Package: blaseRtools (via r-universe)

June 14, 2024

Title R Tools for Blaser Lab Data Analysis

Version 0.0.0.9163

Description This is a repository of R tools for Single Cell RNA seq and other lab data analysis functions.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

URL https://github.com/blaserlab/blaseRtools

BugReports https://github.com/blaserlab/blaseRtools/issues

Imports CellChat (>= 1.4.0), BiocManager, knitr, org.Dr.eg.db, org.Hs.eg.db, org.Mm.eg.db, rmarkdown, DESeq2, genefilter, annotate, geneplotter, ashr, readxl, cowplot, monocle3, MASS, Matrix, Seurat, SeuratDisk, SingleCellExperiment, broom, ggpubr, ggrepel, grid, hexbin, pheatmap, rrvgo, topGO, scater, huxtable, stats, dplyr, ggplot2, stringr, Signac, methods, circlize, GenomicRanges, BiocGenerics, Biostrings, GenomeInfoDb, IRanges, digest, fs, ggforce, GenomicAlignments, Rsamtools, rstatix, batchelor, leidenbase, DoubletFinder, ComplexHeatmap, blaseRtemplates, prompt, SeuratObject, SummarizedExperiment, cli, future, R.utils, MatrixGenerics, forcats, ggrastr, grr, purrr, readr, remotes, rlang, tibble, tidyr, tidyselect, BSgenome, BSgenome.Drerio.UCSC.danRer11, BSgenome.Hsapiens.UCSC.hg38, TxDb.Hsapiens.UCSC.hg38.knownGene, TxDb.Drerio.UCSC.danRer11.ensGene, AnnotationDbi, GenomicFeatures, lubridate, edgeR, ggtext, speedglm, hdf5r, assertthat, plyr, igraph, reshape2, fields, KernSmooth

Depends tidyverse, blaseRdata, R (>= 2.10)

VignetteBuilder knitr

Remotes github::blaserlab/blaseRdata, github::mojaveazure/seurat-disk, github::blaserlab/DoubletFinder, github::blaserlab/blaseRtemplates, github::sqjin/CellChat

Repository https://blaserlab.r-universe.dev

RemoteUrl https://github.com/blaserlab/blaseRtools

RemoteRef HEAD

RemoteSha 4c6d9427f0ff4af883e9838f158bde64148d7e44

Contents

`SeuratWrappers-package`
blaseRtools: R Tools for Blaser Lab Data Analysis

Description

This is a repository of R tools for Single Cell RNA seq and other lab data analysis functions.

Author(s)

Maintainer: Brad Blaser <bradley.blaser@osumc.edu> (ORCID)

See Also

Useful links:

- https://github.com/blaserlab/blaseRtools
- Report bugs at https://github.com/blaserlab/blaseRtools/issues

add_cds_factor_columns *Add predefined sample-level cell metadata to cell data sets while importing*

Description

Add predefined sample-level cell metadata to cell data sets while importing

Usage

```
add_cds_factor_columns(cds, columns_to_add)
```

Arguments

cds	A cell data set object
columns_to_add	A named vector where the name of each element becomes the name of the new colData column and the value is the value for that particular sample. Best used when importing from a metadata table.

aggregate.Matrix *Compute summary statistics of a Matrix*

Description

Similar to aggregate. Splits the matrix into groups as specified by groupings, which can be one or more variables. Aggregation function will be applied to all columns in data, or as specified in formula. Warning: groupings will be made dense if it is sparse, though data will not.

Usage

```
## S3 method for class 'Matrix'
aggregate(x, groupings = NULL, form = NULL, fun = "sum", ...)
```

Arguments

x	a Matrix or matrix-like object
groupings	an object coercible to a group of factors defining the groups
form	formula
fun	character string specifying the name of aggregation function to be applied to all columns in data. Currently "sum", "count", and "mean" are supported.
...	arguments to be passed to or from methods. Currently ignored

Details

`aggregate.Matrix` uses its own implementations of functions and should be passed a string in the `fun` argument.

Value

A sparse `Matrix`. The rownames correspond to the values of the groupings or the interactions of groupings joined by a _.

There is an attribute `crosswalk` that includes the groupings as a data frame. This is necessary because it is not possible to include character or data frame groupings in a sparse Matrix. If needed, one can `cbind(attr(x,"crosswalk"),x)` to combine the groupings and the aggregates.

See Also

summarise

summarise

aggregate

Ape-class *An S4 class to hold genebank/APE file data.*

Description

An instance of this class is best created by calling "bb_parseape()" on a genebank or APE-formatted file. That function will parse the file, correctly format the sections and place them in the slots of the Ape Object. Technically only "LOCUS" is a required slot for the Ape object, however there is no point without having "ORIGIN" (sequence data), and so bb_parseape() will fail without an "ORIGIN" section. Other slots are optional. Additional slots will be ignored by the constructor function. DNA sequence will be stored in a DNAStringSet object and features in a GRanges object. See https://www.ncbi.nlm.nih.gov/Sitemap/samplerecord.html for genebank file specification.

Slots

`LOCUS` The LOCUS line of the genebank formatted as a character string.

`DEFINITION` The DEFINITION line of the genebank file formatted as a character string.

`ACCESSION` The ACCESSION section of the genebank file formatted as a character string.

`VERSION` The VERSION section of the genebank file formatted as a character string.

`SOURCE` The SOURCE section of the genebank file formatted as a character string.

`COMMENT` The COMMENT section of the genebank file formatted as a character string.

`FEATURES` The FEATURES section of the genebank file formatted as a character string. Created internally from the GRanges object. Caution: some FEATURE attributes may be lost in conversion.

`ORIGIN` The DNA sequence

`end_of_file` The end of the file signal.

`dna_biostring` The entire ORIGIN sequence formatted as a DNAStringSet of length 1.

`granges` Genebank features formatted as a GRanges object.

`Ape.DNA` *Get the DNASringSet Slot from an Ape Object*

Description

Get the DNASringSet Slot from an Ape Object

Usage

```
Ape.DNA(ape)
```

`Ape.fasta` *Save an Ape Instance as a Fasta File*

Description

Save an Ape Instance as a Fasta File

Usage

```
Ape.fasta(ape, feature = NULL, out)
```

Arguments

`feature`	Name of feature to select when writing FASTA file. If null (default), the whole biostring will be saved as a fasta.
`out`	Name of FASTA file to write

Ape.fimo — *Run FIMO on Selected Ape Object Features*

Description

For the supplied Ape object, run FIMO to identify putative transcription factor binding sites in a DNA subsequence.

Usage

```
Ape.fimo(ape, fimo_feature, out = NULL)
```

Arguments

ape	An Ape instance
fimo_feature	A character vector of features from the Ape object that will be used to run fimo.
out	Directory that will be created to hold the fimo results. A date/time stamp will be appended. If null, the objects will not be saved and the function will only return a GRanges object

Ape.granges — *Get the GRanges Slot from an Ape Object*

Description

Get the GRanges Slot from an Ape Object

Usage

```
Ape.granges(ape)
```

Ape.save — *Save an Ape Instance as a Genebank Format File*

Description

Save an Ape Instance as a Genebank Format File

Usage

```
Ape.save(ape, out)
```

Arguments

out	Name of genebank/APE file to write

Ape.setFeatures *Set the FEATURES Slot of a GRanges Object*

Description

Set the FEATURES Slot of a GRanges Object

Usage

```
Ape.setFeatures(ape, gr)
```

Arguments

ape	An ape object
gr	A GRanges object. This object will become the new FEATURES and granges slots for the Ape object. So if you want to keep the old features, the new features need to be appended using c(old_gr, new_gr) as the value for the gr argument.

as.cell_data_set *Convert objects to Monocle3* cell_data_set *objects*

Description

Convert objects to Monocle3 cell_data_set objects

Usage

```
as.cell_data_set(x, ...)

## S3 method for class 'Seurat'
as.cell_data_set(
  x,
  assay = DefaultAssay(object = x),
  reductions = AssociatedDimReducs(object = x, assay = assay),
  default.reduction = DefaultDimReduc(object = x, assay = assay),
  graph = paste0(assay, "_snn"),
  group.by = NULL,
  ...
)
```

Arguments

x	An object
...	Arguments passed to other methods
assay	Assays to convert
reductions	A vector of dimensional reductions add to the cell_data_set object; defaults to all dimensional reductions calculated from assay and all global dimensional reductions
default.reduction	Name of dimensional reduction to use for clustering name
graph	Name of graph to be used for clustering results
group.by	Name of cell-level metadata column to use as identites; pass

Details

The Seurat method utilizes as.SingleCellExperiment to transfer over expression and cell-level metadata. The following additional information is also transferred over:

