use the output of the AntibodyForests_heterogeneous function (bipartite networks for both cells and sequences).
`kernel.method`	string - kernel method to be used, as implemented in the 'graphkernels' R package. 'weisfeiler_lehman' for the Weisfeiler-Lehman kernel, 'graphlet, and 'random_walk'.
`additional.param`	integer - additional kernel options/parameters (e.g., kernel iterations for the Weisfeiler-Lehman kernel).
`max.networks`	integer - maximum number of networks to be compared (will pick the networks with the most number of cells first).

Value

Heatmap of the graph kernel values.

Examples

```
## Not run:
AntibodyForests_kernels(trees, graph.type = 'tree',
kernel.method = 'weisfeiler_lehman',
additional.params = 10, max.networks = 50)

## End(Not run)
```

`AntibodyForests_label_propagation`

Propagate label annotations/values on sparsely labeled networks as AntibodyForests objects.

Description

Performs label diffusion/propagation, using two different algorithms: if propagation.algorithm = 'diffusion', will perform label propagation using the graph heat diffusion method (http://mlg.eng.cam.ac.uk/zoubin/papers/CM CALD-02-107.pdf), 'neighbours' for neihbour majority voting propagation (https://arxiv.org/abs/0709.2938).

Usage

```
AntibodyForests_label_propagation(
  trees,
  features,
  propagation.algorithm,
  diffusion.n.iter,
  diffusion.threshold,
  parallel
)
```

Arguments

`trees`	nested list of AntibodyForests objects or single object, as obtained from the AntibodyForests function.
`features`	vector of strings - features to be propagated in the graph. Must be performed on sparsely-labeled graphs (with NA node attribute values).
`propagation.algorithm`	string - label propagation/diffusion algorithm to be used. If propagation.algorithm = 'diffusion', will perform label propagation using the graph heat diffusion method (http://mlg.eng.cam.ac.uk/zoubin/papers/CMU-CALD-02-107.pdf), 'neighbours' for neihbour majority voting propagation (https://arxiv.org/abs/0709.2938).
`diffusion.n.iter`	integer - number of diffusion iteration if propagation.algorithm = 'diffusion'.

`diffusion.threshold`
: numeric - probability min. threshold for the diffusion algorithm.

`parallel`
: boolean - whether to execute the main subroutine in parallel or not. Requires the 'parallel' R package to be installed.

Value

Nested list of AntibodyForests objects or single object with new propagated labels added as vertex attributes (e.g., feature_label_propagation will be a new vertex attribute in the resulting AntibodyForests objects).

Examples

```
## Not run:
AntibodyForests_label_propagation(ova_trees,
features = 'OVA_binder',
propagation.algorithm = 'diffusion', parallel = T)

## End(Not run)
```

AntibodyForests_metrics

Node metrics for the AntibodyForests sequence similarity networks and minimum spanning trees.

Description

Calculates several node metrics for the resulting AntibodyForests networks, adding these as a per-node dataframe in the new metrics slot. Must be called before AntibodyForests_plot_metrics.

Usage

```
AntibodyForests_metrics(
  trees,
  metrics,
  graph.type,
  features,
  exclude.intermediates,
  exclude.germline,
  separate.bipartite,
  parallel
)
```

Arguments

trees	nested list of AntibodyForests objects or single object, as obtained from the AntibodyForests function.
metrics	vector of strings - node metric to be calculated. Currently supported metrics include: 1. Betweenness centrality - 'betweenness' 2. Closeness centrality - 'closeness' 3. Eigenvector centrality - 'eigenvector' 4. Authority score - 'authority' 5. Local and average clustering coefficients - 'local_cluster_coefficient', 'average_cluster_coefficient' 6. Strength or weighted vertex degree - 'strength' 7. Degree - 'degree' 8. Daughter nodes for directed graphs - 'daughters' 9. Vertex eccentricity (shortest path distance from the farthest other node in the graph) - 'eccentricity' 10. Pagerank algorithm values - 'pagerank' 11. Shortest (weighted) paths from germline - 'path_from_germline', 'weighted_path_from_germline' 12. Shortest(weighted) paths from the most expanded node - 'path_from_most_expanded', 'weighted_path_from_most_expanded', 13. Shortest(weighted) paths from the hub node (highest degree) - 'path_from_hub', 'weighted_path_from_hub'
graph.type	string - the graph type available in the AntibodyForests object which will be used as the function input. Currently supported network/analysis types: 'tree' (for the minimum spanning trees or sequence similarity networks obtained from the main AntibodyForests function), 'heterogeneous' for the bipartite graphs obtained via AntibodyForests_heterogeneous, 'dynamic' for the dynamic networks obtained from AntibodyForests_dynamics.
features	vector of strings - features to be considered when calculating whole-graph metrics (e.g., dominant/most abundant feature per graph).
exclude.intermediates	boolean - if T, will not calculate the node-level metrics for the intermediate nodes obtained from AntibodyForests_expand_intermediates().
exclude.germline	boolean - if T, will exclude the germline nodes from the node metric calculations.
separate.bipartite	boolean - if T and graph.type = 'heterogeneous', will separate the cell and sequence graph and calculate the node metrics independently, then recombine them in the same final graph.
parallel	boolean - whether to execute the main subroutine in parallel or not. Requires the 'parallel' R package to be installed.

Value

nested list of AntibodyForests objects or single AntibodyForests object, with a new class slot added (metrics) = a per-node dataframe of metric values.

Examples

```
## Not run:
AntibodyForests_metrics(trees, graph.type = 'tree', metrics = c('degree', 'pagerank'))

## End(Not run)
```

`AntibodyForests_node_transitions` *Calculates the node transitions frequencies for a given feature and an AntibodyForests object*

Description

Node transitions represent the number of (un)directed edges between two feature values of a given feature type in a single sequence similarity network or minimum spanning trees (e.g., between VDJ_cgene = 'IGHA' and VDJ_cgene = 'IGHM'). The resulting AntibodyForests objects will contain a new slot - node_transitions. Will also output bar plots of the transition frequencies.

Usage

```
AntibodyForests_node_transitions(
  trees,
  features,
  combined,
  graph.type,
  permutation.test,
  exclude.germline,
  exclude.intermediates,
  n.permutations,
  plot.results,
  parallel
)
```

Arguments

`trees`	nested list of AntibodyForests objects or single object, as obtained from the AntibodyForests function.
`features`	vector of strings - features for the node transition counting. Each node will be assigned the dominant feature if its constituent cells have different feature values (e.g., for features = c('seurat_clusters')). These features need to be first added as vertex attributes when creating the AntibodyForests networks, specified in the node.features parameter of AntibodyForests.
`combined`	boolean - if T, will assign a new feature by combining the feature values as specified in the features parameter (e.g., if node 1 has VDJ_cgene = 'IGHM' and seurat_clusters = 1, will create a new feature = 'IGHM 1' for that node, for joint node transitions).
`graph.type`	string - the graph type available in the AntibodyForests object which will be used as the function input. Currently supported network/analysis types: 'tree' (for the minimum spanning trees or sequence similarity networks obtained from the main AntibodyForests function), 'heterogeneous' for the bipartite graphs obtained via AntibodyForests_heterogeneous, 'dynamic' for the dynamic networks obtained from AntibodyForests_dynamics.

`permutation.test`
: boolean - if T, will perform a permutation statistical test on the node feature transition values.

`exclude.germline`
: boolean - if T, will exclude the germline from the node feature transitions counting.

`exclude.intermediates`
: boolean - if T, will exclude the intermediate nodes expanded using AntibodyForests_expand_intermediates() from the node transitions counting.

`n.permutations`
: integer - number of node feature permutations for the permutation test.

`plot.results`
: boolean - if T, will display the node transitions counts as bar plots, per feature specified in the features parameter.

`parallel`
: boolean - whether to execute the main subroutine in parallel or not. Requires the 'parallel' R package to be installed.

Value

nested list of AntibodyForests objects for each clonotype and each sample or single object, with an additional node_transitions slot of transition/edge counts.

Examples

```
## Not run:
AntibodyForests_node_transitions(trees,
graph.type = 'tree', features = 'VDJ_cgene',
plot.results = T)

## End(Not run)
```

AntibodyForests_overlap

Edge overlap heatmaps for a set of AntibodyForests sequence similarity networks or minimum spanning trees.

Description

Similar to the AntibodyForests_node_transitions function, will calculate the incidence of features across undirected edges. In this case, each edge will be considered a unique species - with incidence counts across each unique feature value if a specific edge is connected to a node with that feature. Overlap metrics are then calculated for this edge-feature incidence matrix.

Usage

```
AntibodyForests_overlap(trees, group.by, method)
```

Arguments

trees	nested list of AntibodyForests objects, as obtained from the AntibodyForests function.
group.by	vector of strings - node features to group the edges by (counts edge incidence across the unique feature values for the specified node feature).
method	string - overlap calculator: 'overlap' for unique/public edge counts across the feature values, 'jaccard' to calculate the Jaccard index.

Value

Edge overlap heatmaps for the specific overlap metric/method.

Examples

```
## Not run:
AntibodyForests_overlap(trees, group.by = 'seurat_clusters', method = 'jaccard')

## End(Not run)
```

AntibodyForests_paths *Calculates the longest/shortest paths from a node to a given node for the AntibodyForests minimum spanning trees / sequence similarity networks*

Description

Calculates the longest or shortest paths in a given AntibodyForests graph from a node to another given node. Nodes can be specified as integers (e.g., path.from = 5, picking the fifth node in the igraph vertex list) or by predetermined attributes (e.g., path.from = 'germline' and path.to = 'hub' will calculate the paths between all germlines and hubs in a set of networks. Moreover, there is an option to select paths for nodes given specific node features (e.g., path.from = list('seurat_clusters', '1') and path.to = 'hub' will infer the paths from the nodes with a majority of Seurat clusters = 1, and to the hub nodes).

Usage

```
AntibodyForests_paths(
  trees,
  graph.type,
  path.from,
  path.to,
  paths,
  interlevel.from,
  weighted,
  plot.results,
  color.by,
  cell.frequency,
```

```
  parallel
)
```

Arguments

trees	nested list of AntibodyForests objects or single object, as obtained from the AntibodyForests function.
graph.type	string - the graph type available in the AntibodyForests object which will be used as the function input. Currently supported network/analysis types: 'tree' (for the minimum spanning trees or sequence similarity networks obtained from the main AntibodyForests function), 'heterogeneous' for the bipartite graphs obtained via AntibodyForests_heterogeneous, 'dynamic' for the dynamic networks obtained from AntibodyForests_dynamics.
path.from	string/list of strings/integer - starting nodes for a path. Options are either an integer, selecting the nth most abundant node, 'hub' to select the hub nodes, 'most_expanded' for the nodes with most cells, 'leaf' for leaf nodes, 'germline' for the germline node, or a list of the form list(feature, feature_value) to select nodes with a specific feature value.
path.to	string/list of strings/integer - end nodes for a path. Options are either an integer, selecting the nth most abundant node, 'hub' to select the hub nodes, 'most_expanded' for the nodes with most cells, 'leaf' for leaf nodes, 'germline' for the germline node, or a list of the form list(feature, feature_value) to select nodes with a specific feature value.
paths	string - whether to calculate the longest path ('longest'), shortest path ('shortest'), or both (c('longest', 'shortest'))
interlevel.from	string/list of strings/integer - starting nodes for an interlevel path for the bipartite/heterogeneous networks (if graph.type = 'heterogeneous'). Options are either an integer, selecting the nth most abundant node, 'hub' to select the hub nodes, 'most_expanded' for the nodes with most cells, 'leaf' for leaf nodes, 'germline' for the germline node, or a list of the form list(feature, feature_value) to select nodes with a specific feature value.
weighted	boolean - whether to calculate the weighted or unweighted shortest/longest paths.
plot.results	boolean - if T, will output a bar plot of node feature counts per path (of all nodes in a given path). Features are determined by the color.by parameter.
color.by	string - features for the feature count per path bar plots if plot.results is set to T.
cell.frequency	boolean - whether to consider the node or cell frequency for the feature counts in the resulting bar plot if plot.results is T.
parallel	boolean - whether to execute the main subroutine in parallel or not. Requires the 'parallel' R package to be installed.

Value

nested list of AntibodyForests objects or a single object with a new slot - paths. If plot.results is T, will also output a bar plot of feature counts per path (considering all node in a given path).

Examples

```
## Not run:
AntibodyForests_paths(trees, graph.type = 'tree',
path.from = 'germline', path.to = 'leaf',
plot.results = T, color.by = 'seurat_clusters')

## End(Not run)
```

`AntibodyForests_phylo` *Converts the igraph networks of a given AntibodyForests object into a given (useful to convert the minimum spanning trees into a phylogenetic tree)*

Description

Will automatically convert the minimum spanning trees in a given AntibodyForests object into a phylogenetic tree as a phylo object. This new object will be added into the phylo slot of the AntibodyForests object.

Usage

```
AntibodyForests_phylo(trees, output.format, solve.multichotomies, parallel)
```

Arguments

`trees`	nested list of AntibodyForests objects or single object, as obtained from the AntibodyForests function.
`output.format`	string - 'treedata' will output the phylogenetic tree as a tidytree treedata object, 'phylo' as an ape::phylo object.
`solve.multichotomies`	boolean - whether to remove multichotomies in the resulting phylogenetic tree using ape::multi2di
`parallel`	boolean - whether to execute the main subroutine in parallel or not. Requires the 'parallel' R package to be installed.

Value

nested list of AntibodyForests objects for each clonotype and each sample/timepoint or a single object, with a new phylo slot for the phylogenetic tree.

Examples

```
## Not run:
AntibodyForests_phylo(trees, output.format = 'phylo')

## End(Not run)
```

AntibodyForests_plot *Custom plots for trees/networks created with AntibodyForests*

Description

AntibodyForests_plot takes the input of AntibodyForests and outputs a list of plot-ready graphs (inside the AntibodyForests object) to be further used with plot(). Plots can also be automicatically saved to pdf via the save.pdf parameter. The resulting igraph object have their node/edge colors/shapes/sizes added following the specific parameters in the AntibodyForests_plot function.

Usage

```
AntibodyForests_plot(
  network.list,
  graph.type,
  node.color,
  node.label,
  node.shape,
  node.size,
  max.node.size,
  node.scale.factor,
  edge.length,
  edge.width,
  path.list,
  specific.node.colors,
  specific.node.shapes,
  specific.edge.colors,
  color.by.majority,
  cell.color,
  specific.cell.colors,
  cell.size,
  network.layout,
  save.pdf,
  save.dir,
  show.legend,
  color.gradient
)
```

Arguments

network.list	nested list of igraph objects, as obtained from the AntibodyForests function. input[[1]][[2]] represents the igraph object for the first sample, second clonotype, if the normal AntibodyForests parameters are used.
graph.type	string - graph to be plotted - 'tree' for the normal tree plots, 'heterogeneous' for the single-cell grahs (call AntibodyForests_heterogeneous before), 'phylo' to plot the phylo objects (call AntibodyForests_phylo before), 'dynamic' for the dynamic/temporal graphs.

node.color string specifying the name of the igraph vertex attribute (or the original vgm[[1]]/VDJ column name used in the node.features parameter of AntibodyForests) to be used for coloring the nodes. If the node.shape is also 'pie', the resulting nodes will be a pie chart of the per-node values denoted by the node.color parameter (if there are multiple different values per node - e.g., a node denotes a sequence, which can be further traced back to multiple cells with different barcodes and with potentially different transcriptomic clusters - multiple feature values per node). If the node.color parameter is NULL, the default node color will be '#FFCC00' (yellow) for sequence nodes, lighter gray for intermediate/inferred nodes, and darker gray for germline nodes.

node.label string - 'cells' to label the nodes by the number of cells with that specific sequence, 'rank' for cell count-based ranking of the nodes. If NULL, will not add number labels to the sequence nodes.

node.shape string specifying the the name of the igraph vertex attribute (or the original vgm[[1]]/VDJ column name used in the node.features parameter of AntibodyForests) to be used for node shapes. Shapes will be assigned per unique value from the values of this column/vertex attribute. There is a maximum of 7 unique shapes to be chosen from, therefore node.shapes should be used for features with less than 7 unique values. If the node.shape parameter is null, the node shape for all nodes will default to 'circle'

node.size string denoting either a specific method of scaling the node sizes of the input graph or a specific vertex attribute of numeric values (added via the node.features parameter of the AntibodyForests from the original vgm[[1]]/VDJ dataframe) to be used for node sizes. If NULL, then the sizes will be equal to node.scale.factor * 1. If scaleByEigen, node sizes will be scaled by the eigenvector centrality of each node; scaleByCloseness - closeness centrality; scaleByBetweenness - betweenness centrality; scaleByExpansion - by the sequence frequency of each node, as originally calculated by the AntibodyForests function.

max.node.size Maximum size of any given node

node.scale.factor integer to further refine the size of each node. Each vertex size will be multiplied by scaling factor.

edge.length either NULL - edge lengths are constant, scaleByWeight - edge lengths will be scaled by the edge weight attribute (the string distance between pairs of sequences/nodes, as calculated by the AntibodyForests function) - larger weights/string distances = longer edges/nodes further apart, or a specific igraph edge attribute name.

edge.width either NULL - edge widths are constant, scaleByWeight - edge widths will be scaled by the edge weight attribute (the string distance between pairs of sequences/nodes, as calculated by the AntibodyForests function) - larger weights/string distances = thinner edges), or a specific igraph edge attribute name.

path.list named list of igraph paths, as obtained from the AntibodyForests_metrics function.

specific.node.colors named list of colors to be used for the node.color parameter. If NULL, colors will be automatically added for each unique value. For example, if spe-

cific.node.colors=list('yes'='blue', 'no'='red'), then the nodes labeled as 'yes' will be colored blue, the others red.

`specific.node.shapes`
: named list of node shapes to be used for the node.shapes parameter. Must be shapes compatible with igraph objects, use setdiff(igraph::shapes(), "") to get a list of possible values. For example, if specific.node.shapes=list('yes'='circle', 'no'='square'), then the nodes labele as 'yes' will be circles, the others squares.

`specific.edge.colors`
: named list of edge colors to be used for the edge.colors parameter. The names should be the path metrics names obtained from the nested paths list, from AntibodyForests_metrics. For example, if specific.edge.colors=list('longest.path.weighted'='blue', 'shortest.path.unweighted'='red'), the longest weighted paths obtained from AbtibodyForests will be colored blue for each igraph object, the rest will be red.

`color.by.majority`
: boolean - if T, will color the entire network (all nodes) by the dominant/most frequent node feature, as specified in the node.color parameter.

`cell.color`
: string - cell feature column denoting the cell colors - as denoted by the node.features parameter when calling AntibodyForests_heterogeneous.

`specific.cell.colors`
: named list of cell colors and their features (e.g., for Seurat clusters: list(1 = 'red', 2 = 'blue')). Optional (will auto search for unique colors per feature).

`cell.size`
: integer denoting the size of the cell nodes.

`network.layout`
: either NULL - will default to the automatic igraph::layout_nicely(), 'fr' - igraph::layout_with_fr() - for fully connected graphs/graphs with defined connected components, 'tree' - for tree graphs, as obtained from AntibodyForests with network.algorithm='tree'.

`save.pdf`
: boolean - if T, plots will be automatically saved to pdf, in the current working directory; F - normal output of the function (plot-ready igraph object and specific layout). New folders will be created for each sample of the input nested list of igraph objects.

`save.dir`
: path to the directory oin which the network PDFs will be saved.

`show.legend`
: boolean - whether the legend should be showed in the resulting plots

`color.gradient`
: string - defualt: NULL - the feature whose value will be plot as a color gradient

Value

nested list of plot-ready AntibodyForests objects. Can also save the plots as a PDF file.

See Also

AntibodyForests, AntibodyForests_metrics

Examples

```
## Not run:
AntibodyForests_plot(graphs, node.color='clonotype_id',
node.size='scaleByExpansion', network.layout='tree',
```

```
save.pdf=T)

## End(Not run)
```

`AntibodyForests_plot_metrics` *Plots the resulting node metrics from the AntibodyForests_metrics function*

Description

Will plot the resulting node metrics from the AntibodyForests_metrics function either as violin plots (plot.format = 'violin' or as a scatter plot of the principal components of the metrics dataframe - as long as multiple node metrics are calculated per node). Requires the AntibodyForests_plot_metrics to be called before (as this function uses the resulting node_metrics dataframes).

Usage

```
AntibodyForests_plot_metrics(
  trees,
  plot.format,
  metrics.to.plot,
  group.by,
  max.groups,
  specific.groups,
  sample.by,
  features
)
```

Arguments

`trees`	nested list of AntibodyForests objects or single object, as obtained from the AntibodyForests function.
`plot.format`	string - 'violin' for violin plots of node metrics per feature (as determined by the feature parameter), or 'pca' for performing PCA on the node metrics dataframe.
`metrics.to.plot`	vector of strings - the metrics to be plotted from the metrics dataframe (must be already calculated using the AntibodyForests_metrics function)
`group.by`	string - whether to group the violin/scatter plots by additional features (e.g., group.by = 'sample_id' to get violin plots per feature, per each unique sample).
`max.groups`	integer - maximum number of groups to be considered in the resulting plots if group.by is not NULL.
`specific.groups`	vector of strings - specific groups to plot if group.by is not NULL.
`sample.by`	string - additional grouping factor (e.g., group.by can be set to 'clonotype_id' and sample.by to 'sample_id' for plots grouped by both clonotypes and samples)
`features`	string - will determine the point colors in the PCA scatterplot.

Value

either a violin plot or a scatter plot of the node metrics, as specified in the plot.format parameter

Examples

```
## Not run:
AntibodyForests_plot_metrics(trees,
plot.format = 'violin', metrics.to.plot = 'degree',
group.by = 'sample_id', sample.by = NULL)

## End(Not run)
```

---

automate_GEX *GEX processing wrapper in Platypus V2*

Description

Automates the transcriptional analysis of the gene expression libraries from cellranger. This function will integrate multiple samples

Usage

```
automate_GEX(
  GEX.outs.directory.list,
  GEX.list,
  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,
  groups
)
```

Arguments

`GEX.outs.directory.list`
: The path to the output of cellranger vdj runs. Multiple repertoires to be integrated together should be supplied as a character vector in the first element of a list. For example, if two separate VDJ repertoires should be integrated together (e.g. on the same tSNE plot), GEX.outs.directory.list[[1]] <- c("my.VDJ1.path/outs/","my.VDJ2.path/outs should be stored as input. If these repertoires should be analyzed separately,

>GEX.outs.directory.list[[1]] <- "my.VDJ1.path/outs/" >GEX.outs.directory.list[[2]] <- "my.VDJ2.path/outs/" should be supplied.. This can be left blank if supplying the clonotypes and all_contig files diretly as input. Multiple analyses can be stored

`GEX.list` List containing the output from Seurat Read10x. This must be supplied if GEX.out.directory is not provided.

`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. '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'.

`groups` Integer specifying the groups of the different samples. This is needed if there are multiple biological replicates for a given condition sequenced and aligned through cellranger separately.

Value

Returns a processed Seurat object containing transcriptional information from all samples which can be supplied to the VDJ_GEX_integrate function.

Examples

```
## Not run:
automate_GEX(out_directory=fullRepertoire.output,
 rep.size=3*length(unlist(fullRepertoire.output[[1]])),
 distribution="identical",
 with.germline="FALSE")

## End(Not run)
```

Bcell_sequences_example_tree

Example csv file 1

Description

Example csv file 1

Usage

```
Bcell_sequences_example_tree
```

Format

An object of class `data.frame` with 170 rows and 1 columns.

References

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

Bcell_tree_2 *Example csv file 2*

Description

Example csv file 2

Usage

```
Bcell_tree_2
```

Format

An object of class `data.frame` with 85 rows and 1 columns.

References

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

call_MIXCR *Calls MiXCR VDJ object of Platypus V2*

Description

Extracts information on the VDJRegion level using MiXCR. This function assumes the user can run an executable instance of MiXCR and is elgible to use MiXCR as determined by license agreements. The VDJRegion corresponds to the recombined heavy and light chain loci starting from framework region 1 (FR1) and extending to frame work region 4 (FR4). This can be useful for extracting full-length sequences ready to clone and further calculating somatic hypermutation occurances.