- Cell emebeddings are transferred over to the reducedDims slot. Dimensional reduction names are converted to upper-case (eg. "umap" to "UMAP") to match Monocle 3 style
- Feature loadings are transfered to cds@reduce_dim_aux$gene_loadings if present. **NOTE**: only the feature loadings of the last dimensional reduction are transferred over
- Standard deviations are added to cds@reduce_dim_aux$prop_var_expl if present. **NOTE**: only the standard deviations of the last dimensional reduction are transferred over
- Clustering information is transferred over in the following manner: if cell-level metadata entries "monocle3_clusters" and "monocle3_partitions" exist, then these will be set as the clusters and partitions, with no nearest neighbor graph being added to the object; otherwise, Seurat's nearest-neighbor graph will be converted to an igraph object and added to the cell_data_set object along with Seurat's clusters. No partition information is added when using Seurat's clsuters

Value

A cell_data_set object

See Also

as.SingleCellExperiment

`as.Seurat.cell_data_set` *Convert single cell experiment to Seurat*

Description

Convert single cell experiment to Seurat

Usage

```
## S3 method for class 'cell_data_set'
as.Seurat(
  x,
  counts = "counts",
  data = NULL,
  assay = "RNA",
  project = "cell_data_set",
  loadings = NULL,
  clusters = NULL,
  ...
)
```

Arguments

`loadings`	Name of dimensional reduction to save loadings to, if present; defaults to first dimensional reduction present (eg. `SingleCellExperiment::reducedDimNames(x)[1]`); pass `NA` to suppress transfer of loadings
`clusters`	Name of clustering method to use for setting identity classes

Details

The `cell_data_set` method for `as.Seurat` utilizes the `SingleCellExperiment` method of `as.Seurat` to handle moving over expression data, cell embeddings, and cell-level metadata. The following additional information will also be transfered over:

- Feature loadings from `cds@reduce_dim_aux$gene_loadings` will be added to the dimensional reduction specified by `loadings` or the name of the first dimensional reduction that contains "pca" (case-insensitive) if `loadings` is not set
- Monocle 3 clustering will be set as the default identity class. In addition, the Monocle 3 clustering will be added to cell-level metadata as “monocle3_clusters”, if present
- Monocle 3 partitions will be added to cell-level metadata as “monocle3_partitions”, if present
- Monocle 3 pseudotime calculations will be added to “monocle3_pseudotime”, if present
- The nearest-neighbor graph, if present, will be converted to a `Graph` object, and stored as “`assay`_monocle3_graph”

See Also

as.Seurat.SingleCellExperiment

---

bb_aggregate *Aggregate Single Cell Gene Expression*

Description

Generates a matrix of counts aggregated by gene and/or cell group.

Usage

```
bb_aggregate(
  obj,
  assay = "RNA",
  experiment_type = "Gene Expression",
  gene_group_df = NULL,
  cell_group_df = NULL,
  norm_method = c("log", "binary", "size_only"),
  pseudocount = 1,
  scale_agg_values = TRUE,
  max_agg_value = 3,
  min_agg_value = -3,
  binary_min = 0,
  exclude.na = TRUE
)
```

Arguments

obj	A Seurat or cell data set object
assay	Gene expression assay to use for aggregation; currently only applies to Seurat objects, Default: 'RNA'
gene_group_df	A 2-column dataframe with gene names or ids and gene groupings, Default: NULL
cell_group_df	A 2-coumn dataframe with cell ids and gene groupings, Default: NULL
norm_method	Gene normalization method, Default: c("log", "binary", "size_only")
pseudocount	Pseudocount, Default: 1
scale_agg_values	Whether to scale the aggregated values, Default: TRUE
max_agg_value	If scaling, make this the maximum aggregated value, Default: 3
min_agg_value	If scaling, make this the minimum aggregated value, Default: -3
binary_min	Minimum value below which a cell is considered not to express a feature, Default: 0
exclude.na	Exclude NA?, Default: TRUE

Details

The best way to group genes or cells is by using bb_*meta and then select cell_id or feature_id plus one metadata column with your group labels.

Value

A dense or sparse matrix.

See Also

`cli_div`, `cli_alert` `normalized_counts`, `my.aggregate.Matrix` `character(0)`

`bb_align` *Align a CDS object according to a metadata variable*

Description

Align a CDS object according to a metadata variable

Usage

```
bb_align(cds, align_by, n_cores = 8)
```

Arguments

`cds`	A cell data set object to align
`align_by`	A metadata column to align by
`n_cores`	Number of cores to reduce dimensions by.

Value

A modified cell data set object with aligned dimensions and new metadata columns holding pre-alignment umap coordinates.

bb_annotate_npc *Annotate a Plot using NPC Coordinates*

Description

Annotate a Plot using NPC Coordinates

Usage

```
bb_annotate_npc(label, x, y, ...)
```

Arguments

label	the text label to apply to the plot
x	NPC X coordinate
y	NPC Y coordinate

bb_blind_images *Make a Copy of Image Files and Rename With File Hashes in Blinded Folder*

Description

Will copy and rename the files and generate two files: "blinding_key.csv" with the original and blinded file names, and "scoresheet.csv" with just the blinded filenames. Add columns as needed to scoresheet, for example, runx_count. Then run bb_unblind to rejoin scoresheet to the key and generate an unblinded result file.

Usage

```
bb_blind_images(analysis_file, file_column, output_dir)
```

Arguments

analysis_file	The analysis file for the experiment. It should contain 1 line for every biological sample and should have a filename for every file to be blinded.
file_column	The column name in the analysis_file with the files to be blinded.
output_dir	The linux-style file path for the directory that will hold the blinded images. The directory will be created by the function.

Value

nothing

bb_buff_granges *Clean Up Your GRanges Object*

Description

This function does several things. It removes ranges with non-standard chromosomes and drops their levels. It will optionally set the genome to the user-provided value. Typically we would use "hg38" or "danRer11". This is the exported version because it is so useful.

Usage

```
bb_buff_granges(x, gen)
```

Arguments

x	A Granges object to buff.
gen	An optional genome name to provide. Recommend "hg38" or "danRer11".

Value

A GRanges object

bb_cds_anno *Annotate a CDS by Label Transfer*

Description

Use this function to transfer cell labels from one single cell dataset to another. If a cds is provided for either reference or query, it is converted to a seurat object and the labels are transferred to the query by anchor finding. The assignments will be in the form of a new cds column with name predicted."column name from reference". This should be unique on the first application of the function to a query dataset. However, if running queries against more than 1 reference data set it is possible that you will unintentionally generate the same column name which would overwrite the first assignment column. The function checks for this and aborts with the recommendation to supply a unique id to the unique_id parameter.

Usage

```
bb_cds_anno(query_cds, ref, transfer_col, unique_id = NULL)
```

Arguments

query_cds	The single cell data set you wish to annotate. Must be a CDS.
ref	The reference single cell data set. May be either a CDS or Seurat object.
transfer_col	The column from the reference data set that provides the labels.
unique_id	A unique identifier to add to the column with the transferred labels. Default is NULL but it is recommended to provide an informative label when annotating against more than one reference. Default is NULL.

Value

a CDS with two new cell metadata columns

See Also

cli_abort components, SCTransform, RunPCA, RunUMAP, FindTransferAnchors, MapQuery exprs character(0) reexports select, mutate-joins as_tibble

---

bb_cds_heatmap *Make a Heatmap of Aggregated Gene Expressionf from a CDS*

Description

Supply a cds subset and aggregation values and get a heatmap. Plot the return value using cowplot::plot_grid(). This function wraps several complicated functions from ComplexHeatmap and tries to apply default values for the most common use case: plotting top markers from a cds with cells grouped arbitrarily (usually by cluster of some type). This function provides the option to aggregate by genes as well in order to plot gene modules. For cell and gene aggregation, provide a name (in the form of a string) of the corresponding metadata column and the aggregation will be performed. There are many aesthetic parameters for this function and even more available for the internal ComplexHeatmap::Heatmap() function. The best way to adjust parameters not provided here is to scroll the popup box that appears when you type ComplexHeatmap::Heatmap() in RStudio and pass them in via the ellipsis. The default is to put genes in columns and cells in rows. You can flip this behavior by setting flip_axis = TRUE. This will require adjustment of some aesthetic parameters. Complex annotations (beyond labeling cherry-picked genes) are not currently supported.

Usage

```
bb_cds_heatmap(
  cds_subset,
  cellmeta_col = NULL,
  rowmeta_col = NULL,
  heatmap_highlights = NULL,
  three_colors = c("blue4", "ivory", "red3"),
  flip_axis = FALSE,
  name = NULL,
```

```
  heatmap_legend_param = list(title_gp = gpar(fontface = "plain", fontsize = 9),
    grid_width = unit(0.14, "in"), labels_gp = gpar(fontsize = 8)),
  row_dend_width = unit(5, "mm"),
  column_dend_height = unit(5, "mm"),
  column_dend_side = "bottom",
  show_row_names = T,
  row_names_gp = gpar(fontsize = 9),
  show_column_names = F,
  row_dend_gp = gpar(lwd = 0.5),
  column_dend_gp = gpar(lwd = 0.5),
  row_title = NULL,
  column_title = NULL,
  padding = 1.5,
  labels_rot = 45,
  ...
)
```

Arguments

cds_subset	The subset of cells and genes you want to plot as a heatmap. Best approach is to pipe the cds through filter_cds() and into this function.
cellmeta_col	The name of a cell metadata column to aggregate cells by; one of cellmeta_col and rowmeta_col must not be NULL, Default: NULL
rowmeta_col	The name of a row metadata column to aggregate cells by; one of cellmeta_col and rowmeta_col must not be NULL, Default: NULL
heatmap_highlights	A vector of gene names to highlight using anno_mark(), Default: NULL
three_colors	A vector of colors for the main color scale, Default: c("blue4", "ivory", "red3")
flip_axis	Logical; whether to plot genes as rows (TRUE) or columns (FALSE), Default: FALSE
name	Name of the main color scale, Default: NULL
heatmap_legend_param	Graphical parameters for the main heatmap legend, Default: list(title_gp = gpar(fontface = "plain", fontsize = 9), grid_width = unit(0.14, "in"), labels_gp = gpar(fontsize = 8))
row_dend_width	Row dendrogram width, Default: unit(5, "mm")
column_dend_height	Column dendrogram height, Default: unit(5, "mm")
column_dend_side	Side on which to plot the column dendrogram, Default: 'bottom'
show_row_names	Logical; whether or not to show rownames, Default: T
row_names_gp	Graphical parameters for the row names, Default: gpar(fontsize = 9)
show_column_names	Logical; whether or not to show column names, Default: F
row_dend_gp	Graphical parameters for the row dendrogram, Default: gpar(lwd = 0.5)

column_dend_gp	Graphical parameters for teh column dendrogram, Default: gpar(lwd = 0.5)
row_title	Row title text, Default: NULL
column_title	Column title text, Default: NULL
padding	Padding between gene names on the heatmap highlights, Default: 1.5
labels_rot	Rotation of the heatmap highlight labels, Default: 45
...	Optional arguments to pass to ComplexHeatmap::Heatmap()

Value

A complex heatmap in the form of a gtree.

---

bb_cellchat *Infer Cell-Cell Communication*

Description

Use this function to identify ligand/receptor pairs expressed by cell clusters in human, mouse or zebrafish single cell data. A CellChat object is generated which can be used to visualize these connections using bb_cellchat_heatmap or other tools from package CellChat.

Usage

```
bb_cellchat(
  cds,
  group_var,
  n_cores = 12,
  species = c("human", "mouse", "zebrafish"),
  min_cells = 10,
  prob_type = c("triMean", "truncatedMean", "median"),
  prob_trim = NULL,
  project = TRUE,
  pop_size_arg = TRUE,
  ask = TRUE
)
```

Arguments

cds	The cell data set object. It should usually be pre-filtered to conatin a single biological sample.
group_var	The cell metadata column identifying cell groups for cell-cell communication inference.
n_cores	Number of cores for the analysis, Default: 12
species	Species for the assay, Default: c("human", "mouse", "zebrafish")
min_cells	Cell clusters smaller than this value will be ignored., Default: 10

prob_type	Methods for computing the average gene expression per cell group. By default = "triMean", producing fewer but stronger interactions; When setting 'type = "truncatedMean"', a value should be assigned to 'trim', producing more interactions, Default: c("triMean", "truncatedMean", "median")
prob_trim	the fraction (0 to 0.25) of observations to be trimmed from each end of x before the mean is computed if using truncatedMean, Default: NULL
project	Whether or not to smooth gene expression, Default: TRUE
pop_size_arg	Whether consider the proportion of cells in each group across all sequenced cells. Set population.size = FALSE if analyzing sorting-enriched single cells, to remove the potential artifact of population size. Set population.size = TRUE if analyzing unsorted single-cell transcriptomes, with the reason that abundant cell populations tend to send collectively stronger signals than the rare cell populations., Default: TRUE

Details

see github::sqjin/CellChat

Value

A CellChat object

See Also

normalized_counts mutate-joins,pull tibble createCellChat,CellChatDB.human,CellChatDB.mouse,character(0
character(0) plan

---

bb_cellchat_heatmap *Make a Complex Heatmap From a CellChat Object With Sensible Defaults*

Description

This will generate heatmap from a CellChat object using ComplexHeatmap::Heatmap. Options are provided to filter for sender and receiver cells, to generate simple marginal annotations and for aesthetic control.

Usage

```
bb_cellchat_heatmap(
  object,
  source_filter = NULL,
  target_filter = NULL,
  interaction_filter = NULL,
  interaction_threshold = 0,
  colors = c("transparent", "red3"),
  rowanno = c(NULL, "Annotation", "Pathway"),
```

```
  rowanno_colors = NULL,
  colanno = c(NULL, "Source", "Target"),
  colanno_colors = NULL,
  pval_filter = 0.05,
  heatmap_name = "Interaction\nScore",
  heatmap_show_row_dend = TRUE,
  heatmap_row_dend_width = unit(5, "mm"),
  heatmap_show_column_dend = TRUE,
  heatmap_column_dend_height = unit(5, "mm"),
  heatmap_row_names_gp = gpar(fontsize = 10),
  heatmap_column_names_gp = gpar(fontsize = 10),
  heatmap_column_names_rot = 90,
  heatmap_column_title = NULL,
  heatmap_column_title_gp = gpar(fontsize = 12, fontface = "bold"),
  col_anno_name_gp = gpar(fontsize = 10, fontface = "bold"),
  row_anno_name_gp = gpar(fontsize = 10, fontface = "bold"),
  return_value = c("heatmap", "plot", "matrix")
)
```

Arguments

object	The CellChat object to plot
source_filter	Optional filter for source cell clusters from the object metadata. Accepts a single string or vector of cell groups., Default: NULL
target_filter	Optional filter for target cell clusters from the object metadata. Accepts a single string or vector of cell groups., Default: NULL
interaction_filter	Optional filter to include only certain interactions in the figure.
interaction_threshold	Optional filter to only include interactions above a certain threshold.
colors	Color scale endpoints, Default: c("transparent", "red3")
rowanno	Options for simple row annotation; must be one of c(NULL, "Annotation", "Pathway")
rowanno_colors	Optional colors to replace the poor color selections from Complex heatmap. Must be supplied as a named list with one element each for "Annotation" and "Pathway". Not required if not showing these annotations. The list should be of the form: list(Annotation = c("name1" = "color value1", "name2" = "color_value2")), Default: NULL
colanno	Options for simple column annotation; must be one of c(NULL, "Source", "Target")
colanno_colors	See rowanno_colors, Default: NULL
pval_filter	Filter for significance of associations. CellChat returns pvalues of 0, 0.01, and 0.05; this function will filter and retain values less than or equal to the provided value. Default: 0.05
heatmap_name	Name for the main color scale of the heatmap, Default: 'InteractionScore'

`heatmap_show_row_dend`
: Show row dendrograms? Default: TRUE

`heatmap_row_dend_width`
: Width of row dendrograms Default: unit(5, "mm")

`heatmap_show_column_dend`
: Show column dendrograms?' Default: TRUE

`heatmap_column_dend_height`
: Height of column dendrograms. Default: unit(5, "mm")

`heatmap_row_names_gp`
: Row name graphical params, Default: gpar(fontsize = 10)

`heatmap_column_names_gp`
: Column name graphical params, Default: gpar(fontsize = 10)

`heatmap_column_names_rot`
: Column name rotation, Default: 90

`heatmap_column_title`
: Column title, Default: NULL

`heatmap_column_title_gp`
: Column title graphical params, Default: gpar(fontsize = 12, fontface = "bold")

`col_anno_name_gp`
: Column annotation name graphical params, Default: gpar(fonmtsize = 10, fontface = "bold")

`row_anno_name_gp`
: Row annotation name graphical params, Default: gpar(fontsize = 10, fontface = "bold")

`return_value`
: Return a heatmap plot or a matrix.

Details

see github::sqjin/CellChat

Value

a heatmap as a grid object; plot using cowplot::plot_grid

See Also

`subsetCommunication` `as_tibble`,`c("tibble", "tibble")`,`rownames` `filter`,`mutate`,`select`,`mutate-joins`,`group_by`,`s`
`pivot_wider` `rowAnnotation`,`columnAnnotation`,`draw-dispatch`,`Heatmap` `colorRamp2` `grid.grab`

bb_cellmeta *Get Cell Metadata*

Description

Take a cell_data_set object or a Seurat object and return the cell metadata in the form of a tibble. The unique cell identifier column is labeled cell_id by default. Prior versions of this function would only accept a cell_data_set. The input argument has been changed from cds to obj to reflect the fact that Seurat objects are now also accepted.