Usage

```
call_MIXCR(VDJ.per.clone, mixcr.directory, species)
```

Arguments

`VDJ.per.clone`	The output from the VDJ_per_clone function. This object should have information regarding the contigs and clonotype_ids for each cell.
`mixcr.directory`	The directory containing an executable version of MiXCR. This must be downloaded separately and is under a separate license.
`species`	Either "mmu" for mouse or "hsa" for human. These use the default germline genes for both species contained in MIXCR.

Value

Returns a nested list containing VDJRegion information as determined by MIXCR. The outer list corresponds to the individual repertoires in the same structure as the input VDJ.per.clone. The inner list corresponds to each clonal family, as determined by either the VDJ_clonotype function or the defaul nucleotide clonotyping produced by cellranger.Each element in the inner list corresponds to a dataframe containing repertoire information such as isotype, CDR sequences, mean number of UMIs. This output can be supplied to further package functions such as VDJ_extract_sequences and VDJ_GEX_integrate.

See Also

VDJ_extract_sequences

Examples

```
## Not run:
call_MIXCR(VDJ.per.clone = VDJ.per.clone.output
,mixcr.directory = "~/Downloads/mixcr-3.0.12/mixcr",species = "mmu")

## End(Not run)
```

CellPhoneDB_analyse *Cellphone DB utility*

Description

Function to set up the data so that it can be read and processed by CellPhoneDB, which saves the results of the analysis in a directory "out" and adds them in a new vgm (output of VDJ_GEX_matrix function) list item (CellPhoneDB). Needs Python to be installed in the computer. Running time can take some minutes. Depending on the state of the connection to ensembl and whether this is down, the function might not work if it needs to convert the genes identity. In these cases, try at some other moment and the connection should hopefully be back. !!! In case the python call does is not executed, please refer to the parameter "pre.code" !!!

Usage

```
CellPhoneDB_analyse(
  vgm.input,
  column,
  groups,
  organism,
  gene.id,
  analysis.method,
  project.name,
  iterations,
  threshold,
  result.precision,
```

```
  subsampling,
  subsampling.num.pc,
  subsampling.num.cells,
  subsampling.log,
  pvalue,
  debug.seed,
  threads,
  install.cellphonedb,
  pre.code,
  platypus.version
)
```

Arguments

vgm.input	Output of the VDJ_GEX_matrix function
column	Character vector. Mandatory. Column name of VDJ_GEX_matrix[[2]] where the groups to be tested for interactions are located
groups	Strings vector. Mandatory. Vector with groups of the indicated column to be compared.
organism	Character vector. Defaults to "human". If == "mouse" the function converts the gene's mouse names into the human ones.
gene.id	Character vector. Defaults to "ensembl". Possible arguments: "ensembl" , "hgnc_symbol", "gene_name". Indicates the gene ID used by CellPhoneDB during the analysis. CellPhoneDB specific method argument.
analysis.method	Character vector. Defaults to "statistical_analysis" Possible arguments: "statistical_analysis". CellPhoneDB is developing also "degs_analysis" method, which will be included among the possible arguments once released. Indicates the analysis method used by CellPhoneDB. CellPhoneDB specific method argument.
project.name	Character vector. Defaults to NULL. If a name is given by the user, a subfolder with this name is created in the output folder. CellPhoneDB specific method argument.
iterations	Numerical. Defaults to 1000. Number of iterations for the statistical analysis. CellPhoneDB specific method argument.
threshold	Numerical. By defaults not specified. % of cells expressing the specific ligand/receptor. Range: 0<= threshold <=1.
result.precision	Numerical. Defaults to 3. Number of decimal digits in results. CellPhoneDB specific method argument.
subsampling	Logical. Defaults to FALSE. If set to TRUE it enables subsampling. CellPhoneDB specific method argument.
subsampling.num.pc	Numerical. Defaults to 100, if subsampling == TRUE. Number of PCs to use. CellPhoneDB specific method argument.

`subsampling.num.cells`
: Numerical. Defaults to 1/3 of cells, if subsampling == TRUE. Number of cells to subsample. CellPhoneDB specific method argument.

`subsampling.log`
: Logical. No default, mandatory when subsampling. Enables subsampling log1p for non log-transformed data inputs. CellPhoneDB specific method argument.

`pvalue`
: Numerical. Defaults to 0.05. P-value threshold. CellPhoneDB specific statistical method argument.

`debug.seed`
: Numerical. Deafults to -1. Debug random seed. To disable it use a value >=0. CellPhoneDB specific statistical method argument

`threads`
: Numerical. Defaults to -1. Number of threads to use (needs to be >=1). CellPhoneDB specific statistical method argument.

`install.cellphonedb`
: Logical. Defaults to TRUE. Installs the CellPhoneDB Python package if set ==TRUE.

`pre.code`
: Character string. One command line or multiple command lines separated by " && " of code to execute in the console before cellphonedb is called. In case Cellphonedb is installed within a Conda enviroment (highly recommended), set this to "conda activate MyEnviroment", "activate MyEnviroment" or "conda init cmd.exe && activate MyEnviroment" depending on your Conda installation.

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

Value

VDJ_GEX_matrix object with additional list item (VDJ_GEX_matrix[[10]]), containing results and plots of CellPhoneDB analysis. Saves in the directory the input files, results and plots of CellPhoneDB analysis.

Examples

```
## Not run:
vgm_cellphonedb<-CellPhoneDB_analyse(vgm.input=vgm_m,
organism="mouse", groups=c(3,6,9),
gene.id="ensembl",
analysis.method= "statistical_analysis",
install.cellphonedb = FALSE,
subsampling= TRUE,
subsampling.num.pc=100,
subsampling.num.cells=70,
subsampling.log=TRUE,
project.name = "test")

## End(Not run)
```

class_switch_prob_hum *class_switch_prob_hum The probability matrix of class switching for human b cells. The row names of the matrix are the isotypes the cell is switching from, the column names are the isotypes the cell is switching to. All B cells start from IGHM, and switch to one of the other isotypes or remain the same.*

Description

class_switch_prob_hum The probability matrix of class switching for human b cells. The row names of the matrix are the isotypes the cell is switching from, the column names are the isotypes the cell is switching to. All B cells start from IGHM, and switch to one of the other isotypes or remain the same.

Usage

```
data("class_switch_prob_hum")
```

Format

A 8*8 matrix.The row and clumn names are "IGHM","IGHD","IGHG1","IGHG2","IGHG3","IGHG4","IGHE","IGHA". The probability for a cell to switch from "IGHM" to "IGHD" is the value at class_switch_prob_hum[1,2].

class_switch_prob_mus *class_switch_prob_mus The probability matrix of class switching for mouse b cells. The row names of the matrix are the isotypes the cell is switching from, the column names are the isotypes the cell is switching to. All B cells start from IGHM, and switch to one of the other isotypes or remain the same.*

Description

class_switch_prob_mus The probability matrix of class switching for mouse b cells. The row names of the matrix are the isotypes the cell is switching from, the column names are the isotypes the cell is switching to. All B cells start from IGHM, and switch to one of the other isotypes or remain the same.

Usage

```
data("class_switch_prob_mus")
```

Format

A 9*9 matrix.The row and clumn names are "IGHM","IGHD","IGHG1","IGHG2A","IGHG2B","IGHG2C","IGHG3","IGHE
The probability for a cell to switch from "IGHM" to "IGHD" is the value at class_switch_prob_mus[1,2].

clonofreq *Plot clonal frequency barplot of the outout simulated data*

Description

Plot the top abundant clonal frequencies in a barplot with ggplot2

Usage

```
clonofreq(clonotypes, top.n, y.limit)
```

Arguments

clonotypes	The clonotypes dataframe, which is the second element in the simulation output list.
top.n	The top n abundant clones to be shown in the plot. If missing, all clones will be shown.
y.limit	The upper limit for y axis in the plot.

Value

top abundant clonal frequencies in a barplot with ggplot2

clonofreq.isotype.data *Get information about the clonotype counts grouped by isotype.*

Description

Return

Usage

```
clonofreq.isotype.data(all.contig.annotations, top.n)
```

Arguments

all.contig.annotations	The output dataframe all_contig_annotation from function simulate.repertoire.
top.n	The top n abundant clones to be shown in the plot. If missing, all clones will be shown.

Value

dataframes containing the top n abundant clonotypes and their frequency and isotype information for further processing.

clonofreq.isotype.plot *Get information about the clonotype counts grouped by isotype.*

Description

Plot a stacked barplot for clonotype counts grouped by isotype.

Usage

```
clonofreq.isotype.plot(all.contig.annotations, top.n, y.limit, colors)
```

Arguments

all.contig.annotations	The output dataframe all_contig_annotation from function simulate.repertoire.
top.n	The top n abundant clones to be shown in the plot. If missing, all clones will be shown.
y.limit	The upper limit for y axis in the plot.
colors	A named character vector of colors, the names are the isotypes. If missing, the default has 11 colors coresponding to the default isotype names.

Value

a stacked barplot for clonotype counts grouped by isotype

clonofreq.trans.data *Get information about the clonotype counts grouped by transcriptome state(cell type).*

Description

Dataframe with clonotype counts grouped by transcriptome state(cell type).

Usage

```
clonofreq.trans.data(all.contig.annotations, history, trans.names, top.n)
```

Arguments

all.contig.annotations	The output dataframe all_contig_annotation from function simulate.repertoire.
history	The dataframe history from simulate output.
trans.names	The names of cell types which are used in transcriptome.switch.prob argument in the simulation.
top.n	The top n abundant clones to be shown in the plot. If missing, all clones will be shown.

Value

a dataframe with clonotype counts grouped by transcriptome state(cell type).

`clonofreq.trans.plot` *Get information about the clonotype counts grouped by transcriptome state(cell type).*

Description

Plot a stacked barplot for clonotype counts grouped by transcriptome state(cell type).

Usage

```
clonofreq.trans.plot(
  all.contig.annotations,
  history,
  trans.names,
  top.n,
  y.limit,
  colors
)
```

Arguments

`all.contig.annotations`	The output dataframe all_contig_annotation from function simulate.repertoire.
`history`	The dataframe history from simulate output.
`trans.names`	The names of cell types which are used in transcriptome.switch.prob argument in the simulation.
`top.n`	The top n abundant clones to be shown in the plot. If missing, all clones will be shown.
`y.limit`	The upper limit for y axis in the plot.
`colors`	A named character vector of colors, the names are the isotypes. If missing, the default has 11 colors coresponding to the default isotype names.

Value

a stacked barplot for clonotype counts grouped by transcriptome state(cell type).

cluster.id.igraph *Get clone network igraphs colored by seurat cluster id.*

Description

Get clone network igraphs colored by seurat cluster id.

Usage

```
cluster.id.igraph(meta.data, history, igraph.index, empty.node)
```

Arguments

meta.data	the meta.data dataframe from the Seurat object of the simulation. The object should be pre-processed and has cluster ids in the meta.data.
history	The dataframe 'history' from the simulation output.
igraph.index	The list 'igraph.index' from the simulation output.
empty.node	If TRUE, there will be empty node in igraph. if FALSE, the empty node will be deleted.

Value

a list of clone network igraphs colored by seurat cluster id

colors *colors A vector of characters specifying colors used in igraph phylogenetic tree. Default colors: "#66C2A5", "#FC8D62", "#8DA0CB", "#E78AC3" ,"#A6D854"*

Description

colors A vector of characters specifying colors used in igraph phylogenetic tree. Default colors: "#66C2A5", "#FC8D62", "#8DA0CB", "#E78AC3" ,"#A6D854"

Usage

```
data("colors")
```

Format

a character vector

dot_plot *Function to cutomise the Dot Plot of CellPhoneDB analysis results.*

Description

Function to cutomise the Dot Plot of CellPhoneDB analysis results.

Usage

```
dot_plot(
  vgm.input,
  selected.rows,
  selected.columns,
  threshold.type,
  threshold.value,
  project.name,
  filename,
  width,
  height,
  text.size,
  return.vector,
  platypus.version
)
```

Arguments

vgm.input	Output of the VDJ_GEX_matrix function. Mandatory. Object where to save the dotplot.
selected.rows	Strings vector. Defaults to NULL. Vector of rows to plot (interacting genes pair), one per line.
selected.columns	Strings vector. Defaults to NULL. Vector of columns (interacting groups) to plot, one per line
threshold.type	Character vector. Defaults to NULL. Possible arguments: "pvalue", "log2means", "pvalue_topn", "log2means_topn". Which thresholding system the user wants to use.
threshold.value	Numerical. Defaults to NULL. Value below/above (depending on whether it's pvalue or log2means) which genes to plot are selected.
project.name	Character vector. Defaults to NULL. Subfolder where to find the output of the CellPhoneDB analysis and where to save the dot_plot output plot.
filename	Character vector. Defaults to "selection_plot.pdf". Name of the file where the dot_plot output plot will be saved.
width	Numerical. Defaults to 8. Width of the plot.
height	Numerical. Defaults to 10. Height of the plot.

text.size	Numerical. Defaults to 12. Font size of the plot axes
return.vector	Logical. Defaults to FALSE. If set to TRUE, it includes the vector of genes_pairs present in the dot_plot in the VDJ_GEX_matrix[[10]] list.
platypus.version	This function works with "v3" only, there is no need to set this parameter

Value

VDJ_GEX_matrix object with output dot plot added to VDJ_GEX_matrix[[10]] list.

Examples

```
## Not run:
vgm_cellphonedb<-dot_plot(vgm.input=vgm_cellphonedb,
selected.columns = c("group_1_3|group_2_6", "group_1_3|group_3_9", "group_2_6|group_1_3",
"group_2_6|group_3_9", "group_3_9|group_1_3", "group_3_9|group_2_6"),
threshold.type="pvalue_topn", threshold.value=50,
project.name = "test", height = 12, width=6, text.size=10, return.vector=TRUE)

## End(Not run)
```

Echidna_simulate_repertoire

Simulate immune repertoire and transcriptome data

Description

Simulate repertoire and transcriptome matrix, with igraph tree plot for each clone showing the evolution process. the node in the tree plot are colored with transcriptome state and isotype.

Usage

```
Echidna_simulate_repertoire(
  initial.size.of.repertoire,
  species,
  cell.type,
  cd4.proportion,
  duration.of.evolution,
  complete.duration,
  vdj.productive,
  vdj.model,
  vdj.insertion.mean,
  vdj.insertion.stdv,
  vdj.branch.prob,
  clonal.selection,
  cell.division.prob,
  sequence.selection.prob,
```

```
  special.v.gene,
  class.switch.prob,
  class.switch.selection.dependent,
  class.switch.independent,
  SHM.method,
  SHM.nuc.prob,
  SHM.isotype.dependent,
  SHM.phenotype.dependent,
  max.cell.number,
  max.clonotype.number,
  death.rate,
  igraph.on,
  transcriptome.on,
  transcriptome.switch.independent,
  transcriptome.switch.prob,
  transcriptome.switch.isotype.dependent,
  transcriptome.switch.SHM.dependent,
  transcriptome.switch.selection.dependent,
  transcriptome.states,
  transcriptome.noise,
  seq.name
)
```

Arguments

`initial.size.of.repertoire`
: The initial number of existing cells when the evolution starts. Default is 10.

`species`
: The species of the simulated repertoire, can be "mus" for mouse or "hum" for human. Default is "mus".

`cell.type`
: The cell type for the simulation. "B" or "T"

`cd4.proportion`
: A number between 0 and 1 specifying the proportion of Cd4+ T cells, when cell.type is "T" and transcirptime states data is default. Default is 1, all the cells are Cd4. When user specify transciptome data for T cells, mixture of CD4+ and CD8+ T cells are not applicable.

`duration.of.evolution`
: The maxim time steps for simulation.

`complete.duration`
: TRUE or FALSE. Default is TURE. If TURE, after cell number or clone number reaches the upper limit, the evolution(class switch, mutation, transcriptional state switch) will continue until the duration.of.evolution is complete. If FLASE, the evolution will stop when either cell number or clone number reaches the limit.

`vdj.productive`
: "random": the sequence will be generated from random VDJ recombination, there might be a proportion of unproductive sequences.These VDJ genes were taken from IMGT. When more than one allele was present for a given gene, the first was used. "naive": the VDJ sequence be sampled from a pool of productive sequences obtained by filtering randomly simulated sequences with MIXCR.

"vae": the VDJ sequence be sampled from a pool of productive sequences obtained by filtering sequences generated from vae models with MIXCR.

`vdj.model` Specifies the model used to simulate V-D-J recombination. Can be either "naive" or "data". "naive" is chain independent and does not differentiate between different species. To rely on the default "experimental" options, this should be "data" and the parameter vdj.insertion.mean should be "default". This will allow for different mean additions for either the VD and JD junctions and will differ depending on species.

`vdj.insertion.mean`
Integer value describing the mean number of nucleotides to be inserted during simulated V-D-J recombination events. If "default" is entered, the mean will be normally distributed.

`vdj.insertion.stdv`
Integer value describing the standard deviation corresponding to insertions of V-D-J recombination. No "default" parameter currently supported but will be updated with future experimental data. This should be a number if using a custom distribution for V-D-J recombination events, but can be "default" if using the "naive" vdj.model or the "data", with vdj.insertion.mean set to "default".

`vdj.branch.prob`
Probability of new VDJ recombination event in each time step.when new VDJ recombination happen, a new cell with a new sequence will be generated. Default is 0.2.

`clonal.selection`
TRUE or FALSE. If TURE, cells in clones with higher frequency have their division probability proportional to the clonal frequency. If FALSE, clones with higher frequency will have lower probability to expand.

`cell.division.prob`
Probability of cells to be duplicated in each time step. Default is 0.1. If uneven probability for different clones is needed, the input should be a vector of 2 numeric items, with the first item being the lower bound, the second item being the upper bound of the division rate. The most abundant clone will get the highest division rate, and division rate of other clones will follow arithmetic progression and keep decreasing until the last abundant clone with the lower limit of division rate. If input 3 values, the third value will be the division rate for cells with selected sequences. If a fourth number is given, the division probability of selected sequence will be sampled between the third number and the fourth number.

`sequence.selection.prob`
Probability of each unique sequence to be selected as expanding sequence.Expanding sequences can have their division rate specified in the third element of cell.division.prob.

`special.v.gene` If TRUE, simulation will apply special sequence.selection.prob for heavy and light chain v gene combination specified in dataframe "special_v".

`class.switch.prob`
Probability matrix of class switching for b cells. The row names of the matrix are the isotypes the cell is switching from, the column names are the isotypes the cell is switching to. All B cells start from IGHM, and switch to one of the other isotypes or remain the same. Default values are in the attaching matrix

"class_switch_prob_hum" and "class_switch_prob_mus". The order of isotype in rows and columns should be the same.

`class.switch.selection.dependent`
: If TRUE,class switching will happen when the cell is selected,if the cell has IgM or IgD isotype.

`class.switch.independent`
: If TRUE, class switching will happen randomly at each time step for all cells. If FALSE, random class switching will be switched off.

`SHM.method`
: The mode of SHM speciation events. Options are either: "poisson","data","motif","wrc", and "all". Specifying either "poisson" or "naive" will result in mutations that can occur anywhere in the heavy chain region, with each nucleotide having an equal probability for a mutation event. Specifying "data" focuses mutation events during SHM in the CDR regions (based on IMGT), and there will be an increased probability for transitions (and decreased probability for transversions). Specifying "motif" will cause neighbor dependent mutations based on a mutational matrix from high throughput sequencing data sets (Yaari et al., Frontiers in Immunology, 2013). "wrc" allows for only the WRC mutational hotspots to be included (where W equals A or T and R equals A or G). Specifying "all" will use all four types of mutations during SHM branching events, where the weights for each can be specified in the "SHM.nuc.prob" parameter.

`SHM.nuc.prob`
: Specifies the rate at which nucleotides change during speciation (SHM) events. This parameter depends on the type of mutation specified by SHM.method. For both "poisson" and "data", the input value determines the probability for each site to mutate (the whole sequence for "poisson" and the CDRs for "data"). For either "motif" or "wrc", the number of mutations per speciation event should be specified. Note that these are not probabilities, but the number of mutations that can occur (if the mutation is present in the sequence). If "all" is specified, the input should be a vector where the first element controls the poisson style mutations, second controls the "data", third controls the "motif" and fourth controls the "wrc".

`SHM.isotype.dependent`
: If TRUE, somatic hypermutation of certain isotype will happen based on probability specified in dataframe "iso_SHM_prob".

`SHM.phenotype.dependent`
: If TRUE, somatic hypermutation of certain phenotype will happen based on probability specified in dataframe "pheno_SHM_prob".

`max.cell.number`
: Integer value describing maximum number of cells allowed. Default is 1500.

`max.clonotype.number`
: Integer value describing maximum number of clones allowed. cell derived from the same mother cell belong to same clone.

`death.rate`
: Probability of cell death happen to each cell in each time step.

`igraph.on`
: If TRUE, mutational network for every B cell clone will be in the output. If False, the igraphs will not be included.

`transcriptome.on`
: If TRUE, the simulation will include transcriptome data. If FALSE, only vdj sequence will be simulated.

`transcriptome.switch.independent`
: TRUE or FALSE value describing whether transcriptome state is allowed to switch independently, not dependent on class switching or somatic hypermutation. If TURE, transcriptome.switch.prob should be specified to control the probability of transcriptome state switching.

`transcriptome.switch.prob`
: Probability of transcriptome state switching independently. Default values are in the attaching matrix "trans_switch_prob_b" and "trans_switch_prob_t". The order of cell type in rows and columns should be the same, and the order of the cell type in the matrix should match cell type names in transcriptome.states.

`transcriptome.switch.isotype.dependent`
: TRUE or FALSE value describing whether transcriptome state of a cell is allowed to switch depending on isotype switching. If TRUE, transcriptome state will switch once class switching happens.

`transcriptome.switch.SHM.dependent`
: TRUE or FALSE value describing whether transcriptome state of a cell is allowed to switch depending on somatic hypermutation. If TRUE, transcriptome state will switch once somatic hypermutation happens.

`transcriptome.switch.selection.dependent`
: If TRUE, selected cells will undergo transcriptome state switching if their transcriptome state is 1.

`transcriptome.states`
: A data.frame specifying base gene expression for different cell type, with gene names as row names, cell type names as column names. When missing, a default data.frame will be used. Default data.frame includes "GerminalcenterBcell", "NaiveBcell", "Plasmacell", "MemoryBcell" for B cells, and "Naïve Cd4", "ActivatedCd4", "MemoryCd4", "NaiveCd8", "EffectorCd8" ,"MemoryCd8","ExhaustedCd8" for T cells. The order of the cell type names in transcriptome.states should match cell type names in the transcriptome.switch.prob matrix.

`transcriptome.noise`
: A character expression specifying the distribution of noise ratio to be multiplied with the base gene expression for each cell. It should be a text expression that generates a numeric vector, which is of the same length as gene names in the trasncriptome.state input. Default value is "rnorm(nrow(transcriptome.states), mean = 1, sd = 0.3)".

`seq.name`
: Integer specifies how many top-ranking clones are included in Seq_Name dataframe in the output list for phylogenetic tree plotting in other pipeline. If missing, Seq_Name won't be included in the output.

Value

A list containing the VDJ sequence and corresponding transcriptome data: "all_contig_annotations", "clonotypes", "all_contig", "consensus","reference","reference_real", "transcriptome","igraph_list_iso","igraph_list_trans","

Echidna_vae_generate *Simulate B or T cell receptor sequences by variational autoencodes(VAEs) trained with experimental data.*

Description

Simulate B or T cell receptor sequences by variational autoencodes(VAEs) trained with experimental data.

Usage

```
Echidna_vae_generate(
  sequence,
  n.train,
  n.sample,
  batch.size,
  latent.dim,
  intermediate.dim,
  epochs,
  epsilon.std,
  null.threshold
)
```

Arguments

sequence	a vector of seuqnece the model to be trained on
n.train	number of sequence to be used in training set, the rest will be in testing set
n.sample	number of new sequence to generate from VAE model
batch.size	set to larger to save time, set to smaller to same computing power
latent.dim	parameter used in VAE model
intermediate.dim	parameter used in VAE model
epochs	parameter used in VAE model
epsilon.std	parameter used in VAE model
null.threshold	threshold of predicted value to be considered as an existing base, default is 0.05. When generated sequence is too short, lower this threshold.