Usage

```
bb_cellmeta(obj, row_name = "cell_id", cds = NULL)
```

Arguments

obj	A cell_data_set or Seurat object.
row_name	Optional name to provide for cell unique identifier, Default: 'cell_id'
cds	Provided for compatibility with prior versions, Default: NULL

Details

If a value is supplied for cds, a warning will be issued and the function will pass the value of cds to obj.

Value

A tibble

bb_cite_umap *Plot a UMAP Showing Cite-Seq Antibody Binding*

Description

Requires a cds with an alt experiment established. Use bb_split_citeseq to generate this and to normalize binding data using the CLR method. Returns a ggplot.

Usage

```
bb_cite_umap(
  cds,
  antibody,
  assay = "CLR_counts",
  cell_size = 1,
  alpha = 1,
```

```
  alt_dim_x = NULL,
  alt_dim_y = NULL,
  plot_title = NULL,
  color_legend_title = NULL,
  order = TRUE,
  rescale = NULL,
  ncol = NULL
)
```

Arguments

cds	The cds with an "Antibody Capture" alt experiment to plot.
antibody	The name of the antibody to plot. Equivalent to gene_short_name. Accepts a character vector.
assay	The binding assay to use, Default: "CLR_counts"
cell_size	Size of points to plot, Default: 1
alpha	Alpha for the plotted points, Default: 1
alt_dim_x	Alternate/reference dimensions to plot by.
alt_dim_y	Alternate/reference dimensions to plot by.
plot_title	Optional title for the plot, Default: NULL
color_legend_title	Optional title for the color scale., Default: NULL
order	Whether or not to order cells by gene expression. When ordered, non-expressing cells are plotted first, i.e. on the bottom. Default: TRUE
rescale	Optional redefinition of the color scale, Default: NULL
ncol	If specified, the number of columns for facet_wrap, Default: NULL

Value

a ggplot

See Also

reducedDims

---

bb_cluster_representation

Cluster Representation By Fisher Exact Test Per Cell

Description

Use this function to determine the differential representation of cells in clusters. It will determine fold change in a single experimental class over a single control or reference class. This value is normalized to the number of cells captured in all clusters from the class. Significance is determined using Fisher's exact test. This test may overestimate significance in large data sets. In this case, bb_cluster_representation2 may be more robust.

Usage

```
bb_cluster_representation(
  cds,
  cluster_var,
  class_var,
  experimental_class,
  control_class,
  pseudocount = 1,
  return_value = c("table", "plot")
)
```

Arguments

cds	A cell data set object
cluster_var	The CDS cell metadata column holding cluster data. There can be any number of clusters in this column.
class_var	The CDS cell metadata column holding sample class data. There can be only 2 classes in this column. You may need to subset or reclass the samples to achieve this.
experimental_class	The value from the class column indicating the experimental group.
control_class	The value from the class column indicating the control or reference class.
return_value	Option to return either a plot or a data table for plotting in a separate step. Must be either "plot" or "table".

Value

A ggplot or a table of data for plotting

---

bb_cluster_representation2

Cluster Representation By Regression Per Sample

Description

Use this function to determine the differential representation of cells in clusters. It uses a regression method to determine fold change between groups of biological samples. It can only compare two sample groups, e.g. control vs experimental at this point. See parameter descriptions for how to identify these properly.

Usage

```
bb_cluster_representation2(
  obj,
  sample_var,
  cluster_var,
  comparison_var,
  comparison_levels = NULL,
  color_pal = c("red3", "blue4"),
  sig_val = c("FDR", "PValue"),
  return_val = c("plot", "data")
)
```

Arguments

obj	The (possibly filtered) single cell object to operate on. Can be either Seurat or monocle/CDS object.
sample_var	The metadata column holding the biological sample information.
cluster_var	The metadata column holding the clustering or other cell classification information.
comparison_var	The metadata column holding the comparison group information. There can be only two levels in this column. Character data will be converted to factors.
comparison_levels	A character vector identifying the order of the levels to compare. The first value will be shown with negative log2Fold Change and the second will be positive. If NULL (default), R will pick for you.
color_pal	Color palette for the comparison levels, Default: c("red3", "blue4")
sig_val	Report PValue or FDR, Default: "FDR"
return_val	Value to return, Default: c("plot", "table)

Details

DETAILS

Value

OUTPUT_DESCRIPTION

Source

http://bioconductor.org/books/3.13/OSCA.multisample/differential-abundance.html

bb_doubletfinder *Use doubletfinder to model and mark doublets*

Description

Use doubletfinder to model and mark doublets

Usage

```
bb_doubletfinder(cds, doublet_prediction, qc_table, ncores = 1)
```

Arguments

cds	A cell data set object
doublet_prediction	Predicted proportion of doublets fom 0 to 1
qc_table	A table of qc calls from the blaseRtools qc function

Value

A tibble of low- and high-confidence doublet calls by barcode

bb_extract_msig *Extract MSIGDB Gene Sets*

Description

Use this to extract gene sets from the MSIGDB. Most gene sets are known by "STANDARD_NAME". You can filter the gene set list by supplying a named filter list to the bb_extract_msig function. The name of each list element should be one of the metadata column names and the list element contents should be the values to filter for. Filtering works in an additive way, meaning if you supply a filter list with two elements it will extract gene sets passing filters 1 AND 2.

Usage

```
bb_extract_msig(
  filter_list = NULL,
  return_form = c("id_list", "name_list", "tibble")
)
```

Arguments

filter_list	A named list to filter the MSIGDB by. Defaults to NULL which will return the whole MSIGDB
return_form	Select from a list of gene ids or a list of gene names by gene set. This is a useful format for the fgsea package. Alternatively "tibble" can be select and all filtered gene sets will be bound into a long-form (tidy) tibble.

Value

Gene set as a list or tibble.

`bb_fix_file_path` *Convert windows filepath to linux-compatible.*

Description

This will only work if the backslashes have already been escaped to double backslashes. For example, readr automatically does this when reading in windows file paths from csv files. If you are using as a standalone function, run this in the terminal: scan(what = "character, n = 1), paste the unmodified filepath at the blank line, and copy the modified file path as the argument, x.

Usage

```
bb_fix_file_path(x)
```

Arguments

`x`	A character string filepath copied from Windows with escaped backslashes.

Value

A linux-compatible filepath

`bb_fragment_replacement`
Replace The Path to a Fragment File In a Signac Object

Description

Often you will wish to package a Signac object or share the object to someone who does not have access to the same directories as you. This is a problem because one of the internal sub-objects, the Fragment object, holds a file path to a directory containing an atac fragments file and its .tbi intex. Often this file will only be accessbile to you. This function allows you to replace the fragments file path that is held internally within the Signac object. Just copy the files to a shared location and provide that file location to the function.

For Signac objects with multiple internal Fragments objects, provide vector of file paths.

For packaged fragments files, use system.file or fs::path_package to access this, usually from extdata.

Usage

```
bb_fragment_replacement(obj, new_paths)
```

Arguments

obj	A signac object.
new_paths	A character vector of new file paths. You must have the same number of new file paths as Fragments sub-objects within the signac object.

Value

A modified Signac object.

See Also

c("Cells.Fragment", "CountFragments", "CreateFragmentObject", "FilterCells", "Fragment-class", "Fragments", "Fragments", "SplitFragments", "UpdatePath", "ValidateCells", "ValidateFragments", "ValidateHash", "head.Fragment", "subset.Fragment"), CreateFragmentObject cli_abort file_access map2

---

bb_genebubbles *Create a Gene Bubble/Dot Plot*

Description

This is a very data-dense plot and is the recommended way for showing expression of single markers/genes by cell group. By default, this function will return an unfaceted ggplot with cell groups on the X axis and genes on the Y axis with dot size representing proportion of cells in the cell group expressing a gene and color scale representing per-cell expression.

But it also may be of interest to add aesthetic variables such as facets or additional color scales. There are two ways this function will facilitate that. First, you can supply a vector of cell groups to the cell_grouping argument and a the cells will be grouped by the composite value of these factors. Usually if you are doing this, you also will want to have access to the components of this composite variable to facet by. So you can supply "data" to the return_value argument to get a tibble. From there you can modify as necessary and generate a ggplot assigning aesthetics and scales as desired and using geom_point.

This function also supports visualizing citeseq data. These data should be allocated to an alternative experiment in the cds object. To show these data, set experiment_type to "Antibody Capture" or the name of the alternate experiment with citeseq data. The genes parameter should be the name assigned to the antibody derived tag. Expression threshold is particularly useful in this case because of the background binding observed with antibodies. The default is 0 and so by default any cell with more than 0 counts will be considered an expressor of that marker. This threshold is applied before scaling across markers. The best way to set this threshold is to visualize your markers of interest and isotypes with expression_threshold = 0 and scale_expr = FALSE. Then pick a threshold value based on the color scale and rerun with scale_expr either TRUE or FALSE.