Value

A simulated VDJ repertoire on the basis of the input experimental repertoire

`get.avr.mut.data` *Get information about somatic hypermutation in the simulation. This function return a barplot showing the average mutation.*

Description

Get information about somatic hypermutation in the simulation. This function return a barplot showing the average mutation.

Usage

```
get.avr.mut.data(igraph.index.attr, history, clonotype.select, level)
```

Arguments

`igraph.index.attr`	A list "igraph.index.attr" from the simulation output.
`history`	A dataframe "history" from the simulation output.
`clonotype.select`	The selected clonotype index, can be the output of the function "select.top.clone.clone".
`level`	Can be "clone" or "cell". If "clone", the function will return average mutation on unique variant level. Otherwise it will return on cell level.

Value

a bar plot showing the average mutation on clone or cell level.

`get.avr.mut.plot` *Get information about somatic hypermutation in the simulation. This function return a barplot showing the average mutation.*

Description

Get information about somatic hypermutation in the simulation. This function return a barplot showing the average mutation.

Usage

```
get.avr.mut.plot(igraph.index.attr, history, clonotype.select, level, y.limit)
```

Arguments

igraph.index.attr	A list "igraph.index.attr" from the simulation output.
history	A dataframe "history" from the simulation output.
clonotype.select	The selected clonotype index, can be the output of the function "select.top.clone".
level	Can be "node" or "cell". If "node", the function will return average mutation on unique variant level. Otherwise it will return on cell level.
y.limit	The upper limit for y axis in the plot.

Value

a barplot showing the average mutation per node (same heavy and light chain set) or per cell.

---

get.barplot.errorbar *Return a barplot of mean and standard error bar of certain value of each clone.*

Description

Return a barplot of mean and standard error bar of certain value of each clone.

Usage

```
get.barplot.errorbar(data, y.lab, y.limit)
```

Arguments

data	A dataframe. Columns are different simulations, rows are the top clones. The first row is the top abundant clone.
y.lab	A string specifies the y lable name of the barplot.
y.limit	The upper limit for y axis in the plot.

Value

a barplot of mean and standard error bar of certain value of each clone.

get.elbow — *Get the seurat object from simulated transciptome output.*

Description

Get the seurat object from simulated transciptome output.

Usage

```
get.elbow(data)
```

Arguments

data	The output "transcriptome" dataframe from simulation output.

Value

the seurat object from simulated transciptome output.

get.n.node.data — *Get the number of unique variants in each clone in a vector. The output is the vector representing the numbers of unique variants.*

Description

Get the number of unique variants in each clone in a vector. The output is the vector representing the numbers of unique variants.

Usage

```
get.n.node.data(data, clonotype.select)
```

Arguments

data	The output "igraph.index.attr" list from simulation output.
clonotype.select	The index of the clones to be shown. If missing, all clones will be included.

Value

the number of unique variants in each clone in a vector. The output is the vector representing the numbers of unique variants.

`get.n.node.plot` *Get the number of unique variants in each clone in a vector and the barplot. The first item in the output is the vector representing the numbers of unique variants, the second item is the barplot.*

Description

Get the number of unique variants in each clone in a vector and the barplot. The first item in the output is the vector representing the numbers of unique variants, the second item is the barplot.

Usage

```
get.n.node.plot(igraph.index.attr, clonotype.select, y.limit)
```

Arguments

`igraph.index.attr`	The output "igraph.index.attr" list from simulation output.
`clonotype.select`	The index of the clones to be shown. If missing, all clones will be included.
`y.limit`	The upper limit for y axis in the plot.

Value

the number of unique variants in each clone in a vector and the barplot. The first item in the output is the vector representing the numbers of unique variants, the second item is the barplot.

`get.seq.distance` *Computing sequence distance according to the number of unmatched bases.*

Description

Computing sequence distance according to the number of unmatched bases.

Usage

```
get.seq.distance(germline, sequence)
```

Arguments

`germline`	A string representing the germline sequence.
`sequence`	A string of the sequence to be compared, which has the same length as germline.

Value

the number of unmatched bases in 2 sequences.

`get.umap` *Further process the seurat object from simulated transciptome output and make UMAP ready for plotting.*

Description

Further process the seurat object from simulated transciptome output and make UMAP ready for plotting.

Usage

```
get.umap(gex, d, reso)
```

Arguments

`gex`	output from get.elbow function.
`d`	dims argurment of in Seurat::FindNeighbors() and Seurat::RunUMAP
`reso`	resolution argument in Seurat::FindClusters()

Value

Further processed seurat object from simulated transciptome output with UMAP ready for plotting.

`get.vgu.matrix` *Get paired v gene heavy chain and light chain matrix on clonotype level. A v gene usage pheatmap can be obtain by p<-pheatmap::pheatmap(vgu_matrix,show_colnames= T, main = "V Gene Usage"), where the vgu_matrix is the output of this function.*

Description

Get paired v gene heavy chain and light chain matrix on clonotype level. A v gene usage pheatmap can be obtain by p<-pheatmap::pheatmap(vgu_matrix,show_colnames= T, main = "V Gene Usage"), where the vgu_matrix is the output of this function.

Usage

```
get.vgu.matrix(all.contig.annotations, level)
```

Arguments

`all.contig.annotations`	The dataframe "all_contig_annotation" from simulation output.
`level`	Can be "clone" or "cell". If "clone", the function will return paired v gene usage matrix on clonotype level. Otherwise it will return on cell level.

Value

a paired v gene heavy chain and light chain matrix on clonotype level.

`GEX_clonotype` *Platypus V2 GEX and VDJ integration for clonotypes*

Description

Platypus V2: 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.

Usage

```
GEX_clonotype(GEX.object, VDJ.per.clone)
```

Arguments

`GEX.object`	A single seurat object from automate_GEX function. This will likely be supplied as automate_GEX.output[[1]].
`VDJ.per.clone`	Output from the VDJ_per_clone function. Each element in the list should be found in the output from the automate_GEX function.

Value

Returns a dataframe containing repertoire information, such as isotype, CDR sequences, mean number of UMIs. This output can be supplied to furhter packages VDJ_extract_sequences and VDJ_GEX_integrate

Examples

```
## Not run:
GEX_clonotype(GEX.object=automate.GEX.output[[1]], VDJ.per.clone=vdj.per.clone.output)

## End(Not run)
```

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

```
#Platypus version v2
#GEX_cluster_genes(GEX =automate_GEX_output[[i]], min.pct = .25
#, filter = c("MT-", "RPL", "RPS"))

#Platypus version v3
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

```
## Not run:
#For Platypus version 2
cluster_defining_gene_heatmap <- GEX_cluster_genes_heatmap(GEX = automate_GEX_output[[i]]
,GEX_cluster_genes.output=GEX_cluster_genes_output
,n.genes.per.cluster=5,metric="p.value",max.cell=5)

#For Platypus version 3

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

## End(Not run)
```

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

```
#Platypus v2
#GEX_cluster_membership(GEX=automate_GEX_out[[2]], platypus.version = "v2")
#Platypus v3
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 == F or a ggplot if plot.dotmap == T detailing the coexpression levels of selected genes within the given cell population

Examples

```
#To return a dataframe with coefficients
#GEX_coexpression_coefficient(GEX = VDJ_GEX_matrix.output[[2]]
#, genes = c("CD19", "EBF1","SDC1"), subsample.n = "none", plot.dotmap = FALSE)

#To return a dotplot detailing coexpression and overall expression
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

```
#Basic run between two samples
DEGs <- GEX_DEgenes(GEX = Platypus::small_vgm[[2]],min.pct = .25,
group1 = "s1",group2 = "s2", return.plot = "volcano")
#DEGs[[1]] => Table of DEGs
#DEGs[[2]] => Volcano plot

#Getting DEGs between two seurat clusters
#GEX_DEgenes(GEX = Platypus::small_vgm[[2]],min.pct = .25,
#grouping.column = "seurat_clusters",group1 = "0",group2 = "1")

#Plotting a heatmap by foldchange of sample markers
#GEX_DEgenes(GEX = VDJ_GEX_matrix.output[[2]]
#,min.pct = .25,group1 = "s1",group2 = "s2", return.plot = "heatmap"
#, up.genes = 10, down.genes = 10, logFC = TRUE)

#Plotting volcano by p value of sample markers. Label additional genes of interest
#GEX_DEgenes(GEX = VDJ_GEX_matrix.output[[2]],min.pct = .25
#,group1 = "s1",group2 = "s2", return.plot = "volcano", logFC = FALSE
#, label.n.top.genes = 40, genes.to.label = c("CD28", "ICOS"))

#Generate a heatmap from an already existing FindMarkers output
#GEX_DEgenes(GEX = VDJ_GEX_matrix.output[[2]]
#, FindMarkers.out = FindMarkers.output.dataframe, return.plot = "heatmap"
#, up.genes = 10, down.genes = 10, logFC = TRUE, platypus.version = "v3")
```

GEX_DEgenes_persample *Platypus V2 Differentially expressed genes*

Description

!Only for Platypus version v2. For more flexibility and platypus v3 please refer to 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 and further extracting sample information from the "sample_id" component of the Seurat object.

Usage

```
GEX_DEgenes_persample(
  automate.GEX,
  min.pct,
  sample1,
```

```
  sample2,
  by.group,
  filter,
  return.plot,
  logFC,
  up.genes,
  down.genes,
  base
)
```

Arguments

automate.GEX	Output Seurat object from automate_GEX function that contained at least two distinct biological samples. The differential biological samples correspond to integer values in the order of the working directories initially supplied to the automate_GEX function.
min.pct	The minimum percentage of cells expressing a gene in either of the two groups to be compared.
sample1	either character or integer specifying the first sample that should be compared.
sample2	either character or integer specifying the first sample that should be compared.
by.group	Logical specifying if groups should be used instead of samples. If TRUE, then the argument in sample1 and sample2 will correspond to cells found in the groups from sample1 or sample2.
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	Logical specifying if a heatmap of the DEX genes is to be returned. If TRUE then @return is a list where the first element is a dataframe and the second a heatmap (see @return)
logFC	Logical specifying whether the genes will be displayed based on logFC (TRUE) or pvalue (FALSE).
up.genes	Integer specifying the number of upregulated genes to be shown.
down.genes	Integer specifying the number of downregulated genes to be shown.
base	The base with respect to which logarithms are computed. Default: 2

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

```
## Not run:
GEX_DEgenes_persample(automate.GEX=automate.GEX.output[[i]]
,min.pct = .25,sample1 = "1",sample2 = "2")

## End(Not run)
```

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

```
#To return a plot detailing the expression of common genes by seurat cluster
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

```
## Not run:
# Pre-defined gene set for CD4 T cells
GEX_gene_visualization(GEX = VGM$GEX, predefined_genes = "CD4_T_cell")

# Pre-defined gene set for CD8 T cells
GEX_gene_visualization(GEX = VGM$GEX, predefined_genes = "CD8_T_cell")

# Pre-defined gene set for B cells
GEX_gene_visualization(GEX = VGM$GEX, predefined_genes = "B_cell")

# Gene set defined by user
GEX_gene_visualization(GEX = VGM$GEX, gene_set=c("CD8A","CD3E","SELL","FAS","ID3","SDC1"))

## End(Not run)
```

GEX_GOterm *GEX GO-Term analysis and plotting*

Description

Runs a GO term analysis on a submitted list of genes. Works with the output of GEX_topN_DE_genes_per_cluster or a custom list of genes to obtain GOterms.

Usage

```
GEX_GOterm(
  GEX.cluster.genes.output,
  topNgenes,
  ontology,
  species,
  up.or.down,
  MT.Rb.filter,
  kegg,
  go.plots,
  top.N.go.terms.plots,
  kegg.plots,
  top.N.kegg.terms.plots
)
```

Arguments

GEX.cluster.genes.output	Either output of Platypus::GEX_cluster_genes or custom character vector containing gene symbols. A custom gene list will not be further filtered or ordered.
topNgenes	How many of the most significant up or down regulated genes should be considered for GO term analysis. All genes will be used if left empty.
ontology	Ontology used for the GO terms. "MF", "BP" or "CC" possible. Default: "BP"
species	The species the genes belong to. Default: "Mm" (requires the package "org.Mm.eg.db"). Set to "Hs" for Human (requires the package "org.Hs.eg.db")
up.or.down	Whether up or downregulated genes should be used for GO term analysis if GEX_cluster_genes output is used. Default: "up"
MT.Rb.filter	logical, if mitochondrial and ribosomal genes should be filtered out.
kegg	logical, if KEGG pathway analysis should be conducted. Requires internet connection. Default: False.
go.plots	logical, if top GO-terms should be visualized. Default: False. If True, for each cluster the top N (top.N.GO.terms.plots) Go-terms for each cluster will be plotted to the working directory and saved as a list element. Plots are made both based on padj and ratio.

top.N.go.terms.plots
: The number of most significant GO-terms to be incluted in the go.plots. Default: 10.

kegg.plots
: logical, if top KEGG-terms should be visualized. Default: False. If True, for each cluster the top N (top.N.kegg.terms.plots) KEGG-terms for each cluster will be plotted to the working directory and saved as a list element. Plots are made both based on padj and ratio.

top.N.kegg.terms.plots
: The number of most significant KEGG-terms to be incluted in the kegg.plots. Default: 10.

Value

Returns a list of data frames and plots containing the TopGO and the TopKEGG output containing the significant GO/KEGG terms and their visualizations.

Examples

```
## Not run:

GEX_GOterm(DE_genes_cluster,MT.Rb.filter = TRUE, ontology = "MF")
GEX_GOterm(rownames(DE_genes_cluster[[1]]),MT.Rb.filter = TRUE, species= "Mm",
ontology = "BP", go.plots = TRUE)

#Install the needed database with
#if (!requireNamespace("BiocManager", quietly = TRUE))
#install.packages("BiocManager")
#BiocManager::install("org.Mm.eg.db")
#BiocManager::install("org.Hs.eg.db")

## End(Not run)
```

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,
  verbose
)
```

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 == T. 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.
verbose	Print run parameters and status to console

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

```
## Not run:
df <- GEX_cluster_genes(gex_combined[[1]])

#Using gmt file to perform gsea
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)
View(gex_gsea[[1]])

#Directly downloading gene set collection from MSigDB to perform gsea
output <- GEX_GSEA(GEX.cluster.genes.output =  df[[1]], MT.Rb.filter = TRUE
, path.to.pathways = c("Mus musculus", "C7"))

#Using your own gene list to perform gsea
output <- GEX_GSEA(GEX.cluster.genes.output =  df[[1]], MT.Rb.filter = TRUE
, my.own.geneset = my_geneset)

## End(Not run)
```

`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

```
#prep the small_vgm sample dataset
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

```
## Not run:
lineage_trajectories <- GEX_lineage_trajectories( VGM$GEX,
 grouping = 'group_id',
  cluster.num = "3")

## End(Not run)
```

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
)
```

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 indipendently of their p value. Can be used in combination with label.n.genes.
save.plot	Boolean. Defaults to True. Whether to save plots as appropriately named .png 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)
```

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

```
#For testing: only a single clonotype in two samples
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_projecTILS *ProjectTILs tool utility*

Description

Projection of scRNA-seq data into reference single-cell atlas, enabling their celltype annotation based on the single-cell atlas.

Usage

```
GEX_projecTILS(
  ref_path,
  GEX,
  split_by,
  filtering = c(TRUE, FALSE),
  NA_cells = c(TRUE, FALSE)
)
```

Arguments

ref_path	Path to reference TIL atlas file (ex: c:/Users/.../ref_TILAtlas_mouse_v1.rds). The atlas can be downloaded from the GitHub of ProjecTILs.
GEX	GEX output of the VDJ_GEX_matrix function (VDJ_GEX_matrix[[2]])).
split_by	Optional character vector to specify how the GEX should be split for analysis. This parameter can refer to any column in the GEX. If none is given by the user the analysis will take the whole GEX.
filtering	Logical, if TRUE a filtering is apply which eliminates unwanted cells. By default it is set to FALSE.
NA_cells	Logical, if TRUE the cells not assigned by projecTILs are kept in the bar plot, if FALSE, not assigned cells are filtered out. By default it is set to TRUE.

Value

Return a list. Element[[1]] is the GEX data frame containing two new columns containing ProjecTILs cell type assignment. Element[[2]] is the output of make.projection function from projecTILs based on the given GEX. Element[[3]] contains a UMAP plot per each groups based on projecTILs assignment. Element[[4]] plots of the fraction of cells with predicted state per cluster.

Examples

```
## Not run:
#Without splitting argument, considering the whole VGM
output_projecTILS_whole_VGM<-GEX_projecTILS(
ref_path = "c:/Users/.../ref_TILAtlas_mouse_v1.rds", GEX = VGM$GEX,
filtering =TRUE)
output_projecTILS_whole_VGM[[3]] #Umap
output_projecTILS_whole_VGM[[4]] #Barplots

#With splitting argument by groups_id
output_projecTILS_split_by_group<-GEX_projecTILS(
ref_path = "c:/Users/.../ref_TILAtlas_mouse_v1.rds", GEX = VGM$GEX,
filtering =  TRUE, split_by = "group_id", NA_cells = FALSE)
output_projecTILS_split_by_group[[3]] #Umap
output_projecTILS_split_by_group[[4]] #Barplots

## End(Not run)
```

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, verbose)
```

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
verbose	Print information about factor levels and ordering to console

Value

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

Examples

```
#To return a normal barplot which shows the % of cells of
#each sample contained in each cluster
GEX_proportions_barplot(GEX = Platypus::small_vgm[[2]], source.group = "sample_id"
, target.group = "seurat_clusters",stacked.plot = FALSE)

#To return a stacked barplot which shows the % of cells of each
#cluster attributed to each sample
GEX_proportions_barplot(GEX = Platypus::small_vgm[[2]],
source.group = "seurat_clusters", target.group = "sample_id"
,stacked.plot = TRUE)
```

GEX_pseudobulk *Function that performs pseudo-bulking on the data (VGM input), according to criteria specified by the User, and uses the pseudo-bulked data to perform Differential Gene Expression (DGE) analysis.*

Description

Function that performs pseudo-bulking on the data (VGM input), according to criteria specified by the User, and uses the pseudo-bulked data to perform Differential Gene Expression (DGE) analysis.

Usage

```
GEX_pseudobulk(
  vgm.input,
  column.group,
  group1,
  group2,
  column.comparison,
  comparison,
  pool,
  platypus.version
)
```

Arguments

vgm.input	Output of the VDJ_GEX_matrix function. Mandatory
column.group	Character vector. Mandatory. Column name of VDJ_GEX_matrix[[2]] where the groups to be tested for differenetial gene expression are located
group1	Strings vector. Mandatory. Samples to be grouped together for differential expression analysis against group2 (if pool=TRUE or column.comparison!=NULL). If pool=FALSE, vector containing samples to be tested individually against samples with the same index in the vector of group2.
group2	Strings vector. Mandatory. Samples to be grouped together for differential expression analysis against group1 (if pool=TRUE or column.comparison!=NULL). If pool=FALSE, vector containing samples to be tested individually against samples with the same index in the vector of group1.
column.comparison	Character vector. Defaults to NULL. Column name of VDJ_GEX_matrix[[2]] where the comparison cathegories are located, if DGE between group1 and group2 is performed across diferrent cathegories.
comparison,	Strings vector. Defaults to NULL. Comparison cathegories, if more than one is present.
pool	Logical. Defaults to FALSE. Indicates whether samples specified in group1 and group2 are to be pooled together within the same group.
platypus.version	This function works with "v3" only, there is no need to set this parameter

Value

A data.frame or list of data.frames containing the results of the DGE analysis for every level of pseudo-bulking.

Examples

```
## Not run:
pseudo_DE<-GEX_pseudobulk(
vgm.input=VGM_NP396_GP33_GP66_labelled,
column.group="sample_id",
```

```
group1 = c("s1"), group2 = c("s4"),
column.comparison = "seurat_clusters",
comparison=c(2,3,5,9), pool=FALSE)

## End(Not run)
```

GEX_pseudotime_trajectory_plot

This function plots pseudotime along the trajectories which have been constructed with the GEX_trajectories() function.

Description

This function plots pseudotime along the trajectories which have been constructed with the GEX_trajectories() function.

Usage

```
GEX_pseudotime_trajectory_plot(cds, root.nodes, monocle.version, root.state)
```

Arguments

cds	cell data set object. Output element [[1]] of the GEX_trajectories() function
root.nodes	For monocle3: Root nodes to determine for the pseudotime trajectories. GEX_trajectories output [[3]] yields all the possible root nodes. Choose the ones you like.
monocle.version	Version of monocle. Either monocle2 or monocle3. Has to be the same as in GEX_trajectories().Default is monocle3.
root.state	For monocle2: Root state to determine starting cluster for the pseudotime trajectories. GEX_trajectories ouput [[3]] yields all the possible root states Choose the one you like.

Value

Returns a list.Element [[1]] cell data set object with the pseudotime trajectories. Element [[2]] pseudotime trajectory plot

Examples

```
## Not run:
##monocle3
pseudotime_output <- GEX_pseudotime_trajectory_plot(
GEX_trajectories_output[[1]], root.nodes = c('Y_742','Y_448','Y_964') )

##monocle2
pseudotime_output <- GEX_pseudotime_trajectory_plot(
GEX_trajectories_output[[1]], monocle.version = 'monocle3', root.state = "2")
```

```
## End(Not run)
```

`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_topN_DE_genes_per_cluster`
Platypus V2 GEX DE genes helper

Description

Organizes the top N genes that define each Seurat cluster and converts them into a single dataframe. This can be useful for obtaining insight into cluster-specific phenotypes.

Usage

```
GEX_topN_DE_genes_per_cluster(GEX_cluster_genes.output, n.genes, by_FC, filter)
```

Arguments

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 differentially regulated for each cluster.
n.genes	The number of genes to be selected from each cluster. If n.genes is higher than the number of cells in a cluster then it is silently adjusted to be
by_FC	Logical indicating if the top n genes are selected based on the logFC value instead of p value. Default is FALSE.
filter	Character vector of initials of the genes to be filtered. Default is c("MT-", "RPL", "RPS"), which filters mitochondrial and ribosomal genes.

Value

Returns a dataframe in which the top N genes defining each cluster based on differential expression are selected.

Examples

```
## Not run:
GEX_topDE_genes_per_cluster(GEX_cluster_genes.output=list_of_genes_per_cluster
, n.genes=20, by_FC=FALSE, filter=c("MT-", "RPS", "RPL"))

## End(Not run)
```

GEX_trajectories *This is a function which infers trajectories along ordered cells on dimensionality reduced data. It projects trajectrories on a dim. red. plot such as Umap. This uses Monocle3 or Monocle2.*

Description

This is a function which infers trajectories along ordered cells on dimensionality reduced data. It projects trajectrories on a dim. red. plot such as Umap. This uses Monocle3 or Monocle2.

Usage

```
GEX_trajectories(
  GEX,
  color.cells.by,
  reduction.method,
  cluster.method,
  genes,
  label.cell.groups,
  label.groups.by.cluster,
  labels.per.group,
```

```
  group.label.size,
  monocle.version,
  ordering.cells.method
)
```

Arguments

GEX	GEX output of the VDJ_GEX_matrix function (VDJ_GEX_matrix[[2]]))
color.cells.by	Column name in SummarizedExperiment::colData(GEX). To decide how the cells are colored in the output plot. E.g. color.cells.by = 'group_id' the cells will be colored based on their group_id.
reduction.method	Which method to use for dimensionality reduction for monocle3. Supports "UMAP", "tSNE", "PCA" or "LSI". Default value is "UMAP".
cluster.method	Monocle3 gives two clustering options: Using the Leiden or the Louvain algo. Default is louvain.
genes	Takes a vector of genes (e.g. genes = c('CD3E', 'CD4', 'CD8A', 'CD44')) to highlight the expression of these genes in UMAP and in the trajectory plot in monocle3. Default is NULL.
label.cell.groups	Whether to label cells in each group according to the most frequently occurring label(s) (as specified by color_cells_by) in the group. If false, plot_cells() simply adds a traditional color legend. Default is TRUE
label.groups.by.cluster	Instead of labeling each cluster of cells, place each label once, at the centroid of all cells carrying that label. Default is TRUE
labels.per.group	How many labels to plot for each group of cells. Default is 1
group.label.size	Font size to be used for cell group labels. Default is 1
monocle.version	Version of monocle. Either monocle2 or monocle3. Default is monocle3.
ordering.cells.method	In monocle2 you can choose between selecting genes with high dispersion across cells for ordering cells along a trajectory (= 'high.dispersion'). Or order cells based on genes which differ between clusters, uses an unsupervised procedure called "dpFeature" (= 'differ.genes'). Defalut is "differ.genes"

Value

Returns a list.For monocle3: Element [[1]] returns a cell data set object with a new column for the UMAP clustering. This will be used for the GEX_pseudotime_trajectory_plot() function. [[2]] contains a plot of the clusters. [[3]] contains also a cluster plot but with the inferred trajectories. For monocle2: [[1]] cell data set object. [[2]] Trajetory plot with cells coloured based on their states (important to choose root state for pseudotime plot). [[3]] Trajectory plot based on color.cells.by

Examples

```
## Not run:

trajectory_output <- GEX_trajectories(GEX = vgm[[2]],
 reduction.method = "UMAP",
 color.cells.by = "group_id",
 labels_per_group = 2,
 group_label_size = 3)

 #visualizing gene expressions
 interesting_genes = c("Cxcr6", "Il7r")
 genes_trajectories <- GEX_trajectories(GEX = VGM$GEX,
  color.cells.by = "group_id",
   genes = interesting_genes)

##monocle2 ! DEPRECATED !
#trajectory_output <- GEX_trajectories(GEX = vgm[[2]],
#  monocle.version = "monocle2",
#  ordering.cells.method = "high.dispersion")

## End(Not run)
```

`GEX_visualize_clones` *Platypus V2 GEX and VDJ integration for visualizing clone clustering*

Description

!Only for platypus version v2. For platypus v3 refer to: VDJ_GEX_overlay_clones() Visualize selected clonotypes on the tSNE or UMAP projection.

Usage

```
GEX_visualize_clones(
  GEX.list,
  VDJ.GEX.integrate.list,
  highlight.type,
  highlight.number,
  reduction
)
```

Arguments

`GEX.list`	list of Seurat objects, output of the automate_GEX function.
`VDJ.GEX.integrate.list`	Output of the VDJ_GEX_integrate function.
`highlight.type`	(Optional) either "None" if representation highlighted by cluster, "clonotype" if want to highlight most expanded clonotypes, or "sample" if several samples are within the same Seurat object. Default is None.

highlight.number
: (Optional) an integer or list of integers representing the number of most expanded clonotypes or samples one wants to select eg 4 to highlight the 4th most expanded clonotype or 2:5 to highlight the top 2 to top 5 most expanded clonotype. Only compatible with highlight.type "clonotype" or "sample", will be ignored if type is "None". Default is 1.

reduction
: (Optional) Reduction to plot, either "tsne", "umap", or "harmony". Default is "tsne".

Value

concatenated ggplot2 plot with selected clonotypes highlighted (if None, the coloring is according to the clustering).

Examples

```
## Not run:
GEX_visualize_clones(GEX.list=automate_GEX.output,
 VDJ.per.clone=VDJ_per_clone.output,
 highlight.type="clonotype",
 highlight.number=1:4,
 reduction="umap")

## End(Not run)
```

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

```
## Not run:
#using the findmarkers.output
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)

#using the GEX_cluster_genes output
GEX_volcano(findmarkers.output = GEX_cluster_genes.Output
, cluster.genes.output =TRUE)

## End(Not run)
```

`hotspot_df` *hotspot_df Hotspot mutations taken from Yaari et al., Frontiers in Immunology, 2013. This contains transition probabilities for all 5mer combinations based on high throughput sequencing data. The transition probabilities are for the middle nucleotide in each 5mer set. This can be customized by changing the genes and sequences. Custom mutation hotspots can be supplied by modifying this dataframe. Repeating particular hotspot entries allows for the hotspot to mutate more than one time per SHM event.*

Description

@format A data frame with 1024 rows and 6 variables:

pattern Character array where each entry corresponds to a 5 base motif. The mutation probabilities correspond to the middle nucleotide in each 5mer.

toA The probability for the middle nucleotide in "pattern" to mutate to an adenine

toC The probability for the middle nucleotide in "pattern" to mutate to an cytosine

toG The probability for the middle nucleotide in "pattern" to mutate to an guanine

toT The probability for the middle nucleotide in "pattern" to mutate to an thymine

Source The origin of how this motif was discovered. Either Inferred or Experimental

Usage

```
data("hotspot_df")
```

Format

An object of class `data.frame` with 1024 rows and 6 columns.

Source

Yaari et al., Frontiers in Immunology, 2013

`hum_b_h` *hum_b_h*

Description

human germline IgH (heavy chain v,d,j gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
data("hum_b_h")
```

Format

A list including 3 elements (data frames): v gene, d gene, j gene,respectively.

[[1]]

gene The v gene name

seq The corresponding sequence [[2]]

gene The d gene name

seq The corresponding sequence [[3]]

gene The j gene name

seq The corresponding sequence

Source

IMGT

hum_b_l *hum_b_l*

Description

human germline IgH (light chain v,d,j gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
data("hum_b_l")
```

Format

A list including 2 elements (data frames): v gene, d gene, j gene,respectively.

[[1]]

gene The v gene name

seq The corresponding sequence [[2]]

gene The j gene name

seq The corresponding sequence

Source

IMGT

hum_t_h *hum_t_h*

Description

human germline TRB (heavy chain v,d,j gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
data("hum_t_h")
```

Format

A list including 3 elements (data frames): v gene, d gene, j gene,respectively.

[[1]]

gene The v gene name

seq The corresponding sequence [[2]]

gene The d gene name

seq The corresponding sequence [[3]]

gene The j gene name

seq The corresponding sequence

Source

IMGT

`hum_t_l` *hum_t_l*

Description

human germline TRA (light chain v,d,j gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
data("hum_t_l")
```

Format

A list including 2 elements (data frames): v gene, d gene, j gene,respectively.

[[1]]

gene The v gene name

seq The corresponding sequence [[2]]

gene The j gene name

seq The corresponding sequence

Source

IMGT

iso_SHM_prob *iso_SHM_prob A probability dataframe specifying SHM.nuc.prob for cells of different isotypes. The first column is the names of isotypes, while the second column is the SHM.nuc.prob of cell of that isotype. user can define different SHM.nuc.prob for isotypes.*

Description

iso_SHM_prob A probability dataframe specifying SHM.nuc.prob for cells of different isotypes. The first column is the names of isotypes, while the second column is the SHM.nuc.prob of cell of that isotype. user can define different SHM.nuc.prob for isotypes.

Usage

```
data("iso_SHM_prob")
```

Format

a dataframe with 2 columns

mus_b_h *mus_b_h*

Description

C57BL/6 germline IgH (heavy chain v,d,j gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
data("mus_b_h")
```

Format

A list including 3 elements (data frames): v gene, d gene, j gene,respectively.

[[1]]

gene The v gene name

seq The corresponding sequence [[2]]

gene The d gene name

seq The corresponding sequence [[3]]

gene The j gene name

seq The corresponding sequence

Source

IMGT

`mus_b_l` *mus_b_l*

Description

C57BL/6 germline IgH (light chain v,d,j gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
data("mus_b_l")
```

Format

A list including 2 elements (data frames): v gene, d gene, j gene,respectively.

[[1]]

gene The v gene name

seq The corresponding sequence [[2]]

gene The j gene name

seq The corresponding sequence

Source

IMGT

`mus_b_trans` *mus_b_trans A data frame contains mouse B cell average gene expression for multiple cell types, with the rows representing the gene names, column names representing the cell type names. The original single cell sequencing data is retrieved from 10xgenomics and combined with experimental data The expression level for different cell types are obtained by calculating the average expression after sorting the original data by markers as shown below.*

Description

mus_b_trans A data frame contains mouse B cell average gene expression for multiple cell types, with the rows representing the gene names, column names representing the cell type names. The original single cell sequencing data is retrieved from 10xgenomics and combined with experimental data The expression level for different cell types are obtained by calculating the average expression after sorting the original data by markers as shown below.

Usage

```
data("mus_b_trans")
```

Format

A data frame with 26538 rows and 4 variables, with the rows representing the gene names, column names representing the cell type names.

NaiveBcell Cd19+;Cd27-;Cd38-

GerminalcenterBcell Fas+;Cd19+

Plasmacell Sdc1+

MemoryBcell Cd38+;Fas-

Source

https://support.10xgenomics.com/single-cell-vdj/datasets/3.0.0/vdj_v1_mm_c57bl6_pbmc_5gex; https://support.10xgenomi cell-vdj/datasets/3.0.0/vdj_v1_mm_balbc_pbmc_5gex

---

mus_t_h *mus_t_h*

Description

C57BL/6 germline TRB (heavy chain v,d,j gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
data("mus_t_h")
```

Format

A list including 3 elements (data frames): v gene, d gene, j gene,respectively.

[[1]]

gene The v gene name

seq The corresponding sequence [[2]]

gene The d gene name

seq The corresponding sequence [[3]]

gene The j gene name

seq The corresponding sequence

Source

IMGT

`mus_t_l` *mus_t_l*

Description

C57BL/6 germline TRA (light chain v,d,j gene segments). When multiple alleles were present, the first one was included. These names and sequences can be changed by customized by changing this dataframe. Additionally, repeating elements can give certain germline gene elements a larger probability of being used during repertoire evolution.

Usage

```
data("mus_t_l")
```

Format

A list including 2 elements (data frames): v gene, d gene, j gene,respectively.

[[1]]

gene The v gene name

seq The corresponding sequence [[2]]

gene The j gene name

seq The corresponding sequence

Source

IMGT

`no.empty.node` *Get clone network igraphs without empty mode. Empty node represents the 'extincted' sequences, that are not in any living cell but once existed.*

Description

Get clone network igraphs without empty mode. Empty node represents the 'extincted' sequences, that are not in any living cell but once existed.

Usage

```
no.empty.node(history, igraph.index)
```

Arguments

history	The dataframe 'history' from the simulation output.
igraph.index	The list 'igraph.index' from the simulation output.
empty.node	If TRUE, there will be empty node in igraph. if FALSE, the empty node will be deleted.

Value

a list of clone network igraphs without empty mode.

---

one_spot_df *one_spot_df*

Description

WRC hotspot mutations taken from Yaari et al., Frontiers in Immunology, 2013. These include only the mutations following the WRC pattern, where W equals A or T and R equals A or G). Custom mutation hotspots can be supplied by modifying this dataframe. Repeating particular hotspot entries allows for the hotspot to mutate more than one time per SHM event.

Usage

```
data("one_spot_df")
```

Format

A data frame with 32 rows and 6 variables:

pattern Character array where each entry corresponds to a 5 base motif. The mutation probabilities correspond to the middle nucleotide in each 5mer.

toA The probability for the middle nucleotide in "pattern" to mutate to an adenine

toC The probability for the middle nucleotide in "pattern" to mutate to an cytosine

toG The probability for the middle nucleotide in "pattern" to mutate to an guanine

toT The probability for the middle nucleotide in "pattern" to mutate to an thymine

Source The origin of how this motif was discovered. Either Inferred or Experimental

Source

Yaari et al., Frontiers in Immunology, 2013

pheno_SHM_prob *pheno_SHM_prob A probability dataframe specifying SHM.nuc.prob for cells of different phenotypes. The first column is the names of phenotypes, while the second column is the SHM.nuc.prob of cell of that phenotype. user can define different SHM.nuc.prob for phenotypes.*

Description

pheno_SHM_prob A probability dataframe specifying SHM.nuc.prob for cells of different phenotypes. The first column is the names of phenotypes, while the second column is the SHM.nuc.prob of cell of that phenotype. user can define different SHM.nuc.prob for phenotypes.

Usage

```
data("pheno_SHM_prob")
```

Format

a dataframe with 2 columns

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

```
## Not run:

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)

## End(Not run)
```

PlatypusDB_fetch *Loads and saves RData objects from the PlatypusDB*

Description

Loads and saves RData objects from the PlatypusDB

Usage

```
PlatypusDB_fetch(
  PlatypusDB.links,
  save.to.disk,
  load.to.enviroment,
  load.to.list,
  path.to.save,
  combine.objects
)
```

Arguments

`PlatypusDB.links`
: Character vector. One or more links to files in the PlatypusDB. Links are constructed as follows: "%Project id%/%sample_id%/%filetype%". Any of the three can be "ALL", to download all files fitting the other link elements. If %filetype% is gexVGM, vdjVGM or metadata, %sample_id% needs to be "ALL", as these are elements which are not divided by sample. See examples for clarification. See last example on how to download AIRR compliant data. Feature Barcode (FB) data will be downloaded both for GEX and VDJ if present and does not need to be specified in the path. For sample_id entries the the metadata table for a given project via the function PlatypusDB_list_projects()

`save.to.disk`
: Boolean. Defaults to FALSE. Whether to save downloaded files individually to the directory specified in path.to.save

`load.to.enviroment`
: Boolean. Defaults to TRUE. Whether to load objects directly into the current .GlobalEnv. An array of the names of the loaded objects will be returned. !Be aware of RAM limitations of your machine when downloading multiple large files.

`load.to.list`
: Boolean. Defaults to FALSE. Whether to return loaded objects as a list. !Be aware of RAM limitations of your machine when downloading multiple large files.

`path.to.save`
: System path to save files to.

`combine.objects`
: Boolean. Defaults to TRUE. Whether to combine objects if appropriate. e.g. VDJ and GEX RData objects for a sample are saved as two independent objects and downloaded as such, to allow for flexibility. If combine.objects is set to TRUE, the function will coerce RData objects of each loaded sample or of each loaded VDJ_GEX_matrix appropriately. Combined input of VDJ and GEX Rdata objects can be directly supplied to the VDJ_GEX_matrix function.

Value

A list of loaded project files as R objects if load.to.list = T or a name of these object loaded to the enviroment if load.to.enviroment = T.

Examples

```
## Not run:

#Get a list of available projects by name
names(PlatypusDB_list_projects())

#Load the VDJ_GEX_matrix of a project as an object and
#also save it to disk for later.
#This will download the VDJ and GEX part of the VDJ_GEX_matrix and combine
PlatypusDB_fetch(PlatypusDB.links = c("Kuhn2021a//ALL")
,save.to.disk = FALSE,load.to.enviroment = TRUE, load.to.list = FALSE
, combine.object = TRUE,path.to.save = "/Downloads")

#Load VDJ dataframe of the VDJ GEX matrix for all samples of one project
loaded_list <- PlatypusDB_fetch(PlatypusDB.links = c("Kuhn2021a//VDJmatrix")
,save.to.disk = FALSE,load.to.enviroment = FALSE, load.to.list = TRUE)

#Load the VDJ and GEX RData of 2 samples from
#2 different projects which can be directly passed
#on to the VDJ_GEX_matrix function to integrate
#downloaded_objects <- PlatypusDB_fetch(
#PlatypusDB.links = c("Project1/s1/ALL", "Project1/s2/ALL")
#,save.to.disk = FALSE,load.to.enviroment = FALSE, load.to.list = TRUE
#, combine.objects = TRUE)

#integrated_samples <- VDJ_GEX_matrix_DB(data.in = downloaded_objects)

#Download metadata objects for projects
list_of_metadata_tables <- PlatypusDB_fetch(
PlatypusDB.links = c("Kuhn2021a//metadata")
,save.to.disk = FALSE,load.to.enviroment = FALSE, load.to.list = TRUE)

#Dowload of airr_rearrangement.tsv
#Load VDJ.RData into a list
#downloaded_objects <- PlatypusDB_fetch(
#PlatypusDB.links = c("Project1/ALL/VDJ.RData"),save.to.disk = FALSE
#,load.to.enviroment = FALSE, load.to.list = TRUE)

#Extract airr_rearrangement table for sample 1
#airr_rearrangement <- downloaded_objects[[1]][[1]][[6]]
#Index hierarchy: Sample, VDJ or GEX, VDJ element

#Save for import to AIRR compatible pipeline
#write.table(airr_rearrangement, file = "airr_rearrangement_s1.tsv", sep='\t',
#row.names = FALSE, quote=FALSE)


## End(Not run)
```

PlatypusDB_find_CDR3s *CDR3 query function for PlatypusDB*

Description

Queries for the occurrence of CDR3 sequences in public datasets on PlatypusDB.

Usage

```
PlatypusDB_find_CDR3s(VDJ.cdr3s.aa, VJ.cdr3s.aa, projects.to.search)
```

Arguments

VDJ.cdr3s.aa	Character A VDJ CDR3s amino acid sequence to search for
VJ.cdr3s.aa	Character A VJ CDR3s amino acid sequence to search for
projects.to.search	Optional character vector. Defaults to "ALL". Names of projects to search within.

Value

A list of subsets of VDJ matrices from projects containing the query VDJ CDR3 (out[[1]]), the VJ CDR3 (out[[2]]) and cells containing both the query VDJ and VJ CDR3s (out[[3]])

Examples

```
## Not run:

public_clones <- PlatypusDB_find_CDR3s(VDJ.cdr3s.aa = "CMRYGNYWYFDVW"
, VJ.cdr3s.aa = "CLQHGESPFTF", projects.to.search = "ALL")


## End(Not run)
```

PlatypusDB_list_projects *Metadata download by project for PlatypusDB*

Description

Lists metadata tables of available projects on PlatypusDB

Usage

```
PlatypusDB_list_projects(keyword)
```

Arguments

keyword	Character. Keyword by which to search project ids (First Author, Year) in the database. Defaults to an empty string ("") which will list all projects currently available

Value

A list of metadata tables by project. List element names correspond to project ids to use in the PlatypusDB_fetch function

Examples

```
## Not run:

#Get list of all available projects and metadata.
PlatypusDB_projects <- PlatypusDB_list_projects()

#Names of list are project ids to use in PlatypusDB_fetch function
names(PlatypusDB_projects)
#Common format: first author, date, letter a-z (all lowercase)

#View metadata of a specific project
print(PlatypusDB_projects[["Kuhn2021a"]])


## End(Not run)
```

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

```
## Not run:
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)

## End(Not run)
```

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

```
## Not run:
#complete workflow below

#usage with airr rearrangement tables from PlatypusDB_load_from_disk
#or PlatypusDB_fetch list object
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)
```

```
#usage with airr rearrangement tables from disk
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"))

#usage with airr rearrangement tables from objects in R environment
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))

#Complete workflow
#set paths of cellranger directories containing
#also the airr_rearrangements.tsv file
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")
#Run VGM with GEX and VDJ integration
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)
#Generate AIRR compatible table supplemented by GEX information
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"))

#To save a dataframe as .tsv
write.table(airr_dataframe, file = "supplemented_airr_rearrangements.tsv"
, sep='\t', row.names = FALSE, quote=FALSE)

## End(Not run)
```

PlatypusML_balance *Secondary ML for crossvalidation*

Description

This PlatypusML_classification function takes as input encoded features obtained using the PlatypusML_extract_features function. The function runs cross validation on a specified number of folds

for different classification models and reports the AUC scores and ROC curves.

Usage

```
PlatypusML_balance(matrix, label.1, label.2, proportion, random.seed)
```

Arguments

matrix	Matrix. Output of the PlatypusML_extract_features function, with the last column storing the label.
label.1	String. The label of the first class.
label.2	String. The label of the second class.
proportion	Vector of size 2 (floats between 0 and 1 that need to sum up to 1). Specifies the proportions for the two classes. The smaller proportion will be assigned to the minority class by default.
random.seed	Integer. The seed to be set when sampling for balancing the dataset.

Value

This function returns a matrix containing equal number of samples for the two classes.

Examples

```
## Not run:
TODO: example

## End(Not run)
```

PlatypusML_classification

Core ML for crossvalidation

Description

This PlatypusML_classification function takes as input encoded features obtained using the PlatypusML_extract_features function. The function runs cross validation on a specified number of folds for different classification models and reports the AUC scores and ROC curves.

Usage

```
PlatypusML_classification(features, cv.folds, balancing, proportion)
```

Arguments

features	Matrix. Output of the PlatypusML_extract_features function, containing the desired label in the last column.
cv.folds	Integer. The number of folds to be used in cross validation
balancing	Boolean. Whether to perform class balancing. Defaults to TRUE.
proportion	Vector of size 2 (floats between 0 and 1 that need to sum up to 1). Specifies the proportions for the two classes. The smaller proportion will be assigned to the minority class by default. Defaults to c(0.5,0.5).

Value

This function returns a list containing [["combined"]] summary plot with ROC & confusion matrices, [["ROC"]] the ROC curve, [["confusion"]] confusion matrices for each classifier.

Examples

```
## Not run:
To classify and obtain the performance of different models,
using extracted and encoded features.