Usage

```
bb_genebubbles(
  obj,
  genes,
  cell_grouping,
  experiment_type = "Gene Expression",
  scale_expr = TRUE,
  expression_threshold = 0,
  gene_ordering = c("bicluster", "as_supplied"),
  group_ordering = c("bicluster", "as_supplied"),
  return_value = c("plot", "data")
)
```

Arguments

obj	A Seurat or cell_data_set object.
genes	Gene or genes to plot.
cell_grouping	Cell metadata column to group cells by. Supply more than one in a vector to generate a composite variable.
experiment_type	Experiment data to plot. Usually will be either "Gene Expression" or "Antibody Capture", Default: 'Gene Expression'
scale_expr	Whether to scale expression by gene, Default: TRUE
expression_threshold	Pre-scaling expression value below which a cell is considered not to express a marker. This value is fed to the binary_min parameter of bb_aggregate, Default = 0
gene_ordering	By default, genes will be ordered by a clustering algorithm. Supply "as_supplied" to plot the genes in the order supplied to the "genes" argument , Default: c("bicluster", "as_supplied")
group_ordering	By default, cell groups will be ordered by a clustering algorithm. Supply "as_supplied" to plot the cell groups in the order supplied to "cell_grouping", Default: c("bicluster", "as_supplied")
return_value	Whether to return a plot or data in tibble form, Default: c("plot", "data")

Value

A ggplot or a tibble

bb_gene_dotplot *Make a dotplot of gene expression by cell population*

Description

Make a dotplot of gene expression by cell population

Usage

```
bb_gene_dotplot(
  cds,
  markers,
  group_cells_by,
  reduction_method = "UMAP",
  norm_method = c("size_log", "log_only"),
  scale_expression_by_gene = FALSE,
  lower_threshold = 0,
  max.size = 10,
  group_ordering = "bicluster",
  gene_ordering = NULL,
  pseudocount = 1,
  scale_max = 3,
  scale_min = -3,
  colorscale_name = NULL,
  sizescale_name = NULL,
  ...
)
```

Arguments

cds	A cell data set object
markers	A character vector of genes to plot
group_cells_by	A cds colData column. Use "multifactorial" to pick 2 categorical variables to put on X axis and to facet by. See ordering below.
norm_method	How to normalize gene expression. Size_factor and log normalized or only log normalized.
scale_expression_by_gene	Whether to scale expression values according to gene. Defaults to FALSE.
lower_threshold	Lower cutoff for gene expression
max.size	The maximum size of the dotplot
group_ordering	Defaults to "biclustering" method from pheatmap. Optionally will take a vector of group values to set the axis order explicitly. If using group_cells_by = "multifactorial" you will need a df to define facet and axis levels. See example.
gene_ordering	Optional vector of gene names to order the plot.

pseudocount	Add to zero expressors. Default = 1
scale_max	Expression scale max
scale_min	Expression scale min
colorscale_name	Label for the color scale
sizescale_name	Label for the size scale
...	Additional parameters to pass to facet_wrap.

Value

A ggplot

`bb_gene_modules` *A function to generate gene modules and add them to the Gene Metadata*

Description

Based on Monocle3's gene module functions. Implemented with default values. Will convert a Seurat object to a cell data set using SeuratWrappers and then calculate modules. The function returns an object of the same type.

Usage

```
bb_gene_modules(obj, n_cores = 8, cds = NULL)
```

Arguments

obj	A single cell object of type Seurat or cell_data_set.
n_cores	Number of processor cores to use for the analysis, Default: 8
cds	Provided for backward compatibility. If a value is supplied, it will return a warning and transfer to the obj argument., Default: NULL

Details

see https://cole-trapnell-lab.github.io/monocle3/docs/differential/#gene-modules

Value

An object of the same type: Seurat or cell_data_set

See Also

graph_test, find_gene_modules rename, mutate-joins, mutate, select fct_shift

bb_gene_pseudotime *Plot expression of a gene or genes in pseudotime.*

Description

Plot expression of a gene or genes in pseudotime.

Usage

```
bb_gene_pseudotime(
  cds_subset,
  min_expr = NULL,
  cell_size = 0.75,
  nrow = NULL,
  ncol = 1,
  panel_order = NULL,
  color_cells_by = "pseudotime",
  trend_formula = "~ splines::ns(pseudotime, df=3)",
  label_by_short_name = TRUE,
  vertical_jitter = NULL,
  horizontal_jitter = NULL
)
```

Arguments

cds_subset	A cell data set object subset with only cells and genes of interest
min_expr	Lower threshold of expression for plotting
cell_size	Size of point for plotting
nrow	Number of rows for facetting
ncol	Number of columns for facetting
panel_order	Character string for order of genes to plot
color_cells_by	A cds colData column
trend_formula	Formula for the trend line
label_by_short_name	Boolean to label by gene name or ID
vertical_jitter	Adjustment to vertical jitter. Optional
horizontal_jitter	Adjustment to horizontal jitter. Optional

Value

A ggplot

bb_gene_umap *Make a Plot of Gene Expression in UMAP Form*

Description

Takes in a Seurat or cell_dat_set object, extracts UMAP dimensions and gene expression values. For Seurat, default assay is "RNA"; can be changed to if necessary. For cell_data_set, the assay parameter does nothing; the function extracts log and size-factor normalized counts which are similar but not identical to the Seurat "RNA" assay. If a vector of genes is supplied to gene_or_genes, a faceted plot will be generated. If a dataframe is supplied, an aggregated plot will be generated with a facet for each gene group. The dataframe must be of 2 colums: the first containing feature ids and the second containing grouping information. This is best generated using bb_rowmeta.

Usage

```
bb_gene_umap(
  obj,
  gene_or_genes,
  assay = "RNA",
  order = TRUE,
  cell_size = 1,
  alpha = 1,
  ncol = NULL,
  plot_title = NULL,
  color_legend_title = "Expression",
  max_expr_val = NULL,
  alt_dim_x = NULL,
  alt_dim_y = NULL,
  rasterize = FALSE,
  raster_dpi = 300,
  cds = NULL
)
```

Arguments

obj	A Seurat or cell_data_set object.
gene_or_genes	Individual gene or genes or aggregated genes to plot. Supply a character string for a single gene, a vector for multiple genes or a dataframe for aggregated genes. See description.
assay	For Seurat objects only: the gene expression assay to get expression data from, Default: 'RNA'
order	Whether or not to order cells by gene expression. When ordered, non-expressing cells are plotted first, i.e. on the bottom. Caution: when many cells are overplotted it may lead to a misleading presentation. Generally bb_genebubbles is a better way to present, Default: TRUE
cell_size	Size of the points, Default: 1

alpha	Transparency of the points, Default: 1
ncol	Specify the number of columns if faceting, Default: NULL
plot_title	Optional title for the plot, Default: NULL
color_legend_title	Option to change the color scale title, Default: 'Expression'
max_expr_val	Maximum expression value to cap the color scale, Default: NULL
alt_dim_x	Alternate/reference dimensions to plot by.
alt_dim_y	Alternate/reference dimensions to plot by.
rasterize	Whether to render the graphical layer as a raster image. Default is FALSE.
raster_dpi	If rasterize then this is the DPI used. Default = 300.
cds	Provided for backward compatibility. If a value is supplied a warning will be emitted., Default: NULL

Value

A ggplot

See Also

normalized_counts as_tibble pivot_longer mutate-joins, mutate, select, arrange ggplot, aes, geom_point, scale_colour_viridis_d, labs, facet_wrap, vars, theme, margin

---

bb_gene_violinplot *Make a plot of gene expression in UMAP form*

Description

Make a plot of gene expression in UMAP form

Usage

```
bb_gene_violinplot(
  cds,
  variable,
  genes_to_plot,
  experiment_type = "Gene Expression",
  pseudocount = 1,
  include_jitter = FALSE,
  ytitle = "Expression",
  plot_title = NULL,
  rows = 1,
  show_x_label = TRUE,
  legend_pos = "none",
  comparison_list = NULL,
  palette = NULL,
```

```
  violin_alpha = 1,
  jitter_alpha = 1,
  jitter_color = "black",
  jitter_fill = "transparent",
  jitter_size = 0.5,
  facet_scales = "fixed",
  order_genes = TRUE,
  jitter_match = FALSE,
  rasterize = FALSE,
  raster_dpi = 300
)
```

Arguments

cds	A cell data set object
variable	Stratification variable for x-axis
genes_to_plot	Either a character vector of gene short names or a tbl/df where the first column is gene short name and the second is the gene grouping.
pseudocount	Value to add to zero-cells
include_jitter	Include jitter points
ytitle	Title for y axis
plot_title	Main title for the plot
rows	Number of rows for facetting
show_x_label	Option to show x label
legend_pos	Position for label
comparison_list	Optional list of comparisons for ggpubr
palette	Color palette to use. Viridis is default.
violin_alpha	Alpha value for violin plot
jitter_alpha	Alpha value for jitter plot
jitter_color	Color for the jitter plot. Defaults to black and ignored if jitter_match == TRUE
jitter_fill	Fill for the jitter plot
jitter_size	Size of the jitter points
facet_scales	Scale option for facetting. "Fixed" is default
order_genes	If true, put genes in the same order as variable parameter
jitter_match	If true, match jitter color to violin fill.
rasterize	Whether to render the graphical layer as a raster image. Default is FALSE.
raster_dpi	If rasterize then this is the DPI used. Default = 300.

Value

A ggplot

bb_goenrichment *Go Term Enrichment*

Description

A function to find enriched go terms from a query list of gene names relative to a reference list of gene names.

Usage

```
bb_goenrichment(
  query,
  reference,
  group_pval = 0.01,
  go_db = c("org.Hs.eg.db", "org.Dr.eg.db", "org.Mm.eg.db")
)
```

Arguments

query	A vector of gene names
reference	The background gene list. Usually will be as_tibble(rowData(cds_main)).
group_pval	P value to determine enrichment. Default: 0.01.
go_db	GO term database Default: c("org.Hs.eg.db", "org.Dr.eg.db", "org.Mm.eg.db")

Value

A list of items including the enrichment results.

bb_goscatter *Make a scatter plot of GO term associations*

Description

Make a scatter plot of GO term associations

Usage

```
bb_goscatter(
  simMatrix,
  reducedTerms,
  size = "score",
  addLabel = TRUE,
  labelSize = 4
)
```

Arguments

simMatrix	Take from output of bb_gosummary
reducedTerms	Also take from output of bb_gosummary
size	Variable to map to point size. Defaults to "score".
addLabel	Boolean; whether or not to add text labels
labelSize	Optional label size

Value

A ggplot

---

bb_gosummary *A function to reduce go terms by semantic similarity*

Description

A function to reduce go terms by semantic similarity

Usage

```
bb_gosummary(
  x,
  reduce_threshold = 0.8,
  go_db = c("org.Hs.eg.db", "org.Dr.eg.db", "org.Mm.eg.db")
)
```

Arguments

x	A list go term enrichment results produced by bb_goenrichment.
reduce_threshold	The degree of term reduction. 0 to 1. Higher is more reduction.
go_db	The database to query. Choose from c("org.Hs.eg.db", "org.Dr.eg.db", "org.Mm.eg.db", ...).

Value

A list of items for downstream plotting

bb_load_tenx_h5 *Load a 10X Genomics H5 File and Return a CDS*

Description

This function reads a 10X Genomics H5 file and returns a cell_data_set or CDS. Important notes: This is tested and should work for all single-genome and citeseq data sets. For multigenome data, as long as the features are all contained in the same matrix and identified by a composite reference/gene identifier, it should also work. In this case, the CDS will have to be filtered post-hoc using the sample_barcodes.csv data to get the appropriate species of cell. Also: this function takes in a specific H5 file from a unique biological sample. So it should be wrapped in another function to map across all the samples in a dataset. The wrapper needs to find the appropriate H5 file, e.g. filtered_feature_bc_matrix.h5 for files processed with cellranger count or sample_filtered_feature_bc_matrix.h5 for files processed using cellranger multi. This may change based on the cellranger version used.

Usage

```
bb_load_tenx_h5(filename, sample_metadata_tbl = NULL)
```

Arguments

filename	Path to the h5 file.

Value

A cell data set.

bb_load_tenx_targz *Load 10X Data Into CDS*

Description

Load 10X Data Into CDS

Usage

```
bb_load_tenx_targz(targz_file, umi_cutoff = 100, sample_metadata_tbl = NULL)
```

Arguments

targz_file	A character string of the file path to the multi pipestance directory
umi_cutoff	Don't import cells with fewer UMIs than this value. Defaults to 100.
sample_metadata_tbl	A tibble in wide format with one line. Col names indicate metadata variables to add.

Value

A cell data set object.

`bb_makeTrace` *Construct a Trace object from a Signac Object or Granges object*

Description

Problem 1: Signac objects and GRanges made from bigwigs are large and it is computationally expensive to get data from them when tweaking plots. Problem 2: For the most part we like how Signac plots genomic coverage of track-like data and we would like to show bulk data in a similar way with a similar compuatational interface. The trace object is a small intermediate object that holds the minimal amount of data you need to make a coverage plot showing accessibility or binding to a specific genomic region. Then you can use the trace object to quickly and easily generate tracks as needed for your plot. These tracks are all ggplots and are easy to configure for legible graphics. There are options for displaying groups by color or facet which are built in with good graphical defaults. If these are not suitable, they can be changed post hoc like any other ggplot.

Usage

```
bb_makeTrace(
  obj,
  gene_to_plot,
  genome = c("hg38", "danRer11"),
  extend_left = 0,
  extend_right = 0,
  peaks = NULL,
  bulk_group_col = NULL,
  bulk_group_id = "bulk",
  bulk_coverage_col = "score"
)
```

Arguments

`obj`	A Signac/Seurat object or a GRanges object. Import a bigwig file to a GRanges object using import.bw from rtracklayer. Use plyranges functions to easily pre-filter the GRanges objec e.g. by chr to reduce processing time. The precise range will be defined by gene_to_plot and the extend arguments. You may wish to add grouping metadata columns and to merge several bulk tracks before importing.
`gene_to_plot`	The gene you want to display. Must be a valid gene in the genome assembly being used.
`genome`	The genome assembly. Required. Must be either "hg38" or "danRer11".
`extend_left`	Bases to extend plot_range left, or upstream relative to the top strand.
`extend_right`	Bases to extend plot_range right, or downstream relative to the top strand.

peaks An optional GRanges object holding peak data. Ignored for Signac/Seurat objects which carry this internally.

bulk_group_col This parameter allows identification of a grouping variable in your bigwig/Granges object if you have added one. One example is if you are adding multiple bulk tracks: You can use this to plot the data in different colors or facets. A new metadata column called "group" will be added to hold this. If you haven't added grouping data in upstream processing, leave it NULL (default). In this case the new group column will be filled with a single string (see bulk_group_id). Will be ignored for Signac/Seurat objects.

bulk_group_id The value you want to display with the track data on faceted plots when you haven't added any grouping information before making the object. Defaults to "bulk" which is not a great choice. Will be ignored for Signac/Seurat objects.

bulk_coverage_col
If you are making the object from a bigwig/GRanges, you need to identify which column in your bigwig/GRanges object holds the coverage data to plot on the y axis. Defaults to "score". Will be ignored for Signac/Seurat objects.

---

bb_make_ape_genomic *Make an Ape Genome Object*

Description

This function takes either a gene name or sequence coordinates and returns an ape object with DNA sequence from the the selected genome reference. You can choose from hg38 and GRCz11. Features are generated from ensembl GFF files. Features overlapping the query range (gene or sequence) are returned as a GRanges object and as features within the Ape object. The features included can optionally be filtered using the "include_type" argument to the function. Using the "additional_granges" argument, you can provide additional features not present in the standard gene model which will be added to the Ape object GRanges slot and to the features slot.

Usage

```
bb_make_ape_genomic(
  query,
  genome = c("hg38", "GRCz11"),
  extend_left = 0,
  extend_right = 0,
 include_type = c("ncRNA_gene", "rRNA", "exon", "pseudogene", "pseudogenic_transcript",
  "ncRNA", "gene", "CDS", "lnc_RNA", "mRNA", "three_prime_UTR", "five_prime_UTR",
    "unconfirmed_transcript", "scRNA", "C_gene_segment", "D_gene_segment",
    "J_gene_segment", "V_gene_segment", "miRNA", "tRNA", "snRNA", "snoRNA",
    "lincRNA_gene", "lncRNA_gene", "unconfirmed_transcript"),
  additional_granges = NULL
)
```

Arguments

query	Either a valid gene name or a named numeric vector of genome coordinates. This vector should be of the form: c(chr = 1, start = 1000, end = 2000). The vector must be numeric an must have those names. The chromosome number will be converted to "chr1" etc internally.
genome	The genome to pull from, Default: c("hg38", "GRCz11")
extend_left	Number of bases to extend the query to the left or "upstream" relative to the + strand.
extend_right	Number of bases to extend the query to the right or "downstream" relative to the + strand.
include_type	The type of features to include from the standard gene model. Default: c("ncRNA_gene", "rRNA", "exon", "pseudogene", "pseudogenic_transcript", "ncRNA", "gene", "CDS", "lnc_RNA", "mRNA", "three_prime_UTR", "five_prime_UTR", "unconfirmed_transcript", "scRNA", "C_gene_segment", "D_gene_segment", "J_gene_segment", "V_gene_segment", "miRNA", "tRNA", "snRNA", "snoRNA", "lincRNA_gene", "lncRNA_gene", "unconfirmed_transcript")
additional_granges	A GRanges object with features to add to the Ape Object. Coordinates should all be relative to the reference, *NOT* the sequence extracted for the ape file. The Granges object can be constructed with the following syntax: GenomicRanges::makeGRangesFromDataFrame(data.frame(seqname = "chr6", start = 40523370, end = 40523380, strand = "+", type = "addl_feature", gene_name = "prkcda", label = "feature1"), keep.extra.columns = T). The gene_name argument here is optional. If you have defined features based on the extracted sequence, (i.e. relative to position 1 in the ORIGIN section of the Ape object), the best option is to use the feature setting function FEATURES(instance_of_Ape) <- GRanges_Object.