#extract features
features_VDJ_GP33_binder <- PlatypusML_feature_extraction_VDJ(VGM = VGM,
which.features = c("VDJ_cdr3s_nt"),
which.encoding = c("kmer"),
parameters.encoding.nt = c(3),
which.label = "GP33_binder")

#classify
classifier_GP33_binder <- classification(features = features_VDJ_GP33_binder,
cv.folds = 5,
balancing = TRUE)

#view summary
classifier_GP33_binder$combined

## End(Not run)
```

PlatypusML_feature_extraction_GEX

Extraction of features from GEX matrix of VGM

Description

This PlatypusML_feature_extraction_GEX function takes as input specified features from the second output of the VDJ_GEX_matrix function and encodes according to the specified strategy. The function returns a matrix containing the encoded extracted features as columns and the different cells as rows. This function should be called as a first step in the process of modeling the VGM data using machine learning.

Usage

```
PlatypusML_feature_extraction_GEX(
  VGM,
  encoding.level,
  unique.sequence,
  which.features,
  n.PCs,
  which.label,
  problem,
  verbose.classes,
  platypus.version
)
```

Arguments

VGM	output of the VDJ_GEX_matrix function, containing both VDJ and GEX objects.
encoding.level	String. Specifies on which level the features will be extracted. There are three possible options: "clone" (one random sample per clone), "clone.avg" (average expression per clone), "unique.sequence" (selecting only unique sequences based on a specified sequence (in the unique.sequence argument)). Defaults to "clone.avg".
unique.sequence	String. Needs to be specified only when encoding.level is set to "unique.sequence". The name of the sequence on which unique selection should be based on. Defaults to "VDJ_cdr3s_aa".
which.features	String. Information on which GEX features should be encoded. Options are "varFeatures" (the 1000 most variable features obtained by Seurat::FindVariableFeatures) or "PCs" (the top n PCs, number of PCs to be defined in n.PCs). Defaults to "PCs".
n.PCs	Integer. Number of PCs to be used if choosing which.features == "PCs". Max 50. Defaults to 20.
which.label	String. The name of the column in VGM[[2]] which will be appended to the encodings and used as a label in a chosen ML model later. The label has to be a binary label. If missing, no label will be appended to the encoded features.
problem	String ("classification" or "regression"). Whether the return matrix will be used in a classification problem or a regression one. Defaults to "classification".
verbose.classes	Boolean. Whether to display information on the distribution of samples between classes. Defaults to TRUE. For this parameter to be set to TRUE, classification must all be set to TRUE (default).
platypus.version	This function works with "v3" only, there is no need to set this parameter.

Value

A dataframe containing the encoded features and its label, each row corresponding to a different cell. The label can be found in the last column of the dataframe returned. If which.label="NA" only the encoded features are returned.

Examples

```
## Not run:
To return the encoded gene expression in form of the 20 PCs at the clone level
(average expression per clone).
Attaching the "GP33_binder" label to be used in downstream ML models.

features_PCs_GP33_binder <- PlatypusML_feature_extraction_GEX(
VGM = VGM,
encoding.level = "clone.avg",
which.features = "PCs",
n.PCs = 20,
which.label = "GP33_binder")

To return the encoded gene expression in form of the 1000 most variable features
(genes) at the clone level.
Attaching the "GP33_binder" label to be used in downstream ML models.

features_varFeatures_GP33_binder <- PlatypusML_features_extraction_GEX(
VGM = VGM,
encoding.level = "clone",
which.features = "varFeatures",
which.label = "GP33_binder")

## End(Not run)
```

PlatypusML_feature_extraction_VDJ

Extraction of features from VDJ table of VGM

Description

This PlatypusML_feature_extraction function takes as input specified features from the first output of the VDJ_GEX_matrix function and encodes according to the specified strategy. The function returns a matrix containing the encoded extracted features in the order specified in the input as columns and the different cells as rows. This function should be called as a first step in the process of modeling the VGM data using machine learning.

Usage

```
PlatypusML_feature_extraction_VDJ(
  VGM,
  which.features,
```

```
  which.encoding,
  encoding.level,
  unique.sequence,
  parameters.encoding.nt,
  parameters.encoding.aa,
  which.label,
  problem,
  verbose.classes,
  platypus.version
)
```

Arguments

VGM	VGM output of the VDJ_GEX_matrix function
which.features	String vector. Information on which columns of the VDJ input the function should encode
which.encoding	String vector of size 2. Defaults to 'onehot'. Information on which encoding strategy to be used for the two types of sequences: the first entry of the vector corresponds to the nucleotide type of encoding and the second one to the amino acid type of encoding. If one type of sequence is not among the Other possible values for amino acid sequences are 'kmer', 'blosum', 'dc', 'tc' or 'topoPCA' and for nucleotide sequences 'kmer'.
encoding.level	String. Specifies on which level the features will be extracted. There are three possible options: "cell" (all available), "clone" (one unique sample per clone), "unique.sequence" (selecting only unique sequences based on a specified sequence (int he unique.sequence argument)). It defaults to cell.
unique.sequence	String. Needs to be specified only when encoding.level is set to "unique.sequence". The name of the sequence on which unique selection should be based on.
parameters.encoding.nt	List. Parameters to be used for encoding, if the chosen encoding requires it. 'onehot' -> no parameters necessary, defaults to NULL 'kmer' -> one parameter necessary to set the length of the subsequence, defaults to 3
parameters.encoding.aa	List. Parameters to be used for encoding, if the chosen encoding requires it. 'onehot', 'dc', 'tc' -> no parameters necessary, defaults to NULL 'kmer' -> one parameter necessary to set the length of the subsequence, defaults to 3 'blosum' -> two parameters necessary: k (The number of selected scales (i.e. the first k scales) derived by the substitution matrix. This can be selected according to the printed relative importance values.) and lag (The lag parameter. Must be less than the amino acids.). They default to (5, 7). 'topoPCA' -> three parameters necessary: index (Integer vector. Specify which molecular descriptors to select from the topological descriptors), pc (Integer. Number of principal components. Must be no greater than the number of amino acid properties provided.) and lag(The lag parameter. Must be less than the amino acids.). They default to (c(1:78),5,7).
which.label	String. The name of the column in VDJ which will be used as a label in a chosen model later. If missing, no label will be appended to the encoded features.

problem String ("classification" or "regression"). Whether the return matrix will be used in a classification problem or a regression one. Defaults to "classification".

verbose.classes
Boolean. Whether to display information on the distribution of samples between classes. Defaults to TRUE. For this parameter to be set to TRUE, classification must all be set to TRUE (default).

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

Value

A dataframe containing the encoded features and its label, each row corresponding to a different cell. The encodings are ordered as they have been entered in the 'which.features' parameter. The label can be found in the last column of the dataframe returned.

Examples

```
## Not run:
To return the encoded 'VDJ_cdr3s_nt' sequences using 3mer encoding for nt sequences.
Attaching the "GP33_binder" label to be used in downstream ML models.

features_VDJ_GP33_binder <- PlatypusML_feature_extraction_VDJ(VGM = VGM,
which.features = c("VDJ_cdr3s_nt"),
which.encoding = c("kmer"),
parameters.encoding.nt = c(3),
which.label = "GP33_binder")

## End(Not run)
```

---

select.top.clone *Get the index of top ranking clones.*

Description

Get the index of top ranking clones.

Usage

```
select.top.clone(clonotypes, top.n)
```

Arguments

clonotypes	The output "clonotypes" dataframe from simulation output.
top.n	The top n abundant clones to be selected.

Value

a vector of indexes of top ranking clones

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

Spatial_celltype_plot *Plotting celltype assign to cell according to their phenotype on the spatial image.*

Description

Plotting celltype assign to cell according to their phenotype on the spatial image.

Usage

```
Spatial_celltype_plot(
  sample_names,
  bcs_merge,
  images_tibble,
  vgm_GEX,
  title,
  size,
  legend_title,
  unclassified_cells = c(TRUE, FALSE),
  specific_celltype = c("T", "B", "No", "Unclassified"),
  density = c(TRUE, FALSE)
)
```

Arguments

sample_names	Character vector containing the name of the sample.
bcs_merge	Data frame containing imagerow, imagecol and barcode of the cells belonging to the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 10.
images_tibble	Tbl-df containing the sample name, grob, height and width of the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 5.
vgm_GEX	Data frame containing GEX information (VGM[[2]]). It must have a barcode column containing GEX_barcode and a cell.state column (output from GEX_phenotype).
title	Character vector to name the plot.
size	Number, to define the size of the text, default = 15.
legend_title	Character vector to name the legend scale.
unclassified_cells	Booleans, if TRUE the unclassified cells are also plot and if FALSE they aren't plot exept if the parameter specific_celltype = "Unclassified". In this case the unclassified cells are displayed even unclassified_cells = FALSE. Default = FALSE.
specific_celltype	Character vector, the user can choose to express a specific celltype like T, B or Unclassified cells. Default = No.
density	Booleans, if TRUE a density map is made. Default = FALSE

Value

Returns a ggplot of the celltypes and if density = TRUE a density map of the cells on the spatial image.

Examples

```
## Not run:
Spatial_celltype_plot(bcs_merge = scaling_parameters[[10]],
vgm_GEX = vgm_spatial$GEX@meta.data,images_tibble = scaling_parameters[[5]],
sample_names = sample_names,title="B and T celltype", legend_title = "Celltype",
unclassified_cells = FALSE, specific_celltype = "Unclassified")

## End(Not run)
```

Spatial_cluster *Plotting clusters of cells by choosing between 10X Genomics clustering or reclustering the cells.*

Description

Plotting clusters of cells by choosing between 10X Genomics clustering or reclustering the cells.

Usage

```
Spatial_cluster(
  cluster = c("GEX_cluster", "reclustering"),
  GEX.out.directory.list,
  vgm_VDJ,
  vgm_cluster,
  sample_names,
  bcs_merge,
  images_tibble,
  title,
  size,
  legend_title
)
```

Arguments

cluster	Character vector to describe the clustering, "GEX_cluster" is for plotting 10X Genomics clustering and "reclustering" is for reclustering the cells according to the given subset.
GEX.out.directory.list	Character vector that give the path to filtered_feature_bc_matrix data.
vgm_VDJ	Data frame containing cell of interest and x and y coordinates and GEX_barcode.
vgm_cluster	Data frame containing GEX barcode and cluster given by 10X Genomics. Only needed if cluster parameter is set to "GEX_cluster".
sample_names	Character vector containing the name of the sample.
bcs_merge	Data frame containing imagerow, imagecol and barcode of the cells belonging to the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 10.
images_tibble	Tbl-df containing the sample name, grob, height and width of the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 5.
title	Character vector to name the plot.
size	Number, to define the size of the text, default = 15.
legend_title	Character vector to name the legend scale.

Value

If plotting = TRUE, returns a list containing [[1]] the plot of the selected cells according to their group, [[2]] a data frame that contains the column seurat_clusters with the new cluster. If plotting = FALSE, it returns just the data frame.

Examples

```
## Not run:
#Clustering of whole cells regardless of cell type
GEX_cluster_B_cells<-Spatial_cluster(cluster = "GEX_cluster",
```

```
vgm_cluster = vgm_with_simulated_VDJ$spatial$cluster[[1]],
vgm_VDJ = vgm_with_simulated_VDJ$VDJ,
GEX.out.directory.list = GEX.out.directory.list[[1]],images_tibble=scaling_parameters[[5]],
bcs_merge=scaling_parameters[[10]], title = "B cells",
sample_names = sample_names, legend_title = "GEX clusters" )
GEX_cluster_B_cells[[1]]

#Reclustering with only B cells
reclustering_B_cells<-Spatial_cluster(cluster = "reclustering",
vgm_VDJ = vgm_with_simulated_VDJ$VDJ,
GEX.out.directory.list = GEX.out.directory.list[[1]],
images_tibble=scaling_parameters[[5]],bcs_merge=scaling_parameters[[10]],
title = "B cells", sample_names = sample_names, legend_title = "Reclustering")
reclustering_B_cells[[1]]

## End(Not run)
```

`Spatial_density_plot` *Plotting the contour density of selected cells or of all cells.*

Description

Plotting the contour density of selected cells or of all cells.

Usage

```
Spatial_density_plot(
  sample_names,
  bcs_merge,
  images_tibble,
  vgm_VDJ,
  title,
  size
)
```

Arguments

`sample_names`	Character vector containing the name of the sample.
`bcs_merge`	Data frame containing imagerow, imagecol and barcode of the cells belonging to the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 10.
`images_tibble`	Tbl-df containing the sample name, grob, height and width of the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 5.
`vgm_VDJ`	Data frame containing all the data on the cell. It must contain the column clonotype_id which describes the number of the clonotype to which the cell belongs. This data frame can be obtained by the assignment functions (VDJ_assignment_random_based, VDJ_assignment_density_based and VDJ_assignment_germline_based).

title	Character vector to name the plot.
size	Number, to define the size of the text, default = 15.

Value

Returns a plot of cell contour density on the spatial image.

Examples

```
## Not run:
#Assignment density-based
density_BCR_assignment<-Spatial_VDJ_assignment(GEX_matrix = GEX_matrix,
vgm = vgm_with_simulated_VDJ,
vgm_VDJ = vgm_with_simulated_VDJ$VDJ, celltype = "B",
simulated_VDJ = simulated_B_cells_VDJ,
method = "density")
vgm_with_simulated_VDJ$VDJ<-density_BCR_assignment

top_1_VDJ_BCR_density_data<-Spatial_selection_expanded_clonotypes(
nb_clonotype = 1, vgm_VDJ = vgm_with_simulated_VDJ$VDJ)

p_spatial_BCR_density_clonotype_density<-Spatial_density_plot(
vgm_VDJ = top_1_VDJ_BCR_density_data,
images_tibble = scaling_parameters[[5]],
bcs_merge = scaling_parameters[[10]],sample_names = sample_names,
title = "B cell density assignment")
p_spatial_BCR_density_clonotype_density

## End(Not run)
```

Spatial_evolution_of_clonotype_plot

Plotting the phylogenetic network of a clonotype based on the somatic hypermutations of the immune repertoire sequences on a spatial image.

Description

Plotting the phylogenetic network of a clonotype based on the somatic hypermutations of the immune repertoire sequences on a spatial image.

Usage

```
Spatial_evolution_of_clonotype_plot(
  simulation = c(TRUE, FALSE),
  AbForest_output,
  VDJ,
  nb_clonotype,
  simulated_VDJ,
```

```
  tracking_type = c("closest", "all"),
  sample_names,
  bcs_merge,
  images_tibble,
  title,
  size,
  legend_title
)
```

Arguments

simulation	Logical operator, to describe which type of data we want to plot, TRUE if the data are output of Echidna simulation and FALSE if the we use real dataset.
AbForest_output	Igraph of phylogenetic tree of a clonotype of interest found in the large list output from AntibodyForest function, only needed if we use real dataset.
VDJ	Data frame containing VDJ information, found in the vgm made by platypus. It must have x and y coordinates column and the column containing the factor to plot.
nb_clonotype	Numeric, value which designates the clonotype we want to study if we use simulated data (Echidna output).
simulated_VDJ	Large list, output of Echidna simulate_repertoire function. Only needed if we use simulated data.
tracking_type	Integer, to define how daughter cells are linked to mother cells.If "all" parameter it means that each daughter cell is link by all these potential mother cells and if "closest" parameter, only closest potential mother cell is link to the daughter cell.
sample_names	Character vector containing the name of the sample.
bcs_merge	Data frame containing imagerow, imagecol and barcode of the cells belonging to the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 10.
images_tibble	Tbl-df containing the sample name, grob, height and width of the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 5.
title	Character vector to name the plot.
size	Number, to define the size of the text, default = 15.
legend_title	Character vector to name the legend scale.

Value

Plot of phylogenetic network of a clonotype of interest.

Examples

```
## Not run:
Spatial_evolution_of_clonotype_plot(simulation = FALSE,
```

```
tracking_type = "closest",AbForest_output = forest$s1$clonotype10,VDJ=vgm$VDJ,
sample_names = sample_names, images_tibble = scaling_parameters[[5]],
bcs_merge = scaling_parameters[[10]],
title = "Tracking evolution of clonotype 10", legend_title = "nb of SHM" )

Spatial_evolution_of_clonotype_plot(simulation = FALSE,tracking_type = "all",
AbForest_output = forest$s1$clonotype10,VDJ=vgm$VDJ,
sample_names = sample_names, images_tibble = scaling_parameters[[5]],
bcs_merge = scaling_parameters[[10]],
title = "Tracking evolution of clonotype 10", legend_title = "nb of SHM" )

Spatial_evolution_of_clonotype_plot(simulation = TRUE,tracking_type = "closest",
nb_clonotype = 11 ,simulated_VDJ = simulated_B_cells_VDJ,
VDJ =vgm_with_simulated_VDJ$VDJ,bcs_merge = bcs_merge,
images_tibble = scaling_parameters[[5]],title = "B cell density",
legend_title = "nb_of_SHM",sample_names=sample_names)

Spatial_evolution_of_clonotype_plot(simulation = TRUE,tracking_type = "all",
nb_clonotype = 11 ,simulated_VDJ = simulated_B_cells_VDJ,
VDJ =vgm_with_simulated_VDJ$VDJ,bcs_merge = bcs_merge,
images_tibble = scaling_parameters[[5]],title = "B cell density",
legend_title = "nb_of_SHM",sample_names=sample_names)

## End(Not run)
```

`Spatial_marker_expression` *Plotting a gene of interest in selected cells on the spatial image.*

Description

Plotting a gene of interest in selected cells on the spatial image.

Usage

```
Spatial_marker_expression(
  sample_names,
  bcs_merge,
  images_tibble,
  matrix,
  marker,
  GEX_barcode,
  title,
  threshold,
  size
)
```

Arguments

sample_names	Character vector containing the name of the sample.
bcs_merge	Data frame containing imagerow, imagecol and barcode of the cells belonging to the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 10.
images_tibble	Tbl-df containing the sample name, grob, height and width of the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 5.
matrix	Data frame containing all the genes detected per cells. This data frame can be obtained by the scaling_parameters functions.
marker	Character vector containing the name of a gene of interest.
GEX_barcode	Character vector containing the GEX barcode of the cell of interest with the -1 at the end
title	Character vector to name the plot.
threshold	Number, to define the threshold. If threshold = No, plot of the module and if threshold is a number, plot show the cells above the threshold.
size	Number, to define the size of the text, default = 15.

Value

Returns a plot of the level of expression of a gene in cells.

Examples

```
## Not run:
GEX_BCR_barcode<-vgm_with_simulated_VDJ$VDJ$GEX_barcode
GEX_BCR_barcode<-paste0(GEX_BCR_barcode,"-1") #Add -1 at the end of each barcode
#Without expression threshold
Spatial_marker_expression(matrix=scaling_parameters[[9]],
marker="CD3E",bcs_merge=scaling_parameters[[10]],
images_tibble=scaling_parameters[[5]],
GEX_barcode=GEX_BCR_barcode,sample_names=sample_names, title = "B cells",
threshold = "No")

#With expression threshold
Spatial_marker_expression(matrix=scaling_parameters[[9]],
marker="CD3E",bcs_merge=scaling_parameters[[10]],
images_tibble=scaling_parameters[[5]],
GEX_barcode=GEX_BCR_barcode,sample_names=sample_names, title = "B cells",
threshold = 5)

## End(Not run)
```

`Spatial_module_expression` *Plotting the expression of a gene module on the spatial image with or without a threshold.*

Description

Plotting the expression of a gene module on the spatial image with or without a threshold.

Usage

```
Spatial_module_expression(
  sample_names,
  gene.set,
  GEX.out.directory.list,
  bcs_merge,
  images_tibble,
  title,
  size,
  threshold,
  legend_title
)
```

Arguments

`sample_names`	Character vector containing the name of the sample.
`gene.set`	Charcter vector containing the markers name.
`GEX.out.directory.list`	Character vector that give the path to filtered_feature_bc_matrix data.
`bcs_merge`	Data frame containing imagerow, imagecol and barcode of the cells belonging to the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 10.
`images_tibble`	Tbl-df containing the sample name, grob, height and width of the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 5.
`title`	Character vector to name the plot.
`size`	Number, to define the size of the text, default = 15.
`threshold`	Number, to define the threshold. If threshold = No, plot of the module and if threshold is a number, plot show the cells above the threshold.
`legend_title`	Character vector to name the legend scale.

Value

Returns a ggplot of gene module expression.

Examples

```
## Not run:
gene.set <- list() # make empty list
gene.set[[1]] <- c("CD19","XBP1","SDC1") # put gene set in list

#Without expression threshold
Spatial_module_expression(sample_names = sample_names,gene.set = gene.set,
GEX.out.directory.list = GEX.out.directory.list[[1]],bcs_merge = scaling_parameters[[10]],
images_tibble = scaling_parameters[[5]], threshold = "No")

#With expression threshold
Spatial_module_expression(sample_names = sample_names,gene.set = gene.set,
GEX.out.directory.list = GEX.out.directory.list[[1]],bcs_merge = scaling_parameters[[10]],
images_tibble = scaling_parameters[[5]], threshold = 1)

## End(Not run)
```

Spatial_nb_SHM_compare_to_germline_plot

Plotting number of somatic hypermutation of clones compare to the germline sequence of the clonotype.

Description

Plotting number of somatic hypermutation of clones compare to the germline sequence of the clonotype.

Usage

```
Spatial_nb_SHM_compare_to_germline_plot(
  simulation = c(TRUE, FALSE),
  vgm_VDJ,
  AbForest_output,
  nb_clonotype,
  simulated_VDJ,
  sample_names,
  bcs_merge,
  images_tibble,
  title,
  size,
  legend_title
)
```

Arguments

simulation	Logical operator, to describe which type of data we want to plot, TRUE if the data are output of Echidna simulation and FALSE if the we use real dataset.
vgm_VDJ	Data frame containing cell of interest and x and y coordinates and GEX_barcode.

AbForest_output
: Igraph of phylogenetic tree of a clonotype of interest found in the large list output from AntibodyForest function, only needed if we use real dataset.

nb_clonotype
: Numeric, value which designates the clonotype we want to study if we use simulated data (Echidna output).

simulated_VDJ
: Large list, output of Echidna simulate_repertoire function. Only needed if we use simulated data.

sample_names
: Character vector containing the name of the sample.

bcs_merge
: Data frame containing imagerow, imagecol and barcode of the cells belonging to the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 10.

images_tibble
: Tbl-df containing the sample name, grob, height and width of the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 5.

title
: Character vector to name the plot.

size
: Number, to define the size of the text, default = 15.

legend_title
: Character vector to name the legend scale.

Value

Spatial plot with cells colored by number of somatic hypermutation

Examples

```
## Not run:
Spatial_nb_SHM_compare_to_germline_plot(simulation = FALSE,
AbForest_output=forest[[1]][[2]], vgm_VDJ = vgm$VDJ,
images_tibble = scaling_parameters[[5]],bcs_merge = scaling_parameters[[10]],
sample_names = sample_names,
title = "Number of SHM of clonotype 10", legend_title = "nb of SHM")

## End(Not run)
```

Spatial_scaling_parameters

Scaling of the spatial parameters to be able to express the gene expression on the spatial image.

Description

Scaling of the spatial parameters to be able to express the gene expression on the spatial image.

Usage

```
Spatial_scaling_parameters(vgm_spatial, GEX.out.directory.list, sample_names)
```

Arguments

vgm_spatial	List containing the output of VDJ_GEX_matrix function from Platypus with at least the gene expression data and the addition of spatial parameters: image, scalefacor, tissue, cluster and matrix.
GEX.out.directory.list	Path to the filtered feature bc matrix data.
sample_names	Character vector containing the name of the sample.

Value

Returns a list containing all parameters to scale the data on the spatial image. List element [[1]]: images_cl. List element [[2]]: height of the image. List element [[3]]: width of the image. List element [[4]]: grobs. List element [[5]]: images_tibble. List element [[6]]: scales. List element [[7]]: cluster. List element [[8]]: bcs. List element [[9]]: matrix. List element [[10]]: bcs_merge.

Examples

```
## Not run:
scaling_parameters<-Spatial_scaling_parameters(vgm_spatial = vgm_spatial,
GEX.out.directory.list = GEX.out.directory.list,
sample_names = sample_names)

## End(Not run)
```

Spatial_selection_expanded_clonotypes

Selection of VGM[[1]]/VDJ data of the x more expanded clonotypes.

Description

Selection of VGM[[1]]/VDJ data of the x more expanded clonotypes.

Usage

```
Spatial_selection_expanded_clonotypes(nb_clonotype, vgm_VDJ)
```

Arguments

nb_clonotype	Number that describes how many clonotypes we want to extract from the VGM[[1]].
vgm_VDJ	Data frame containing VDJ information, found in the vgm made by platypus. It must have x and y coordinates column and the column containing the factor to plot.

Value

Returns a data frame with only the data belonging to the number of selected clonotypes. The clonotypes being the most expanded ones.

Examples

```
## Not run:
top_5_VDJ_data<-Spatial_selection_expanded_clonotypes(nb_clonotype = 5, vgm_VDJ = vgm$VDJ)

## End(Not run)
```

Spatial_selection_of_cells_on_image

Allows to select an area on the spatial image and to isolate the cells expressed on this part and repeat this process several times.

Description

Allows to select an area on the spatial image and to isolate the cells expressed on this part and repeat this process several times.

Usage

```
Spatial_selection_of_cells_on_image(
  vgm_VDJ,
  alpha,
  bcs_merge,
  images_tibble,
  sample_names,
  nbpoints,
  title,
  size,
  plotting
)
```

Arguments

vgm_VDJ	Data frame containing all the data on the cell. It must contain the column clonotype_id which describes the number of the clonotype to which the cell belongs. This data frame can be obtained by the assignment functions (VDJ_assignment_random_based, VDJ_assignment_density_based and VDJ_assignment_germline_based).
alpha	Number that give the transparency coefficient (value between 0 and 1). If it is not given it will automatically be 0.
bcs_merge	Data frame containing imagerow, imagecol and barcode of the cells belonging to the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 10.
images_tibble	Tbl-df containing the sample name, grob, height and width of the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 5.
sample_names	Character vector containing the name of the sample.

nbpoints	Numerical value that limite the maximum number of mouse click for the selection, default = 100.
title	Character vector to name the plot.
size	Number, to define the size of the text, default = 15.
plotting	Character vector to return (TRUE) or not (FALSE) the plot of the selection

Value

If plotting = TRUE, returns a list containing [[1]] the plot of the selected cells according to their group, [[2]] a data frame that contains all the cells but the selected cells are distinguished. If plotting = FALSE it juste returns the dataframe.

Examples

```
## Not run:
test<-Spatial_selection_of_cells_on_image(
vgm_VDJ = vgm_spatial_simulated$VDJ$B_cells$random_BCR_assignment,
images_tibble = scaling_parameters[[5]],
bcs_merge = scaling_parameters[[10]],sample_names = sample_names,
plotting = TRUE)

## End(Not run)
```

Spatial_VDJ_assignment

Assign simulated immune repertoire sequences (BCR or TCR) simulated by Echidna to transcriptome and location in a spatial image in function of cell type.

Description

Assign simulated immune repertoire sequences (BCR or TCR) simulated by Echidna to transcriptome and location in a spatial image in function of cell type.

Usage