Value

An APE object

---

bb_make_ape_transcript

Make an Ape Transcriptome Object

Description

This function takes a specific ensembl transcript identifier, such as ENST00000348343.11, and gets the cDNA sequence from the corresponding transcriptomic reference. This is returned as an Ape object with the UTR's and the CDS annotated as features.

Usage

```
bb_make_ape_transcript(query, transcriptome = c("hg38", "GRCz11"))
```

Arguments

query	A specific ensembl transcript identifier.
transcriptome	Genome/transcriptome reference to use, Default: c("hg38", "GRCz11")

Details

DETAILS

Value

OUTPUT_DESCRIPTION

See Also

TxDb.Hsapiens.UCSC.hg38.knownGene BSgenome.Hsapiens.UCSC.hg38 org.Hs.eg.db character(0)
BSgenome.Drerio.UCSC.danRer11 org.Dr.eg.db cli_abort matchPattern

Examples

```
## Not run:
if(interactive()){
 #EXAMPLE1
 }

## End(Not run)
```

---

bb_merge_narrowpeaks *A Function To Merge Replicate Narrow Peaks GRanges*

Description

@description This function merges replicate peaks GRanges objects into 1. @param peaks_gr A regular list of GRanges objects (Not a GRangesList). @return A GRanges object @export @import IRanges GenomicRanges

Usage

```
bb_merge_narrowpeaks(peaks_gr)
```

bb_metafeature *A Function to Generate Data For Making MetaPlots*

Description

Use this function to generate data for making TSS enrichment plots or other metafeature plots that are centered on a single genomic locus. This function returns the data you need for the plot. Use the tibble element that is returned to plot the enrichment plot and the matrix for the heatmap. The problem currently is that the binwidths for the enrichment plot need to be smaller than the binwidths for the heatmap to look good. If you use good binwidths for the enrichment plot, the heatmap will crash. So either reduce the size of the heatmap matrix before plotting that or rerun the function with a different bin size. This function allows sample names to be added, so several samples can be column-bound together for comparison. Each gene is normalized to its own outer flanks so this should account for differences in sequencing depth to some degree. You also have the option to include all possible TSS in the plot (i.e. including zeros) which you may want to do if comparing several samples. To do this, set select_hits to FALSE.

Usage

```
bb_metafeature(
  query,
  targets,
  select_hits = TRUE,
  width = 2000,
  binwidth = 10,
  sample_id = NULL
)
```

Arguments

query	A GRanges object. This should be from a bam file so you can plot read coverage across the metagene.
targets	A GRanges object. The targets you want to plot around.
select_hits	Do you want to plot only the targets that have overlappign query reads? Defaults to true.
width	The width of the analysis in bp.
binwidth	The binwidth in bp. Width must be evenly divided by binwidth.
sample_id	An optional sample id if you want to join this matrix up with another one.

Value

A list including a matrix and a tibble.

bb_monocle_regression *A function to perform regression on single cell data.*

Description

Note: this function is deprecated in favor of bb_monocle_regression_better which returns log2FoldChange instead of estimates. The log2FoldChange value is the normalized_effect value returned from the monocle regression functions.

Usage

```
bb_monocle_regression(
  cds,
  gene_or_genes,
  stratification_variable = NULL,
  stratification_value = NULL,
  form,
  linking_function = "negbinomial"
)
```

Arguments

cds	A cell data set object.
gene_or_genes	Genes to regress by.
stratification_variable	Optional colData column to subset the cds by internal to the function.
stratification_value	Optional value to stratify by.
form	The regression formula in the form of "~var1+var2+..."
linking_function	For the generalized linear model.

Value

A tibble containing the regression results.

bb_monocle_regression_better
An Improved Function to Perform Regression on Single Cell Data.

Description

This function replaces bb_monocle_regression. It returns log2FoldChange instead of estimates. The log2FoldChange value is the normalized_effect value returned from the monocle regression functions.

Usage

```
bb_monocle_regression_better(
  cds,
  gene_or_genes,
  stratification_variable = NULL,
  stratification_value = NULL,
  form,
  linking_function = "negbinomial"
)
```

Arguments

cds	A cell data set object.
gene_or_genes	Genes to regress by.
stratification_variable	Optional colData column to subset the cds by internal to the function.
stratification_value	Optional value to stratify by.
form	The regression formula in the form of "~var1+var2+..."
linking_function	For the generalized linear model.

Value

A tibble containing the regression results.

---

bb_parseape *Parse a Genebank File and Construct an APE Object*

Description

This is the main function for reading file in genebank/ape/equivalent format and generating an instance of the Ape class. String manipulations are used to parse the input ape file. Biostrings and GRanges functions are called to generate DNAStringSet and GRanges objects to store sequence and feature data, respectively. The Ape constructor function is called internally at the end.

Usage

```
bb_parseape(input_file)
```

Arguments

input_file	The genebank/ape file to parse and construct into an instance of the Ape class.

Value

An Ape object

bb_plotfootprint *Plot motif footprinting results*

Description

Plot motif footprinting results

Usage

```
bb_plotfootprint(
  object,
  features,
  alt_main_title = NULL,
  alt_color_title = NULL,
  legend_pos = "right",
  colorscale = NULL,
  assay = NULL,
  group.by = NULL,
  idents = NULL,
  label = TRUE,
  repel = TRUE,
  show.expected = TRUE,
  normalization = "subtract",
  label.top = 3,
  label.idents = NULL,
  fontsize = 14,
  linesize = 0.2
)
```

Arguments

object	A Seurat object
features	A vector of features to plot
alt_main_title	Alternative title for the main plot. Accepts markdown.
alt_color_title	Alternative title for the color scale
legend_pos	Position to place the legend
colorscale	Named vector of colors to apply to the top plot.
assay	Name of assay to use
group.by	A grouping variable
idents	Set of identities to include in the plot
label	TRUE/FALSE value to control whether groups are labeled.
repel	Repel labels from each other
show.expected	Plot the expected Tn5 integration frequency below the main footprint plot

normalization	Method to normalize for Tn5 DNA sequence bias. Options are "subtract", "divide", or NULL to perform no bias correction.
label.top	Number of groups to label based on highest accessibility in motif flanking region.
label.idents	Vector of identities to label. If supplied,
fontsize	Theme font size
linesize	Size to draw the footprint lines label.top will be ignored.

---

bb_plot_genes_in_pseudotime

Plots expression for one or more genes as a function of pseudotime

---

Description

Plots expression for one or more genes as a function of pseudotime

Usage

```
bb_plot_genes_in_pseudotime(
  cds,
  gene_or_genes,
  pseudotime_dim,
  min_expr = NULL,
  cell_size = 0.75,
  nrow = NULL,
  ncol = 1,
  panel_order = NULL,
  color_cells_by = pseudotime_dim,
  trend_formula_df = 3,
  label_by_short_name = TRUE,
  vertical_jitter = NULL,
  horizontal_jitter = NULL,
  legend_title = NULL
)
```

Arguments

cds	Cell data set to plot.
gene_or_genes	Gene or genes for which to plot pseudotime.
pseudotime_dim	The column holding the pseudotime dimension to plot along.
min_expr	the minimum (untransformed) expression level to plot.
cell_size	the size (in points) of each cell used in the plot.
nrow	the number of rows used when laying out the panels for each gene's expression.

ncol — the number of columns used when laying out the panels for each gene's expression

panel_order — vector of gene names indicating the order in which genes should be laid out (left-to-right, top-to-bottom). If label_by_short_name = TRUE, use gene_short_name values, otherwise use feature IDs.

color_cells_by — the cell attribute (e.g. the column of colData(cds)) to be used to color each cell. Defaults to the value provided for pseudotime_dim.

trend_formula_df — degrees of freedom for the model formula used to fit the expression trend over pseudotime. The formulat takes the form of "~ splines::ns(pseudotime_dim, df = trend_formula_df)".

label_by_short_name — label figure panels by gene_short_name (TRUE) or feature ID (FALSE).

vertical_jitter — A value passed to ggplot to jitter the points in the vertical dimension. Prevents overplotting, and is particularly helpful for rounded transcript count data.

horizontal_jitter — A value passed to ggplot to jitter the points in the horizontal dimension. Prevents overplotting, and is particularly helpful for rounded transcript count data.