```
Spatial_VDJ_assignment(
  GEX_matrix,
  vgm,
  vgm_VDJ,
  celltype,
  simulated_VDJ,
  method = c("random", "density", "germline")
)
```

Arguments

GEX_matrix	Dataframe containing barcode, imagecol and imagerow from bcs_merge.
vgm	Output of VDJ_GEX_matrix function with already the simulated VDJ data.
vgm_VDJ	Dataframe from VDJ_GEX_matrix output (vgm[[1]]).
celltype	Character designating the cell type that we want to study either "B" or "T".
simulated_VDJ	Large list, output of Echidna simulate_repertoire function. Only needed if we use simulated data.
method	Character to chose the assignment method of BCR or TCR to transcriptomic information, it can be "random", "density" or "germline".

Value

A dataframe corresponding to the VDJ (VGM[[1]]) with GEX_barcode and x, y coordinates column (allowing to localise each BCR or TCR on the spatial image).

Examples

```
## Not run:
#1)Assignment random to GEX
random_BCR_assignment <- Spatial_VDJ_assignment(GEX_matrix = GEX_matrix,
vgm = vgm_with_simulated_VDJ,
vgm_VDJ = vgm_with_simulated_VDJ$VDJ, celltype = "B",
simulated_VDJ = simulated_B_cells_VDJ, method = "random")

#2)Assignment density-based
density_BCR_assignment<-Spatial_VDJ_assignment(GEX_matrix = GEX_matrix,
vgm = vgm_with_simulated_VDJ,
vgm_VDJ = vgm_with_simulated_VDJ$VDJ, celltype = "B",
simulated_VDJ = simulated_B_cells_VDJ, method = "density")

#3)Assignment germline-based
germline_BCR_assignment<-Spatial_VDJ_assignment(GEX_matrix = GEX_matrix,
vgm = vgm_with_simulated_VDJ,
vgm_VDJ = vgm_with_simulated_VDJ$VDJ, celltype = "B",
simulated_VDJ = simulated_B_cells_VDJ, method = "germline")

## End(Not run)
```

Spatial_VDJ_plot *Plotting immune repertoire data as clonotype or isotype for cells on a spatial image.*

Description

Plotting immune repertoire data as clonotype or isotype for cells on a spatial image.

Usage

```
Spatial_VDJ_plot(
  sample_names,
  bcs_merge,
  images_tibble,
  title,
  size,
  legend_title,
  vgm_VDJ,
  analysis
)
```

Arguments

sample_names	Character vector containing the name of the sample.
bcs_merge	Data frame containing imagerow, imagecol and barcode of the cells belonging to the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 10.
images_tibble	Tbl-df containing the sample name, grob, height and width of the spatial image. It can also be created by the function scaling_spatial_image_parameter by selecting the output parameter 5.
title	Character vector to name the plot.
size	Number, to define the size of the text, default = 15.
legend_title	Character vector to name the legend scale.
vgm_VDJ	Data frame containing VDJ information, found in the vgm made by platypus. It must have x and y coordinates column and the column containing the factor to plot.
analysis	Column in the dataframe containing the factor of interest to plot on the spatial image.

Value

Returns a plot of the factor of interest express on a spatial image.

Examples

```
## Not run:
Spatial_VDJ_plot(vgm_VDJ = top_5_VDJ_data,analysis = top_5_VDJ_data$VDJ_cgene,
images_tibble = scaling_parameters[[5]], bcs_merge = scaling_parameters[[10]],
sample_names = sample_names, title = "B cell", legend = "Isotype")

## End(Not run)
```

Spatial_vgm_formation *Addition of the spatial information to the VGM matrix, output of VDJ_GEX_matrix()*

Description

Addition of the spatial information to the VGM matrix, output of VDJ_GEX_matrix()

Usage

```
Spatial_vgm_formation(
  vgm,
  tissue_lowres_image_path,
  scalefactors_json_path,
  tissue_positions_list_path,
  cluster_path,
  matrix_path
)
```

Arguments

vgm	Large list, output of VDJ_GEX_matrix()
tissue_lowres_image_path	Path to file containing the image of the tissue in png format
scalefactors_json_path	Path to a file for converting pixel positions in the original, full-resolution image to pixel positions in the histological image in json format
tissue_positions_list_path	Path to a text file containing a table with rows that correspond to spots in csv format
cluster_path	Path to 10X Genomic clustering file that is not specific for immune cells, in csv format
matrix_path	Path to the filtered feature barcode matrix containing barcode from fixed list of known-good barcode sequences in the h5 format

Value

Returns the input VGM matrix (output of VDJ_GEX_matrix()) with an additional list containing the spatial information.

Examples

```
## Not run:
#Needed spatial files
tissue_lowres_image_path<-list()
tissue_lowres_image_path[[1]]<-c("c:/.../tissue_lowres_image.png")
```

```
scalefactors_json_path<-list()
scalefactors_json_path[[1]]<-c("c:/.../scalefactors_json.json")

tissue_positions_list_path<-list()
tissue_positions_list_path[[1]]<-c("c:/.../tissue_positions_list.csv")

cluster_path<-list()
cluster_path[[1]]<-c("c:/.../analysis/clustering/graphclust/clusters.csv")

matrix_path<-list()
matrix_path[[1]]<-c("c:/.../filtered_feature_bc_matrix/filtered_feature_bc_matrix.h5")

#VGM formation with spatial data
vgm_spatial<-Spatial_vgm_formation(vgm = vgm_without_spatial_data_and_VDJ,
tissue_lowres_image_path = tissue_lowres_image_path,
tissue_positions_list_path = tissue_positions_list_path,
scalefactors_json_path = scalefactors_json_path,
cluster_path = cluster_path, matrix_path = matrix_path)

## End(Not run)
```

`special_v` *special_v a dataframe, of heavy and light chain v gene combination and their probability to be selected for expansion.*

Description

special_v a dataframe, of heavy and light chain v gene combination and their probability to be selected for expansion.

Usage

```
data("special_v")
```

Format

An object of class `data.frame` with 5 rows and 3 columns.

`trans_switch_prob_b` *trans_switch_prob_b The probability for B cell transcriptome states switching. The row names of the matrix are the cell states the cell is switching from, the column names are the cells states the cell is switching to.*

Description

trans_switch_prob_b The probability for B cell transcriptome states switching. The row names of the matrix are the cell states the cell is switching from, the column names are the cells states the cell is switching to.

Usage

```
data("trans_switch_prob_b")
```

Format

A 4*4 matrix. The row and clumn names are: "GerminalcenterBcell","NaiveBcell","Plasmacell","MemoryBcell". The probability for a cell to switch from "GerminalcenterBcell" to "Plasmacell" is the value at trans_switch_prob_b[1,3].

`trans_switch_prob_t` *trans_switch_prob_t The probability for T cell transcriptome states switching. The row names of the matrix are the cell states the cell is switching from, the column names are the cells states the cell is switching to.*

Description

trans_switch_prob_t The probability for T cell transcriptome states switching. The row names of the matrix are the cell states the cell is switching from, the column names are the cells states the cell is switching to.

Usage

```
data("trans_switch_prob_t")
```

Format

A 7*7 matrix. The row and clumn names are: "NaiveCd4","ActivatedCd4","MemoryCd4","NaiveCd8","EffectorCd8","Mem

---

`umap.top.highlight` *Set idents for top abundant clones in Seurat object, get ready for highlight the top abundant clones in UMAP.*

Description

Set idents for top abundant clones in Seurat object, get ready for highlight the top abundant clones in UMAP.

Usage

```
umap.top.highlight(gex, all.contig.annotations, top.n)
```

Arguments

gex	output from get.umap function.
all.contig.annotations	The output dataframe all_contig_annotations from simulation.
top.n	The top n abundant clones to be shown in the plot. If missing, all clones will be shown.

Value

a Seurat object ready for highlight the top abundant clones in UMAP

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_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 = small_vgm[[1]],
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

```
## Not run:
 alpha_beta_VJgene <- VDJ_alpha_beta_Vgene_circos(vgm[[1]])
 # print circos plot:
 alpha_beta_VJgene[[1]]

## End(Not run)
```

`VDJ_analyze` *Platypus V2 VDJ processing wrapper.*

Description

Platypus V2 Processes and organizes the repertoire sequening data from cellranger vdj and returns a list of dataframes, where each dataframe corresponds to an individual repertoire. The function will return split CDR3 sequences, germline gene information, filter out those clones with either incomplete information or doublets (multiple CDR3 sequences for a given chain). This function should be called once for desired integrated repertoire and transcriptome. For example, if there are 3 VDJ libraries and 3 GEX libraries and the goal is to analyze all three GEX libraries together (e.g. one UMAP/tSNE reduction) this then function should be called one time and the three VDJ directories should be provided as input to the single function call.

Usage

```
VDJ_analyze(
  VDJ.out.directory,
  filter.1HC.1LC,
  clonotype.list,
  contig.list,
  filtered.contigs
)
```

Arguments

`VDJ.out.directory`
: Character vector with each element containing the path to the output of cellranger vdj runs. Multiple repertoires to be integrated in a single transcriptome should be supplied as multiple elements of the character vector. This can be left blank if supplying the clonotypes and contig files directly as input. 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).

`filter.1HC.1LC`
: Logical indicating whether only those clones containing 1 VH/TRB and VL/TRA should be maintined for furhter analysis. Default is set to TRUE, which restricts the analysis to only clones with exactly 1 heavy chain and 1 light chain (or 1 beta + 1 alpha in the case of T cells).

`clonotype.list`
: List of dataframes containing clonotyping information for each repertoire. The column names should correspond to the clonotypes.csv file from cellranger vdj output.

`contig.list`
: List of dataframes containing the contig information for each repertoire. The column names should correspond to the all_contigs.csv file from cellranger vdj output.

filtered.contigs
: Logical indicating if the filtered contigs file should be used. TRUE will read VDJ information from only the filtered output of cellranger. FALSE will read the all contigs file from cellranger. Default set to TRUE (filtered output)

Value

Returns a list of dataframes where each dataframe corresponds to one input directory. If only one file is supplied, the output list will only contain one element. This output can be supplied as input to other functions including VDJ_per_clone, VDJ_network, VDJ_germline_genes, VDJ_expansion, visualize_clones_GEX, VDJ_phylo, VDJ_clonotype. Germline gene information is based on the majority of cells within each clonotype. For example, if the majority of cells in clonotype1 have the IGHG1 isotype then then entire clonal family will be determined as IGHG1. For a cell-specific investigation, the output of this function can be supplied to the function VDJ_per_clone, which will provide isotype, sequence, germline gene, etc information for each cell within the each clone.

Examples

```
## Not run:
example.vdj.analyze <- VDJ_analyze(
VDJ.out.directory = "~/path/to/cellranger/vdj/outs/", filter.1HC.1LC = T)

## End(Not run)
```

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

```
## Not run:
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')

## End(Not run)
```

`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 == T. 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 == T 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

```
## Not run:

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 = TRUE, filename = "PnP_seq_example")


## End(Not run)
```

VDJ_bulk_to_vgm *Utility function for bulk data to standard Platypus format conversion*

Description

The VDJ_bulk_to_vgm function converts bulk output files from MIXCR or MAF into a vgm-format compatible with most downstream Platypus functions used for VDJ repertoire analysis.

Usage

```
VDJ_bulk_to_vgm(
  VDJ.bulk.out.directory.list,
  input.type,
  integrate.MIXCR.output,
  vgm.expanded,
  clone.strategy,
  group.id,
  cell.type,
  batches,
  best.match.only
)
```

Arguments

VDJ.bulk.out.directory.list	List containing paths to bulk VDJ output files from MIXCR or MAF. TRUST4 (and TRUST4.FULL) require an RDS file as input
input.type	Character vector. Defaults to "MIXCR". "MIXCR", "MAF", "TRUST4", and "TRUST4.FULL" are supported. "TRUST4.FULL" contains TRUST additional columns, which were not originally supported by vgm: "cdr1", "cdr2", "v_cigar", "d_cigar", "j_cigar", "v_identity", "j_identity", "complete_vdj".

integrate.MIXCR.output
: Boolean. Defaults to TRUE. Whether to include in the VGM output additional MiXCR (49-78) columns.

vgm.expanded
: Boolean. Defaults to TRUE. Whether to include vgm[[9]] in the output list, where vgm[[9]] is the expanded version of vgm[[1]] having 1 line per read. For some Platypus functions, only vgm[[9]] (and not vgm[[1]]) may be compatible.

clone.strategy
: Character vector to specify the clonotyping strategy. Defaults to "cdr3.aa". Note that MIXCR input comes with clonotypes already assigned, and therefore clone.strategy should be specified only when the user wants to change the clonoyping strategy, and if no clone.strategy is provided, re-clonotyping will not be performed. Meanwhile, MAF inputs do not come with the clonotypes pre-assigned. Hence, if no clone.strategy is specified, "cdr3.aa" will be used as the default clonotyping strategy. The clonotyping strategies available in this function are: "cdr3.aa", "VDJJ.VJJ", "VDJJ.VJJ.cdr3length".

group.id
: Numeric vector. Defaults to NA. The user can specify to which group does each file belong to (e.g. a group could correspond to some specific treatment). The length of this numeric vector should match the number of samples in the VDJ.bulk.out.directory.list input.

cell.type
: Character vector. Defaults to NA. Cell type (e.g., "Bcell") of the MIXCR or MAF file that is provided as input.

batches
: Numeric vector. Defaults to NA. An additional grouping parameter that can be specified by the user. The length of this numeric vector should match the number of samples in the VDJ.bulk.out.directory.list input.

best.match.only
: Boolean. Whether only the highest scoring gene (V,J,D,C gene should) should be included in the output, or all matching genes in MIXCR should be included (MAF outputs: for the same read we can only have one possible V,J,D or C gene). Defaults to TRUE.

Value

a VGM object (vgm.bulk.list). vgm.bulk.list[[1]]: each line correspond to a clonotype. vgm.bulk.list[[9]] (if vgm.expanded==TRUE): each line correspond to a read. The other (2-8) entries of the list are left empty for compatibility with Platypus functions.

Examples

```
## Not run:
Run from local directory using MIXCR/MAF bulk VDJ-repertoire files as inputs:
VDJ.bulk.out.directory.list <- list()
VDJ.bulk.out.directory.list[[1]] <- c("~/MIXCR_vdj_cdr3_clonotyping/C4.txt")
VDJ.bulk.out.directory.list[[2]] <- c("~/MIXCR_vdj_cdr3_clonotyping/C6.txt")
bulk.vgm.MIXCR <- VDJ_bulk_to_vgm(VDJ.bulk.out.directory.list = VDJ.bulk.out.directory.list,
input.type = 'MIXCR',
integrate.MIXCR.output = TRUE,
group.id = c(1,2),
cell.type = "Bcells",
batches = c(1,1),
```

```
vgm.expanded = TRUE,
best.match.only = FALSE)

To re-clonotype MIXCR samples based on e.g., the CDR3 a.a. sequence:
bulk.vgm.MIXCR <- VDJ_bulk_to_vgm(VDJ.bulk.out.directory.list = VDJ.bulk.out.directory.list,
input.type = 'MIXCR',
integrate.MIXCR.output = TRUE,
group.id = c(1,2),
cell.type = "Bcells",
batches = c(1,1),
vgm.expanded = TRUE,
best.match.only = FALSE,
clone.strategy = "cdr3.aa")

## End(Not run)
```

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

```
## Not run:
VDJ_call_enclone(vdj, VDJ.directory, samples.to.combine = c('s1', 's2', 's3'), global.clonotype = T)

## End(Not run)
```

VDJ_call_MIXCR *MiXCR wrapper for Platypus V3 VDJ object*

Description

Extracts information on the VDJRegion level using MiXCR on WINDOWS, MAC and UNIX systems for input from both Platypus v2 (VDJ.per.clone) or v3 (Output of VDJ_GEX_matrix) This function assumes the user can run an executable instance of MiXCR and is elgible to use MiXCR as determined by license agreements. ! FOR WINDOWS USERS THE EXECUTABLE MIXCR.JAR HAS TO PRESENT IN THE CURRENT WORKING DIRECTORY ! The VDJRegion corresponds to the recombined heavy and light chain loci starting from framework region 1 (FR1) and extending to frame work region 4 (FR4). This can be useful for extracting full-length sequences ready to clone and further calculating somatic hypermutation occurrences.

Usage

```
VDJ_call_MIXCR(
  VDJ,
  operating.system,
  mixcr.directory,
  species,
  simplify,
  platypus.version
)
```

Arguments

VDJ	For platypus.version = "v2" the output from the VDJ_per_clone function. This object should have information regarding the contigs and clonotype_ids for each cell. For platypus.version = "v3" the VDJ dataframe output of the VDJ_GEX_matrix function (VDJ.GEX.matri.output[[1]])
operating.system	Can be either "Windows", "Darwin" (for MAC) or "Linux". If left empty this is detected automatically
mixcr.directory	The directory containing an executable version of MiXCR. FOR WINDOWS USERS THIS IS SET TO THE CURRENT WORKING DIRECTORY (please paste the content of the MIXCR folder after unzipping into your working directory. Make sure, that mixcr.jar is not within any subfolders.)
species	Either "mmu" for mouse or "hsa" for human. These use the default germline genes for both species contained in MIXCR. Defaults to "hsa"
simplify	Only relevant when platypus.version = "v3". Boolean. Defaults to TRUE. If FALSE the full MIXCR output and computed SHM column is appended to the VDJ If TRUE only the framework and CDR3 region columns and computed SHM column is appended. To discriminate between VDJ and VJ chains, prefixes are added to all MIXCR output columns
platypus.version	Character. Defaults to "v3". Can be "v2" or "v3" dependent on the input format

Value

For platypus.version = "v3" returns input VDJ dataframe supplemented with MIXCR output information. For platypus.version = "v2" returns a nested list containing VDJRegion information as determined by MIXCR. The outer list corresponds to the individual repertoires in the same structure as the input VDJ.per.clone. The inner list corresponds to each clonal family, as determined by either the VDJ_clonotype function or the defaul nucleotide clonotyping produced by cellranger.Each element in the inner list corresponds to a dataframe containing repertoire information such as isotype, CDR sequences, mean number of UMIs. This output can be supplied to further package functions such as VDJ_extract_sequences and VDJ_GEX_integrate.

See Also

VDJ_extract_sequences

Examples

```
## Not run:
#For platypus version 2
VDJ_call_MIXCR(VDJ = VDJ.per.clone.output,
mixcr.directory = "~/Downloads/mixcr-3.0.12/mixcr",species = "mmu")

#For platypus version 3 on a Windows system
VDJ_call_MIXCR(VDJ = VDJ_GEX_matrix.output[[1]],
mixcr.directory = "WILL BE SET TO CURRENT WORKING DIRECTORY",
species = "mmu", platypus.version = "v3", simplify = TRUE)

## End(Not run)
```

VDJ_call_MIXCR_full *MiXCR wrapper for Platypus V3 VDJ object. In addition to the VDJ_call_MIXCR function, the output also contains the concatenated sequences from FR1 all the way to FR2 for both the VDJ and VJ.*

Description

Extracts information on the VDJRegion level using MiXCR on WINDOWS, MAC and UNIX systems for input from both Platypus v2 (VDJ.per.clone) or v3 (Output of VDJ_GEX_matrix) This function assumes the user can run an executable instance of MiXCR and is elgible to use MiXCR as determined by license agreements. ! FOR WINDOWS USERS THE EXECUTABLE MIXCR.JAR HAS TO PRESENT IN THE CURRENT WORKING DIRECTORY ! The VDJRegion corresponds to the recombined heavy and light chain loci starting from framework region 1 (FR1) and extending to frame work region 4 (FR4). This can be useful for extracting full-length sequences ready to clone and further calculating somatic hypermutation occurrences. In addition to the VDJ_call_MIXCR function, the output also contains the concatenated sequences from FR1 all the way to FR2 for both the VDJ and VJ.

Usage

```
VDJ_call_MIXCR_full(
  VDJ,
  mixcr.directory,
  species,
  platypus.version,
  operating.system,
  simplify
)
```

Arguments

VDJ	For platypus.version = "v2" the output from the VDJ_per_clone function. This object should have information regarding the contigs and clonotype_ids for each cell. For platypus.version = "v3" the VDJ dataframe output of the VDJ_GEX_matrix function (VDJ.GEX.matri.output[[1]])

`mixcr.directory`
: The directory containing an executable version of MiXCR. FOR WINDOWS USERS THIS IS SET TO THE CURRENT WORKING DIRECTORY (please paste the content of the MIXCR folder after unzipping into your working directory. Make sure, that mixcr.jar is not within any subfolders.)

`species`
: Either "mmu" for mouse or "hsa" for human. These use the default germline genes for both species contained in MIXCR. Defaults to "hsa"

`platypus.version`
: Character. Defaults to "v3". Can be "v2" or "v3" dependent on the input format

`operating.system`
: Can be either "Windows", "Darwin" (for MAC) or "Linux". If left empty this is detected automatically

`simplify`
: Only relevant when platypus.version = "v3". Boolean. Defaults to TRUE. If FALSE the full MIXCR output and computed SHM column is appended to the VDJ If TRUE only the framework and CDR3 region columns and computed SHM column is appended. To discriminate between VDJ and VJ chains, prefixes are added to all MIXCR output columns

---

`VDJ_call_RECON` *Calls the Kaplinsky/RECON tool*

Description

Calls the Kaplinsky/RECON tool on the VDJ/VDJ.GEX.matrix[[1]] object to infer the parent distribution of species and estimate their diversity. Outputs either a dataframe of the resulting means and weights of the RECON species parent distribution estimation or a plot of the original species distribution along resampled values from the reconstructed parent distribution.

Usage

```
VDJ_call_RECON(
  VDJ,
  recon.directory,
  feature.columns,
  grouping.column,
  VDJ.VJ.1chain,
  max.features,
  size.threshold,
  resample,
  max.feature.size,
  reticulate,
  operating.system
)
```

Arguments

VDJ	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
recon.directory	directory containing recon executable. Defaults to working directory/Recon
feature.columns	vector of strings/ string - columns denoting the unique species for RECON - e.g., could be the CDRH3s if feature.columns = 'VDJ_cdr3s_aa'. If more than one column is provided (e.g. c("VDJ_cdr3s_aa","VJ_cdr3s_aa")) these columns will be pasted together.
grouping.column	string - column determining the groups/ samples for the species observations. Defaults to 'sample_id' for per-repertoire analysis
VDJ.VJ.1chain	boolean, defaults to TRUE. Whether to filter out aberrant cells (more than 1 VDJ or VJ chain).
max.features	integer or 'all' - maximum number of features/species ot be considered for the RECON estimation. If 'all', will consider all species.
size.threshold	integer - the size threshold parameter for the RECON tool, as specified by the '-t' parameter.
resample	boolean - if T, will also perform and output a resample of the species frequencies/sizes from the inferred parent distribution.
max.feature.size	integer - the maximum size of species/features to be considered in the resulting plot (maximum number of elements on the x axis).
reticulate	boolean - if T, will create a new environment to install python and run the RECON tool, else, your environment must have a python version compatible with RECON installed.
operating.system	string - operating system on which RECON will be run. 'Windows' for Windows, 'Linux' for Linux, 'Darwin' for MacOS.

Value

The resulting means and weights of the RECON-inferred distribution as a seprate dataframe or appended to the VDJ, or a plot of resampled species sizes from the inferred distribution vs original sizes/frequencies.

Examples

```
## Not run:
VDJ_call_RECON(VDJ, recon.directory='./Recon',
feature.columns = 'VDJ_cdr3s_aa', grouping.column = 'VDJ_cdr3s_aa')

## End(Not run)
```

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

```
## Not run:
# manual replotting of Circos plot:
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")


## End(Not run)
```

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 v3 input. Only available for Platypus "v3" available. For 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
)
```

Arguments

`VDJ`	VDJ dataframe generated using the VDJ_GEX_matrix function (VDJ_GEX_matrix.output[[1]])
`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.

Value

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

Examples

```
#Standard B cell plot for platypus version v3
#Will generate one plot per sample (from sample_id column)
clonal_out <- VDJ_clonal_expansion(VDJ = Platypus::small_vgm[[1]],
 celltype = "Bcells", clones = 30,subtypes = FALSE, species = "Mouse"
```

```
 ,treat.incomplete.clones = "exclude"
 ,treat.incomplete.cells = "proportional")

#Regrouped and recolored plot in v3
#Will generate a plot for each sample.
#Bars are filled by the sample with the highest proportion of cells in a given clonotype
clonal_out <- VDJ_clonal_expansion(VDJ = Platypus::small_vgm[[1]]
, celltype = "Bcells", clones = 30,subtypes = FALSE, species = "Mouse"
,treat.incomplete.clones = "exclude"
,treat.incomplete.cells = "proportional"
,color.by = "seurat_clusters") #change grouping with group.by = "column name"
clonal_out[[1]] #list of plots
clonal_out[[2]] #list of source dataframes

#T cell plot with recoloring by vgene
#VDJ_clonal_expansion(VDJ = Platypus::small_vgm[[1]]
#,celltype = "Tcells", clones = 30, group.by = "sample_id"
#,color_by = "VDJ_vgene")

#Plotting only IgD clones. Increased the value for clones to scan more of the dataset
#VDJ_clonal_expansion(VDJ = Platypus::small_vgm[[1]]
#,celltype = "Bcells", clones = 150,subtypes = FALSE
#,species = "Mouse",treat.incomplete.clones = "include"
#,treat.incomplete.cells = "proportional", isotypes.to.plot = "IGHD")

#Plotting only clones containing cells with the IGHG2c isotype (For murine data only!)
#VDJ_clonal_expansion(VDJ = Platypus::small_vgm[[1]]
#,celltype = "Bcells", clones = 150,subtypes = TRUE, species = "Mouse"
#,treat.incomplete.clones = "exclude"
#,treat.incomplete.cells = "proportional", isotypes.to.plot = "IGHG2c")
```

`VDJ_clonal_expansion_abundances` *Wrapper function for VDJ_abundances to obtain ranked clonotype barplots*

Description

Wraps the VDJ_abundances function and output a barplot of clonotypes ranked by expansion (x axis) with counts of the specific feature values per clonotype (y axis). For a more in-depth configuration of the barplots (e.g., including clonotypes with missing features, different strategies for NA values, etc.), use the VDJ_abundances function with output.format='plots'.

Usage

```
VDJ_clonal_expansion_abundances(
  VDJ,
  features,
```

```
  count.level,
  max.clonotypes,
  rank.clonotypes,
  specific.feat.colors
)
```

Arguments

VDJ	VDJ or VDJ.GEX.matrix[[1]] object, as obtained from the VDJ_GEX_matrix function in Platypus.
features	string or vector of strings, denoting the columns of the VDJ/VDJ.GEX.matrix[[1]] object from which to extract the unique feature values.
count.level	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.
max.clonotypes	integer or NULL, the maximum number of clonotypes for which to count features. If NULL, it will count for all clonotypes.
rank.clonotypes	boolean, if T - clonotypes will be ranked and order according to their expansion.
specific.feat.colors	named list (or NULL) of specific colors to be used in the final barplots.

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

```
## Not run:
VDJ_clonal_expansion_abundances(VDJ = small_vgm[[1]],
features='VDJ_cgene',count.level='absolute',
max.clonotypes=30, rank.clonotypes=T, specific.feat.colors=NULL)

## End(Not run)
```

VDJ_clonal_lineages *Platypus V2 lineage utility*

Description

Only Platypus V2 Organizes and extracts full-length sequences for clonal lineage inference. The output sequence can either contain the germline sequence as determined by cellranger or can just contain the sequences contained in each clonal family.

Usage

```
VDJ_clonal_lineages(
  VDJ,
  VDJ_extract_germline.output,
  as.nucleotide,
  with.germline,
  platypus.version
)
```

Arguments

VDJ	For platypus v2 the output of the call_MIXCR function containing the full-length VDJRegion sequences.For v3 the VDJ matrix output of the VDJ_GEX_matrix function ran with trim.and.align = TRUE. (VDJ_GEX_matrix.output[[1]])
VDJ_extract_germline.output	The output from the VDJ_extract_germline function. This should have the germline information. This needs to be supplied if the with.germline argument is set to true.
as.nucleotide	Logical determining whether the full-length VDJRegion sequence should use nucleotide seqeunce. TRUE indicates nucleotide sequences and FALSE will extract amino acid sequences.
with.germline	Logical determining whether the germline sequence as determined by cellranger should be included in the output list of sequences. If so, the germline will be added to the last row of each dataframe object.
platypus.version	Default is "v3".

Value

returns a list containing the sequences for each clonal family as determined by the input clonotyping strategy to call_MIXCR and VDJ_extract_germline. The outer list corresponds to distinct repertoires supplied to the call_MIXCR function (e.g. VDJ.clonal.lineage.output[[i]][[j]] will contain a dataframe of the j'th clone in the i'th repertoire)

Examples

```
## Not run:
clonal_lineages <- VDJ_clonal_lineages(VDJ=call_MIXCR_output,
VDJ_extract_germline.output=VDJ_extract_germline_output,as.nucleotide=F,with.germline=T)

## End(Not run)
```

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)

reclonotyped_vgm <- VDJ_clonotype(VDJ=Platypus::small_vgm[[1]],
clone.strategy="cdr3.homology", homology.threshold = 0.5,
hierarchical = "single.chains", 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

```
## Not run:
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")

## End(Not run)
```

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

```
## Not run:
VDJ_db_annotate(VDJ=VDJ,db.list=db.list,database.features='Epitope',match='cdr3.aa',homology=FALSE)

## End(Not run)
```

`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

```
## Not run:
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')

## End(Not run)
```

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_GEX_matrix 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

```
#Calculate shannon index for VDJ CDR3s by sample
plot <- VDJ_diversity(VDJ = Platypus::small_vgm[[1]],
,feature.columns = c("VDJ_cdr3s_aa"), grouping.column = "sample_id"
,metric = "shannon")

#Calculate Gini-simpson and Simpson index for VDJ and VJ CDR3s by sample
VDJ_diversity(VDJ = Platypus::small_vgm[[1]],
,feature.columns = c("VDJ_cdr3s_aa","VJ_cdr3s_aa"), grouping.column = "sample_id"
,metric = "ginisimpson")

#Calculate Jaccard index of J gene usage between two samples
VDJ_diversity(VDJ = Platypus::small_vgm[[1]],
,feature.columns = c("VDJ_jgene"), grouping.column = "sample_id"
,metric = "jaccard")
```

VDJ_dublets *Platypus V2 annotation utility*

Description

Only Platypus v2 Produces a matrix indicating either the number of cells or clones which contain multiple heavy or light chains (or alpha/beta in the case of T cells).

Usage

```
VDJ_dublets(clonotype.list, clone.level)
```

Arguments

clonotype.list	Output from VDJ_analyze function. This should be a list of clonotype dataframes, with each list element corresponding to a single VDJ repertoire.
clone.level	Logical indicating whether the matrix should display information on the clone level. TRUE will result in matrices containing information about the number of chains on the clonal level. FALSE will result in matrices depicting the numnber of cells.

Value

Returns a list of matrices containing the number of heavy/light chains per either cell or clone depending on the clone.level parameter. This can then be supplied to heatmap functions directly. Each list element corresponds to each of the input list elements of clonotypes.

Examples

```
## Not run:
example.vdj.analyze <- VDJ_dublets(clonotype.list = "VDJ.analyze.output", clone.level=T)

## End(Not run)
```

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 = 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_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_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_GEX_matrix function (VDJ_GEX_matrix.output[[1]])
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)

reclonotyped_vgm <- VDJ_clonotype(VDJ=Platypus::small_vgm[[1]],
clone.strategy="cdr3.homology", homology.threshold = 0.5,
hierarchical = "single.chains", global.clonotype = TRUE)
```

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

VDJ_extract_germline *Platypus V2 utility for full germline sequence via MiXCR*

Description

Only Platypus v2. Extracts the full-length germline sequence as determined by cellranger. This function returns an object that now contains the reference germline for each of the clones. If multiple clones (as determined by cellranger) have been merged using the VDJ_clonotype function then these sequences may have distinct germline sequences despite being in the same clonal family (nested list). This is particularly possible when homology thresholds were used to determine the clonotypes.

Usage

```
VDJ_extract_germline(
  VDJ.per.clone,
  mixcr.directory,
  extract.VDJRegion,
  species
)
```

Arguments

VDJ.per.clone	The output from the VDJ_per_clone function. This object should have information regarding the contigs and clonotype_ids for each cell.
mixcr.directory	The directory containing an executable version of MiXCR. This must be downloaded separately and is under a separate license.
extract.VDJRegion	Default is TRUE. Future iterations will allow for distinct gene regions to be extracted.
species	Either "mus" or "hsa" for mouse and human respectively. Default is set to mouse.

Value

Returns a dataframe containing repertoire information, such as isotype, CDR sequences, mean number of UMIs. This output can be supplied to furhter packages VDJ_extract_sequences and VDJ_GEX_integrate

Examples

```
## Not run:
VDJ_extract_germline(VDJ.per.clone=VDJ.per.clone.output
,mixcr.directory="~/Downloads/mixcr-3.0.12/mixcr"
,extract.VDJRegion=T,species = "mmu")

## End(Not run)
```

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

VDJ_GEX_clonal_lineage_clusters

Platypus V2 lineage - GEX integration utility

Description

only Platypus v2 Integrates the transcriptional cluster information into the clonal lineages. This requires that automate_GEX, VDJ_clonal_lineages, and VDJ_GEX_integrate have already been ran. The transcriptional cluster will be added to the end of the Name for each sequence.

Usage

```
VDJ_GEX_clonal_lineage_clusters(
  VDJ_GEX_integrate.output,
  VDJ_clonal_lineages.output
)
```

Arguments

`VDJ_GEX_integrate.output`
: The output from the VDJ_GEX_integrate function that is performed on the VDJ_per_clone level. This involves a nested list where the outer list corresponds to the repertoire and inner lists correspond to specific clones based on the clonotyping strategy.

`VDJ_clonal_lineages.output`
: Output from VDJ_clonal_lineages. This should be nested list, with the outer list element corresponding to the individual repertoire and the inner list corresponding to individual clonal lineages based on the initial clonotyping strategy in the form of a dataframe with either Seq or Name. The Name currently contains the barcode following the last "_".

Value

a nested list in the identical format to the VDJ_clonal_lineages.output but the name of each sequence will have been changed to include the transcriptional cluster corresponding to that barcode from the GEX library. This requires first running the

Examples

```
## Not run:
clonal_lineages <- VDJ_clonal_lineages(call_MIXCR.output=call_MIXCR_output
, VDJ_extract_germline.output=VDJ_extract_germline_output
,as.nucleotide=FALSE,with.germline=TRUE)

## End(Not run)
```

VDJ_GEX_clonotyme *Pseudotime analysis for scRNA and repertoire sequencing datasets*

Description

Pseudotime analysis for scRNA and repertoire sequencing datasets

Usage

```
VDJ_GEX_clonotyme(
  method,
  version,
  top.N.clonotypes,
  vdj.gex.matrix.output,
  vdj.analyze.output,
  gex.automate.output,
  exclude.clusters,
  colors,
  show.cells,
  highlight.genes,
  dropest.output.list,
  velocyto.gex.merged,
  velocyto.file.name,
  velocyto.out.dir,
  velocyto.save.rds,
  velocyto.norm.scale.factor,
  velocyto.n.variable.features,
  velocyto.neighbor.dim,
  velocyto.cluster.resolution,
  velocyto.mds.dim,
  velocyto.nCount_spliced,
  velocyto.percent.mt,
  velocyto.normalisation.method,
  velocyto.selection.method,
  velocyto.deltaT,
  velocyto.kCells,
  velocyto.fit.quantile,
  velocyto.kGenes,
```

```
  root.selection,
  root.marker,
  ridgeline.separator,
  genes.for.module.score,
  root.nodes,
  color.cells
)
```

Arguments

method	Pseudotime analylysis method to be used. Possible parameters are monocle3 or velocyto. monocle3 is being used as a default method. For velocyto analysis please run on Cluster and it is only available for UNIX based systems.
version	Platypus version to use "v2" or "v3". version 2 used by default.
top.N.clonotypes	How many clonotypes to show per sample in the Ridgeline plots and on the Velocyto UMAP.
vdj.gex.matrix.output	If Platypus v3 is used, the input to this function has to be the output of the VDJ_GEX_matrix function.
vdj.analyze.output	If Platypus v2 is used, the VDJ_analyze output has to be supplied.
gex.automate.output	If Platypus v2 is used, the GEX_automate output has to be supplied here.
exclude.clusters	Please enter a cluster number if you'd like to exclude a certain cluster from analysis. Cluster will be assigned to different partition in Monocle3 analysis and therefore pseudotime distance will be set to infinity. Cells from this cluster will be deleted from the dataset in the Velocyto analysis.
colors	Vector containing custom colors to be used for highlighting the clonotypes. If left empty, default colors will be assigned.
show.cells	Logical, should cells be shown in the Ridgeline plots. True by default.
highlight.genes	Vector containing gene names. The expressionlevels of these genes along pseudotime will be plotted.
dropest.output.list	List containing the cell.counts.matrices.rds from the Dropest alignment for Velocyto analysis.
velocyto.gex.merged	Logical whether samples should be shown in combined UMAP or sepeartely.
velocyto.file.name	String used as file name when saving the output pdf
velocyto.out.dir	Directory to save the output files. By default the current working directory.
velocyto.save.rds	If RDS objects should be saved as well. Default = F.

`velocyto.norm.scale.factor`
: Parameter for GEX analysis of the cell.count.matrices.

`velocyto.n.variable.features`
: Parameter for GEX analysis of the cell.count.matrices.

`velocyto.neighbor.dim`
: Parameter for GEX analysis of the cell.count.matrices.

`velocyto.cluster.resolution`
: Parameter for GEX analysis of the cell.count.matrices.

`velocyto.mds.dim`
: Parameter for GEX analysis of the cell.count.matrices.

`velocyto.nCount_spliced`
: Cutoff threshold. cells with less spliced gene counts will be omitted. Filtering of bad quality cells.

`velocyto.percent.mt`
: Parameter for GEX analysis of the cell.count.matrices.

`velocyto.normalisation.method`
: Parameter for GEX analysis of the cell.count.matrices.

`velocyto.selection.method`
: Parameter for GEX analysis of the cell.count.matrices.

`velocyto.deltaT`
: Parameter for Velocyto analysis

`velocyto.kCells`
: Parameter for Velocyto analysis

`velocyto.fit.quantile`
: Parameter for Velocyto analysis

`velocyto.kGenes`
: Parameter for Velocyto analysis

`root.selection` Character. Method for root selection. Defaults to "manual"

`root.marker` Character. Marker to use as Root. Defaults to "SELL"

`ridgeline.separator`
: Character. Variable to group ridgeline plots by. Defaults to "clonotype"

`genes.for.module.score`
: List of vectors of genes. With module scores inferred via Seurat::AddModuleScore(). Default is set to NULL.

`root.nodes` Labeled node from trajectory plot to specify root nodes root nodes for pseudotime trajectory.

`color.cells` For the module score plot decide how the cells should be coloured (based on e.g. group_id, seurat_clusters etc.). Default = 'seurat_clusters'

Value

If method=monocle3, the function returns a list element: [[1]] UMAP colored by Pseudotime, [[2]] Ridgeline plots showing the density of each of the top.N.clonotypes per cluster along pseudotime., [[3]] Gene expression plots highlighting the gene expression across pseudotime colored by transcriptional cluster, [[4]] Gene expression plots highlighting the gene expression across pseudotime colored by colotype. If method=velocyto, plots and RDS will be saved to velocyto.out.dir.

Examples

```
## Not run:
#----Method=monocle3----

# Version 2
   vdj_repertoire_tcells <- VDJ_analyze(VDJ.out.directory =VDJ.out.directory.list,
    filter.1HC.1LC = T)
   gex_acute <- Platypus::GEX_automate(GEX.outs.directory.list = dir_to_gex[1:1],
    integration.method = "scale.data", mito.filter = 20, cluster.resolution = 0.5,
   VDJ.gene.filter = T)

clonotyme_output <- VDJ_GEX_clonotyme(vdj.analyze.output = vdj_repertoire_tcells,
 gex.analyze.output = gex_acute, version="v2", exclude.clusters=7, highlight.genes="sell",
  colors = c("blue", "red", "black", "orange")
clonotyme_output[[4]]

# Version 3
    VGM <-
    readRDS("C:/Users/rapha/Downloads/TEMPLATE_VDJ_GEX_mat_Bcells_r2_150521.rds")

clonotyme_output <- VDJ_GEX_clonotyme(vdj.gex.matrix.output = VGM, version="v3",
 highlight.genes="sell", top.N.clonotypes = 1)


#---Method=velocyto----

#Dropest Alignment: Run on EULER CLUSTER
#env2lmod
#module load gcc/4.8.5 python/3.7.4
#module load gcc/4.8.5 dropest/0.8.6
#module load gcc/4.8.5 r/4.0.2
#module load gcc/4.8.5 hdf5/1.10.1
#module load gcc/4.8.5 openmpi/4.0.2
#module load gcc/4.8.5 r/4.0.2
#bsub -W 2880 -R 'rusage[mem=20000]'
/cluster/home/rakuhn/dropEst/dropest -V -C 6000 -f -g
/cluster/scratch/rakuhn/mm10-2020-A/refdata-gex-mm10-2020-A/genes/genes.gtf
-c /cluster/home/rakuhn/dropEst/configs/10x.xml
/cluster/scratch/rakuhn/cellranger_v5/g1/outs/possorted_genome_bam.bam

#Load required VDJ.analyze.output on EULER CLUSTER

vdj_repertoire_tcells
<- readRDS("/cluster/home/rakuhn/RPII/vdj_repertoire_tcells.rds")
vdj_repertoire_tcells
<- head(vdj_repertoire_tcells,2)
#Only select the first two repertoires since we only want to analyze these two.

# Load the two corresponding Dropest cell.count.matrices.rds

dropest.output.list <- list()
dropest.output.list[[1]]
```

```
<- readRDS("/cluster/home/rakuhn/RPII/old_bam/gex1/cell.counts.matrices.rds")
dropest.output.list[[2]]
<- readRDS("/cluster/home/rakuhn/RPII/old_bam/gex2/cell.counts.matrices.rds")

# Run Velocyto using Clonotyme

VDJ_GEX_clonotyme(method = "velocyto", version = "v2",
 vdj.analyze.output = vdj_repertoire_tcells,
 dropest.output.list = dropest.output.list,
 top.N.clonotypes = 3, exclude.clusters = 8, highlight.genes = "sell",
 velocyto.gex.merged = T, velocyto.out.dir = ".", velocyto.save.rds = F)


## End(Not run)
```

`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

```
## Not run:
 clonotype.clusters <- VDJ_GEX_clonotype_clusters_circos(vgm[[1]], n_cluster=8, topX = 20)
 # print circos plot:
 clonotype.clusters[[1]]

## End(Not run)
```

`VDJ_GEX_expansion` *Platypus V2 utility*

Description

only Platypus v2 Integrates VDJ and gene expression libraries by providing cluster membership seq_per_vdj object. Output will plot which transcriptional cluster (GEX) that the cells of a given clonotype are found in.

Usage

```
VDJ_GEX_expansion(
  GEX.list,
  VDJ.GEX.integrate.list,
  highlight.isotype,
  highlight.number
)
```

Arguments

`GEX.list`	The output of the automate_GEX function.
`VDJ.GEX.integrate.list`	Output from VDJ_GEX_integrate function. This object needs to have the GEX and VDJ information combined and integrated. This should be on the CLONAL level from the VDJ_GEX_integrate function.
`highlight.isotype`	(Optional) isotype to plot, choose between ["None","A","E","M","G","G1","G2A","G2B","G2C","G3"]. Default is None.
`highlight.number`	A vector corresponding to the rank of the clones that should be specified. Default is set to "20", which will present the cluster distribution for the top 20 clones.

Value

ggplot2 plot that breaks down clonotype membership per cluster for the specified input clones.

Examples

```
## Not run:
vdj.gex.expansion <- VDJ_GEX_expansion(GEX.list=GEX.list.output[[1]]
,VDJ.GEX.integrate.list=vdj.gex.integrate.output
,highlight.isotype = "None",highlight.number=1:20)

## End(Not run)
```

`VDJ_GEX_integrate` *only Platypus v2 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.*

Description

only Platypus v2 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.

Usage

```
VDJ_GEX_integrate(GEX.object, clonotype.list, VDJ.per.clone, clonotype.level)
```

Arguments

`GEX.object`	A single seurat object from automate_GEX function. This will likely be supplied as automate_GEX.output[[1]].
`clonotype.list`	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.
`VDJ.per.clone`	Output from the VDJ_per_clone function. Each element in the list should be found in the output from the automate_GEX function.
`clonotype.level`	Logical specifying whether the integration should occur on the cellular level (VDJ_per_clone) or on the clonotype level (e.g. output from VDJ_analyze or VDJ_clonotype). TRUE specifies that the clonotype level will be selected - e.g. the clonotype.list object will now contain information from the GEX object regarding clonal membership.

Value

Returns a nested list containing information corresponding to either the clonal level or the sequence level, depending on the input argument "clonotype.level". This function essentially will update the output of the analyze_VDJ or the VDJ_per_clone functions.

Examples

```
## Not run:
testing_integrate <- VDJ_GEX_integrate(GEX.object = automate.gex.output[[1]]
,clonotype.list =  VDJ.analyze.output
,VDJ.per.clone = VDJ.per.clone.output,clonotype.level = TRUE)

## End(Not run)
```

`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 == T) and metrics_summary.csv (Optional, will be appended to stats table if get.VDJ.stats == T)

`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 Platy-

pusDB_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 == F, 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

```
## Not run:

#FOR EXAMPLES see Platypus vignette at https://alexyermanos.github.io/Platypus/index.html

#Run from local directory input. For run from PlatypusDB input see
#PlatypusDB vignette
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))

# With Feature Barcodes
## Option 1: Cellranger multi or Cellranger count with --libraries output
VDJ.out.directory.list <- list()
VDJ.out.directory.list[[1]] <- "~/VDJ/S1/" #point to outs or per_sample_outs directory content
VDJ.out.directory.list[[2]] <- "~/VDJ/S2/"
GEX.out.directory.list <- list()
GEX.out.directory.list[[1]] <- "~/GEX/S1/"
GEX.out.directory.list[[2]] <- "~/GEX/S2/" #These directories contain two matrices (GEX and FB)
VGM <- VDJ_GEX_matrix(
VDJ.out.directory.list = VDJ.out.directory.list
,GEX.out.directory.list = GEX.out.directory.list,
FB.ratio.threshold = 2)

##Option 2: Separate input of FB data from separate Cellranger count run
VDJ.out.directory.list <- list()
VDJ.out.directory.list[[1]] <- "~/VDJ/S1/"
VDJ.out.directory.list[[2]] <- "~/VDJ/S2/"
GEX.out.directory.list <- list()
GEX.out.directory.list[[1]] <- "~/GEX/S1/"
GEX.out.directory.list[[2]] <- "~/GEX/S2/"
GEX.out.directory.list <- list()
FB.out.directory.list[[1]] <- "~FB/S1/"
FB.out.directory.list[[2]] <- "~FB/S1/"
VGM <- VDJ_GEX_matrix(
VDJ.out.directory.list = VDJ.out.directory.list,
GEX.out.directory.list = GEX.out.directory.list,
FB.out.directory.list = FB.out.directory.list,
FB.ratio.threshold = 2)

##Option 3: FB input for two datasets of which only one contains FB data
VDJ.out.directory.list <- list()
VDJ.out.directory.list[[1]] <- "~/study1/VDJ/S1/"
VDJ.out.directory.list[[2]] <- "~/study2/VDJ/S1/"
VDJ.out.directory.list[[3]] <- "~/study2/VDJ/S2/"
GEX.out.directory.list <- list()
GEX.out.directory.list[[1]] <- "~/study1/GEX/S1/"
GEX.out.directory.list[[2]] <- "~/study2/GEX/S1/"
GEX.out.directory.list[[2]] <- "~/study2/GEX/S2/"
GEX.out.directory.list <- list()
FB.out.directory.list[[1]] <- "PLACEHOLDER" #Study 1 does not contain FB data
FB.out.directory.list[[2]] <- "~/study2/FB/S1/"
FB.out.directory.list[[3]] <- "~/study2/FB/S2/"
VGM <- VDJ_GEX_matrix(
```

```
VDJ.out.directory.list = VDJ.out.directory.list,
GEX.out.directory.list = GEX.out.directory.list,
FB.out.directory.list = FB.out.directory.list,
FB.ratio.threshold = 2)


## End(Not run)
```

`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

```
#To return a single plot with top clones across samples
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)
```

```
#To return a facet plot with top clones for each sample
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)

#To return a facet plot and the legend separately with top clones for each group
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)

#To print both:
#overlay_clones_plot[[1]] #Plot
#gridExtra::grid.arrange(overlay_clones_plot[[2]]) #Legend
#To save, ggsave() is applicable to both

#To return a single plot with selected clones
#add a clonotype_to_plot column
#GEX@meta.data$clonotype_to_plot <- GEX$VJ_vgene == "TRAV5-1"
#Column with TRUE for all clones with a particular V gene
#overlay_clones_plot <- VDJ_GEX_overlay_clones(GEX = GEX, reduction = "umap"
#, clones.to.plot = "clonotype_to_plot", by.sample = TRUE, by.other.group = "none"
#, split.plot.and.legend = FALSE, pt.size = 1.5)
```

`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

```
## Not run:
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 = TRUE ,filename = "stats.csv")

## End(Not run)
```

VDJ_isotypes_per_clone

Platypus V2 clonal utility

Description

Only for Platypus v2 Clonal frequency plot displaying the isotype usage of each clone. ! For platypus v3 use VDJ_clonal_expansion

Usage

```
VDJ_isotypes_per_clone(
  VDJ_clonotype_output,
  VDJ_per_clone_output,
  clones,
  subtypes,
  species,
  sample.names,
  treat.incomplete.clones,
  treat.incomplete.cells,
  platypus.version,
  VDJ.matrix
)
```

Arguments

VDJ_clonotype_output	list of dataframes based on the VDJ_clonotype function output.
VDJ_per_clone_output	list of dataframes based on the VDJ_per_clone function output.
clones	numeric value indicating the number of clones to be displayed on the clonal expansion plot. Can take values between 1-50. Default value is 50.
subtypes	Logical indicating whether to display isotype subtypes or not.
species	Character indicating whether the samples are from mouse or human. Default is set to human. #' @param sample.names Character vector with the same length of the VDJ.GEX.matrix.out list. If a VDJ table is provided, length of samples names must be one. These names are used as references to the output and as title for the plots
sample.names	Vector. Names for samples in the order of the VDJ_GEX_matrix or the VDJ.analyze.output. Defaults to 1-n
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. This may result in a different frequency ranking of clonotypes than in the output of the VDJ_analyse function with filter.1HC.1LC = FALSE. "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 actual

contigs of there 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.

`platypus.version`
: Defaults to "v3". For a more flexible analysis in v3 use VDJ_clonal_expansion()

`VDJ.matrix`
: The VDJ table output of the VDJ_GEX_matrix function. (VDJ_GEX_matrix.output[[1]])

Value

returns a list containing plots with the percentages of isotypes for each clone on the cell level.

Examples

```
## Not run:
VDJ.isotype.per.clone <- VDJ_isotypes_per_clone(
VDJ_clonotype_output = VDJ.analyze.output
,VDJ_per_clone_output = VDJ.per.clone.output, clones = 30)

## End(Not run)
```

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,
  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.

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

```
## Not run:
#Calculate the 3-kmer frequency for CDRH3s and plot the 20 most abundant kmers.
 VDJ_kmers(VDJ = Platypus::small_vgm[[1]],
,sequence.columns = c("VDJ_cdr3s_aa"), grouping.column = "sample_id", kmer.k = 3, max.kmers = 20)

## End(Not run)
```

---

vdj_length_prob *vdj_length_prob A list dataframe specifying lengths and probabilities of bases deleted or inserted at each junction site of VDJ recombination event.*

Description

vdj_length_prob A list dataframe specifying lengths and probabilities of bases deleted or inserted at each junction site of VDJ recombination event.

Usage

```
data("vdj_length_prob")
```

Format

a dataframe:

v3_deletion length and probability of deleted bases at 3' end of V segment

d5_deletion length and probability of deleted bases at 5' end of D segment

d3_deletion length and probability of deleted bases at 3' end of D segment

j5_deletion length and probability of deleted bases at 5' end of J segment

dj_insertion length and probability of inserted bases between D-J segment

vj_insertion length and probability of inserted bases between V-J segment for light or alpha chains

---

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)
```

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"

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 ==F, 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

```
#Platypus v2
#network_out <- VDJ_network(VDJ = VDJ_analyze.out[[1]],per.sample = TRUE,distance.cutoff = 2)
#Platypus v3
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_GEX_matrix 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_cdr3s_aa","VJ_cdr3s_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

```
#PCA dimensionality reduction across samples for CDRH3
plot <- VDJ_ordination(VDJ = Platypus::small_vgm[[1]],
,feature.columns = c("VDJ_cdr3s_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

```
#To test the overlap of CDR3s between multiple samples
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_per_clone *VDJ_per_clone*

Description

only Platypus v2 Analyzes and processes the repertoire sequencing data from cellranger vdj. This provides information on the single-cell level for each clone, as opposed to the output from VDJ_analyze.

Usage

```
VDJ_per_clone(
  clonotype.list,
  VDJ.out.directory,
  contig.list,
  fasta.list,
  reference.list,
  filtered.contigs,
  annotations.json,
  JSON
)
```

Arguments

clonotype.list	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, the i'th entry in the directory supplied to GEX.list should correspond to the i'th element in the clonotype.list object.
VDJ.out.directory	Character vector with each element containing the path to the output of cellranger vdj runs. This corresponds to the same object used for the VDJ_analyze function. Multiple repertoires to be integrated in a single transcriptome should be supplied as multiple elements of the character vector. This can be left blank if supplying the clonotypes and contig files directly as input. 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).
contig.list	List of dataframe based on the all_contigs.csv file from cellranger vdj output. If 10x sequencing was not used then this object should be formatted with the same columns as the 10x object.
fasta.list	Contains the full-length sequence information in the same format as filtered_contig.fasta file from the output of cellranger.
reference.list	Contains the reference sequence information in the same format as concat_ref.fasta file from the output of cellranger.

filtered.contigs
: Logical indicating if the filtered contigs file should be used. TRUE will read VDJ information from only the filtered output of cellranger. FALSE will read the all contigs file from cellranger. Default set to TRUE (filtered output)

annotations.json
: Optional input from loaded all_contig_annotations.json. Will be read in automatically if not provided

JSON
: Boolean. Defaults to FALSE. Whether to load all_contig_annotations.json

Details

Platypus V2 data frame utility

Value

Returns a list of dataframes containing

Examples

```
## Not run:
VDJ_per_clone_out <- VDJ_per_clone(clonotype.list = output.from.VDJ_analyze
,VDJ.out.directory = "path/to/cellranger/outs/")

## End(Not run)
```

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

```
## Not run:
VDJ_phylogenetic_trees(VDJ=VDJ, sequence.type='VDJ.VJ',
trimmed=TRUE, as.nucleotide=TRUE, include.germline=TRUE,
additional.feature.columns=NULL, tree.level='intraclonal',
output.format='tree.df.list')

## End(Not run)
```

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

```
## Not run:
VDJ_phylogenetic_trees_plot(tree.dfs,color.by='clonotype_id', size.by='sequence_frequency')

## End(Not run)
```

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

```
#Simulating SHM data
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))

#Standard plots
SHM_plots <- VDJ_plot_SHM(VDJ = small_vgm[[1]]
```

```
, group.by = "sample_id", quantile.label = 0.9)

#Group by transcriptional cluster and label only top 1\%
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

```
## Not run:
#Rarefaction analysis of CDRH3 across samples
plot <- VDJ_diversity(VDJ = Platypus::small_vgm[[1]],
,feature.columns = c("VDJ_cdr3s_aa"), grouping.column = "sample_id")


## End(Not run)
```

VDJ_reclonotype_list_arrange

Platypus V2 dataframe utility

Description

Only Platypus v2 Organizes the top N genes that define each Seurat cluster and converts them into a single dataframe. This can be useful for obtaining insight into cluster-specific phenotypes.

Usage

```
VDJ_reclonotype_list_arrange(
  VDJ_clonotype.output,
  VDJ_analyze.output,
  Platypus_list.object
)
```

Arguments

`VDJ_clonotype.output`
: The output object from the VDJ_clonotype function. The column of the merged nucleotide clonotype IDs will be used to rearrange the new object.

`VDJ_analyze.output`
: The output from the initial VDJ_analyze, containing clonotype information based on nucleotide sequence.

`Platypus_list.object`
: The new list object from one of Platypus functions (for example, clonal lineages, VDJ_per_clne, etc) that should be merged based on the VDJ_clonotype output structure. nested list structure, where outer list corresponds to repertoire and the inner list corresponds to clones (on the nucleotide level).

Value

Returns a dataframe in which the top N genes defining each cluster based on differential expression are selected.

Examples

```
## Not run:
checking_vdj_reclono <- VDJ_reclonotype_list_arrange(
VDJ_clonotype.output = repertoire_reclonotype
,VDJ_analyze.output = repertoire_list
,Platypus_list.object = repertoire_vdj_per_clone)

## End(Not run)
```

`VDJ_select_clonotypes` *Select clonotypes*

Description

For prediction of antibody structures from a big data set it might be of interest to select the most expanded clonotypes for prediction. This function can select the top most expanded clonotypes based on the desired clone strategy. Among the most expanded clonotypes the cells are ranked according to the UMI count and then the top unique sequences are selected to use for prediction. The function's input is the Platypus VGM object. In order to integrate UMI counts to the data, the raw data which is the output of the PlatypusDB_fetch() function is needed in addition. From the selected clonotypes the germline reference sequences are obtained by calling MIXCR. This requires a local installation of MIXCR on your computer. !FOR WINDOWS USERS THE EXECUTABLE MIXCR.JAR HAS TO PRESENT IN THE CURRENT WORKING DIRECTORY !

The output of the VDJ_select_clonotypes function can directly be used for structure prediction by the AlphaFold_prediction() function.

Usage

```
VDJ_select_clonotypes(
  VGM,
  raw.data,
  clone.strategy,
  VDJ.VJ.1chain,
  donut.plot,
  clonotypes.per.sample,
  top.clonotypes,
  seq.per.clonotype,
  mixcr.directory,
  species,
  platypus.version,
  operating.system,
  simplify
)
```

Arguments

VGM	The platypus vgm object his used as an input for the function.
raw.data	In order to integrate the UMI counts per cell, the raw data has to be specified as a second input to the function which is the output of the PlatypusDB_fetch() function.
clone.strategy	The desired clone strategy can be specified as a string. 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. 10x.default is used as default. cdr3.aa will convert the default cell ranger clonotyping to amino acid bases. '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.
VDJ.VJ.1chain	If the VDJ.VJ.1chain argument is set to TRUE only cells with one VDJ and one VJ sequences are included in the selection.
donut.plot	If set to TRUE a donut plot for visualization of the clonotypes is returned.
clonotypes.per.sample	By default the top clonotypes are selected per sample. If the top clonotypes over all samples are desired the clonotypes.per.sample argument can be set to FALSE.
top.clonotypes	Specify the number of top clonotypes that will be selected either per sample if clonotypes.per.sample = T or over all if clonotypes.per.sample = F.
seq.per.clonotype	Specify the number of unique sequences per clonotype that are selected. The clonotypes are ordered according to UMI expression.

`mixcr.directory`
: The path to the directory containing an executable version of MIXCR.

`species`
: Either "mmu" for mouse or "hsa" for human. These use the default germline genes for both species contained in MIXCR. Default is set to "hsa".

`platypus.version`
: Character. Defaults to "v3". Can be "v2" or "v3" dependent on the input format

`operating.system`
: Can be either "Windows", "Darwin" (for MAC) or "Linux". If left empty this is detected automatically

`simplify`
: Only relevant when platypus.version = "v3". Boolean. Defaults to TRUE. If FALSE the full MIXCR output and computed SHM column is appended to the VDJ If TRUE only the framework and CDR3 region columns and computed SHM column is appended. To discriminate between VDJ and VJ chains, prefixes are added to all MIXCR output columns

Value

ADD DESCRIPTION OF RETURN VALUE HERE

Examples

```
## Not run:

ADD EXAMPLE CODE HERE


## End(Not run)
```

`VDJ_structure_analysis`
Analysis of antibody structures

Description

VDJ_structure_analysis is a Platypus function for the analysis of 3d structures of antibodies. The function is designed in a way to be a follow up of the alphafold_prediction function of the Platypus package. The input of this function is a list that has the VDJ MIXCR output in its first element and a list of structures in its second element. This is the output format of the alphafold_prediction function. The function can also be used to visualize structures from PDB files directly.

The function can visualize the structures of proteins and is especially designed for visualization of antibodies, where the framework and CDR regions are automatically annotated. If the antibody was predicted together with a antigen, the function can visualize binding interaction and detect the binding site residues. Furthermore, it can determine binding site metrics as average distance and model accuracy. The function has a variety of arguments to create the desired visualization.

Usage

```
VDJ_structure_analysis(
  VDJ.structure,
  cells.to.vis,
  rank,
  rank.list,
  overlay,
  PDB.file,
  spin.speed,
  color.frameworks,
  color.cdr3,
  color.cdr2,
  color.cdr1,
  VDJ.anno,
  label,
  label.size,
  bk.opac,
  font.opac,
  font.col,
  anno.seq,
  color.molecule,
  color.sheets,
  color.helix,
  angle.x,
  angle.y,
  angle.z,
  r3dmol.code,
  plddt.plot,
  structure.plot,
  antigen.interaction,
  binding.site.cutoff,
  dist.mat,
  SASA,
  hydrophobicity,
  charge,
  metrics.plot,
  BindingResidues.plot,
  binding.residue.barplot,
  binding.residue.barplot.style
)
```

Arguments

VDJ.structure	The VDJ.structure is a list object with the VDJ MIXCR out data frame as its first element containing the germline reference sequences. The second list element contains a list for every predicted structures with the top ranked predictions in pdb format read in by the bio3d:read_pdb() function.
cells.to.vis	In the cells.to.vis argument, the barcodes of the cells that should be analyzed can

be specified. It can be a single barcode or a list of barcodes. If all the elements of the VDJ.structure object shall be included, the cells.to.vis argument can be set to "ALL" (cells.to.vis = "ALL).

`rank` AlphaFold predicts multiple models for a given structure which are then ranked according to model's confidence with ranked_0.pdb as the best fit. By default the function uses the ranked_0 model but if a different rank is desired this can be specified in teh rank argument (rank = "ranked_5.pdb").

`rank.list` The function can also be used to analyze multiple ranked models per structure. This can be done by specifying the rank.list argument. rank.list = list("ranked_0.pdb", "ranked_1.pdb", "ranked_2.pdb") would use the top three most confident models per structure that are specified in the cells.to.vis argument.

`overlay` Multiple antibody structures are by default visualized separately. In case an overlay is desired this can be done by setting overlay to TRUE (overlay = T).

`PDB.file` Instead of visualizing the structure form the VDJ.structure object, the function can accept a path to a pdb.file as well.

`spin.speed` Protein animations can spin around an internal axis. By default the spin speed is set to 0. To start rotation the spin.speed argument can be set to the desired rotation speed.

`color.frameworks`
The color of the frameworks can be changed by specifying the color.frameworks argument with a the desired color in HEX format (color.frameworks = "#eb4034")

`color.cdr3` The color of the cdr3 can be changed by specifying the color.cdr3 argument with a the desired color in HEX format (color.cdr3 = "#eb4034")

`color.cdr2` The color of the cdr2 can be changed by specifying the color.cdr2 argument with a the desired color in HEX format (color.cdr2 = "#eb4034")

`color.cdr1` The color of the cdr1 can be changed by specifying the color.cdr1 argument with a the desired color in HEX format (color.cdr1 = "#eb4034")

`VDJ.anno` When using a VDJ structure object as an input, the regions of the antibody are by default annotated based on the MIXCR columns. If annotation is not desired the VDJ.anno argument can be set to FALSE.

`label` By default the annotated regions are labeled. The label can be disabled by setting the label argument to FASLE.

`label.size` The label size can be adjusted by specifying the label.size argument. By default the label size is set to 12.

`bk.opac` The label background opacity can be defined with the bk.opac argument. The default opacity is 0.8.

`font.opac` The opacity of the label's font can be set by specifying the font.opac argument. The default opacity is 1.

`font.col` The color of the font can be set by the font.col argument. It has to be in a HEX format.

`anno.seq` By the anno.seq argument any residues of the structure can be annotated. Every domain of the structure is handled as a separate chain, named alphabetically from A-Z, according to the order in the FASTA file. So for an anybody the HC is Chain A, the LC Chain B and the Antigen Chain C. The format of the anno.seq

argument is a vector with the index of the starting residue as its first element, the index of the end residue as second element, the chain in the third element and the color in HEX format as the fourth element. Optionally a label can be added by specifying the text in the fifth element anno.seq = c(4,12,"C","#9900ff","label"). For annotating multiple sequences at once, a list of vectors can be used anno.seq = list(c(4,12,"C","#9900ff","label1"),c(4,12,"B","#9350ff","label2"),...)

`color.molecule` The color of the non annotated residues can be set by the color.molecule argument in HEX format.

`color.sheets` The color of beta sheets can be set by specifying the color.sheets argument with the desired color in HEX format.

`color.helix` The color of alpha helices can be set by specifying the color.helix argument with the desired color in HEX format.

`angle.x` The molecule can be rotated around the x-axis by setting the angle.x argument in degree.

`angle.y` The molecule can be rotated around the y-axis by setting the angle.y argument in degree.

`angle.z` The molecule can be rotated around the z-axis by setting the angle.z argument in degree.

`r3dmol.code` Additional r3dmol code to modify appearance of the final structure. Defaults to "". Character input is run via eval(parse(x))

`plddt.plot` AlphaFold has a pLDDT confidence score for every residue of the structure. An additional plot of the structure colored according to the pLDDT score will be returned by setting the plddt.plot to TRUE.

`structure.plot` By default the structure is visualized. However, if the function is only used to get binding site metrics, the structure.plot argument can be set to FALSE. Like this nos structures are plotted.

`antigen.interaction`
If the antibody is predicted together with the antigen the antigen.interaction argument can be set to TRUE in order to get some binding site metrics. The function will determine the binding site residues based on the bio3d::binding.site() function as well as the average minimal distance of every binding site residue from the antibody to the antigen. Furthermore, the mean confidence (pLDDT) of the binding site residues is calculated. All the results are summarized in a data frame.

`binding.site.cutoff`
Cutoff for the bio3d::binding.site() function. Default is 5.

`dist.mat` If set to TRUE, the binding residues distance matrix for every structure will be returned as a list. Default = FALSE, so only the minimal distances in the summary data frame is returned.

`SASA` if set to TRUE the Solvent Accessible Surface Area will be calculated for every Structure

`hydrophobicity` If set to TRUE, the per residue hydrophobicity will be calculated.

`charge` If set to TRUE, the per residue charge will be calculated.

`metrics.plot` If set to TRUE, a structure plot colored according to the metrics will be shown.

`BindingResidues.plot`
: If the antigen.interaction is enabled, the binding site residues can be visualized on the structure by setting the BindingResidues.plot argument to TRUE.

`binding.residue.barplot`
: The binding.residue.bar plot can be set to TRUE to get a bar plot visualizing to which regions of the antibody the binding site residues belong too. A bar plot is produced for every structure separately as well as one summary bar plot over all analyzed structures.

`binding.residue.barplot.style`
: There are two styles available for the binding site residue bar plot. By default all regions are listed separately. By setting the binding.residue.barplot.style to "FR_CDR", only framework and CDR is distinguished.

Value

ADD DESCRIPTION OF RETURN VALUE HERE

Examples

```
## Not run:

ADD EXAMPLE CODE HERE


## End(Not run)
```

VDJ_tree *Platypus V2 phylogenetic trees.*

Description

Please refer tp VDJ_phylogenetic_tree for Platypus V3. Produces neighbor joining phylogenetic trees from the output of VDJ_clonal_lineages

Usage

```
VDJ_tree(
  clonal.lineages,
  with.germline,
  min.sequences,
  max.sequences,
  normalize.germline.length,
  unique.sequences
)
```

Arguments

`clonal.lineages`
: Output from VDJ_clonal_lineages. This should be nested list, with the outer list element corresponding to the individual repertoire and the inner list corresponding to individual clonal lineages based on the initial clonotyping strategy in the form of a dataframe with either Seq or Name.

`with.germline`
: Logical specifying if the germline should be set as outgroup. Default is set to TRUE.

`min.sequences`
: integer value specifying the minimum number of sequences to be allowed for clonal lineages. Default is 3.

`max.sequences`
: integer value specifying the maximum number of sequences to be allowed for clonal lineages. Default is 500

`normalize.germline.length`
: Logical determining whether or not the branch length separating the germline from the first internal node should be normalized. Potentially useful for visualization if the remainder tips are far from the root. Default is TRUE.

`unique.sequences`
: Logical indicating if those cells containing identical VDJRegion sequences should be merged into single nodes and have their variant added as the tip label. Default is TRUE.

Value

Returns a nested list of phylogenetic trees. The output[[i]][[j]] corresponds to the j'th clone in the i'th input repertoire. plot(output[[i]][[j]]) should display the phylogenetic tree if the ape package is loaded.

Examples

```
## Not run:
vdj.tree <- VDJ_tree(clonal.lineages = VDJ.clonal.lineage.output
,with.germline=TRUE,min.sequences = 5
,max.sequences = 30,unique.sequences = TRUE)

## End(Not run)
```

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_GEX_matrix() function (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.
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

```
## Not run:
VDJ_Vgene_usage_barplot(VDJ = Platypus::small_vgm[[1]],
HC.gene.number = 2, platypus.version = "v3")

## End(Not run)
```

`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

```
#Platypus v3
example.vdj.vgene_usage <- VDJ_Vgene_usage_stacked_barplot(
VDJ = Platypus::small_vgm[[1]], 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_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.
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

```
## Not run:
 usage_circos_VDJVJ <- VDJ_VJ_usage_circos(vgm[[1]])

 # print plot:
 usage_circos_VDJVJ[[1]]

## End(Not run)
```

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

```
#Add info to whole VGM object
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

```
#For Platypus version 3

# 1. If only vdj data (vgm[[1]]) and
#strict feature barcode assignment is used:
vgm_expanded_fb <- VGM_expand_featurebarcodes(
vgm = small_vgm[[1]],
by.majority.barcodes = FALSE,
integrate.in.gex=FALSE, vdj.only= TRUE)

# 2. If whole vgm and strict fb assignment is used
#(gex and vdj - necessary if gene expression analysis
# of sub-samples is desired):
vgm_expanded_fb <- VGM_expand_featurebarcodes(
vgm = small_vgm,
by.majority.barcodes = FALSE,
integrate.in.gex=TRUE, vdj.only= FALSE)

# 3. If whole vgm and majority barcode assignement is used
#(gex and vdj) - necessary if gene expression analysis
#of sub-samples is desired):
vgm_expanded_fb <- VGM_expand_featurebarcodes(vgm = small_vgm,
```

```
by.majority.barcodes = TRUE,
integrate.in.gex=TRUE, vdj.only= FALSE)

#Note: Majority barcode assignment is recommended
#if the assumption that all cells within one clonotype
#originate from the same sample sub-group is feasible.
```

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

```
#Adding a new clonotyping method to VDJ
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
#and update clonotype_id and clonotype_frequency column
#(as does VDJ_clonotype_v3 in VDJ)

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

#Pull genes from GEX and add as metadata column to VDJ

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

Index