Value

a ggplot2 plot object

---

bb_plot_rowData_col

A helper function to generate a data frame in the proper form for aggregate expression plotting with bb_gene_umap.

Description

Use as argument for the "gene_or_genes" parameter for bb_gene_umap.

Usage

```
bb_plot_rowData_col(cds, rowData_col, filter_in = NULL, filter_out = NULL)
```

Arguments

cds — CDS from which to extract the gene metadata. Should be the same cds as the enclosing function.

rowData_col — Gene metadata column to aggregate by.

filter_in — Subset of values to focus on. Each will become a facet in the final plot. Default is to keep everything except NA values.

filter_out — Option to filter out any unwanted values. Default is to not filter out anything.

Value

A data frame in the format needed to pass into bb_gene_umap.

`bb_plot_trace_axis` *Plot The X Axis From A Trace Object*

Description

Generates the X axis for stacking other track plots on top of.

Usage

```
bb_plot_trace_axis(trace, xtitle = NULL)
```

Arguments

`trace`	A Trace object.
`xtitle`	An optional title for the X axis. Defaults to the genome and chromosome.

`bb_plot_trace_data` *Plot The Trace Data From A Trace Object*

Description

A function to generate a line plot from tracklike genomic data. This pulls data from the Trace object trace_data slot. Check what numeric and categorical variables are available for plotting using Trace.data.

Usage

```
bb_plot_trace_data(
  trace,
  yvar = "coverage",
  yvar_label = "Coverage",
  facet_var = "group",
  color_var = "group",
  pal = NULL,
  legend_pos = "none",
  group_filter = NULL,
  group_variable = "group"
)
```

Arguments

trace	A Trace object
yvar	The trace_data metadata variable that will become the y axis. Defaults to "coverage". Must be numeric.
yvar_label	The y-axis label for the coverage track. Defaults to "Coverage".
facet_var	The trace_data metadata variable describing data facets. Each will be placed as a separate horizontal track with the value printed to the left. Optional but recommended. Defaults to "group".
color_var	The variable to color groups of traces by. Optional but recommended. Defaults to "group".
pal	A color palette. Can also be added after the fact.
legend_pos	Color legend position. Can also be added after the fact. Defaults to "none".
group_filter	Optional metadatavariable to filter trace data by. When imported from signac/seurat objects, this value defaults to "group", so that is the default here. However if constructed manually, you may wish to apply filtering to another variable. If so, apply it to this parameter.

---

bb_plot_trace_links *Plot The Link Data From A Trace Object*

Description

A function to generate a link plot from tracklike genomic data. Links will automatically be trimmed to lie entirely within the plot range. An additional, optional score cutoff can be provided.

Usage

```
bb_plot_trace_links(
  trace,
  cutoff = 0,
  link_low_color = "grey80",
  link_high_color = "red3",
  link_range = c(0, 1)
)
```

Arguments

trace	A Trace object containing a valid links slot.
cutoff	Score cutoff for link plotting. Defaults to 0.
link_low_color	The color of a link with value of 0, default = grey80
link_high_color	The color of a link with value of 1, default = red3
link_range	The range of the color scale in terms of link values, default = c(0,1)

bb_plot_trace_model *Plot The Gene Model From A Trace Object*

Description

A function to generate a plot of the underlying gene model. The genes to be plotted are automatically selected according to the genome build and the plot range. The function automatically picks the longest principle transcript to show. Optionally, alternative transcripts can be shown by specifying the select_transcript argument. This must be an ensembl transcript identifier lying within the plot range.

Usage

```
bb_plot_trace_model(
  trace,
  font_face = "italic",
  select_transcript = NULL,
  icon_fill = "cornsilk",
  debug = FALSE
)
```

Arguments

trace	A Trace object.
font_face	Font face option to use. Default = "italic".
select_transcript	Optional selected transcript(s) to plot.
icon_fill	The color to make the exon boxes.
debug	Boolean. Option to show the transcript ID on the final plot to confirm you have the right one. Default = FALSE.

bb_plot_trace_peaks *Plot The peak Data From A Trace Object*

Description

A function to generate a peak plot from tracklike genomic data.

Usage

```
bb_plot_trace_peaks(trace, fill_color = "grey60")
```

Arguments

trace	A Trace object.
fill_color	The color to fill the peak graphics with. Defaults to grey60.

bb_print_full_stats *Print out a stats report*

Description

Print out a stats report

Usage

```
bb_print_full_stats(
  data,
  classification_variable,
  numeric_variable,
  test_type = c("Student", "Welch", "Wilcox"),
  output = NULL
)
```

Arguments

data	A Tibble in tidy data format. Must contain or be filtered to contain only 2 levels in "classification_variable" for comparisons.
classification_variable	Column containing the class variable
numeric_variable	The column containing the numeric values to summarize and compare
test_type	Must be one of "Student", "Welch", and "Wilcox"
output	Output file; if null prints to screen.

Value

A text file

bb_promoter_overlap *Calculate The Overlap Between Peaks and Promoters*

Description

For a given GRanges object containing peaks, determine how many peaks overlap promoters, how many promoters are overlapped by peaks and the significance of enrichment of query peaks relative to promoters.

Usage

```
bb_promoter_overlap(query, tss = c("hg38_tss", "dr11_tss"), width = 200)
```

Arguments

query	A GRanges object containing peaks
tss	The tss data base to use. Must be one of "hg38_tss" or "dr11_tss"
width	The width around the tss to evaluate. Defaults to 200 bp.

Value

A list including overlap information and binomal test results.

---

bb_pseudobulk_mf *Run Multifactor Pseudobulk Analysis using Deseq2*

Description

Use this function to perform Pseudbulk DGE analysis.

Usage

```
bb_pseudobulk_mf(
  cds,
  pseudosample_table,
  design_formula,
  count_filter = 10,
  result_recipe = "default",
  test = "Wald",
  reduced = NULL
)
```

Arguments

cds	The cell data set object subset to analyze
pseudosample_table	A tibble indicating the sample groupings for analysis. This should include 1.) Unique sample identifiers 2.) Any sample-level cell metadata you wish to include in the regression model and 3.) Any Cell-level metadata you may wish to include such as clusters or partitions. Values will be coerced to factors.
design_formula	The regression-style formula for the analysis. In the form of "~ variable1 + variable2 + ... final_variable". The default behavior is to calculate results according to the final_variable in the design_formula with preceding variables as co-variates. The reference class is chosen according to alphabetical order. This behavior can be modified by specifying the result_recipe argument.
count_filter	The minimum number of counts required across all pseudosamples in order to keep a gene in the analysis.
result_recipe	See above for the default recipe. Alternatively, supply a 3-element vector in the form of c("variable", "experimental_level","reference_or_control_level")

Value

A list of results from pseudobulk analysis

`bb_pseudotime` *Learn Graph and Calculate Pseudotime*

Description

This function determines a gene expression trajectory using `learn_graph` from monocle3 and then calculates pseudotime dimensions along this trajectory using `order_cells`. So it is 2 functions wrapped into 1. Usually we will not adjust the parameters for learn_graph with the possible exception of `close_loop` and `use_partition` which are also available in this function with the same defaults. If you need to fine-tune the trajectory, use `monocle3::learn_graph` on the cds object first and then run this function to calculate pseudotime. The graph learning will not be repeated on an object unless `force_graph` is set to `TRUE`.

If you just want to look at the trajectory graph and not calculate pseudotime, change `calculate_pseudotime` to FALSE, or run `monocle3::learn_graph`.

After the pseudotime values are calculated, they are handled differently than in monocle3. In this function, they are copied from the hidden CDS slot and made an explicit cell metadata column. Pseudotime needs a starting point or anchor. There is no interactive option here as in monocle3. To identify this starting point, you identify a cell metadata variable and provide it to `cluster_variable`. This should identify a cohesive group of cells in UMAP space such as a leiden cluster, louvain cluster or partition. Then provide a value corresponding to the cluster of interest to `cluster_value`. The function will start pseudotime at the cell closest to the graph node in that cluster. The pseudotime value column will be named automatically as a composite of the `cluster_variable` and `cluster_value` parameters.

Usage

```
bb_pseudotime(
  cds,
  calculate_pseudotime = TRUE,
  cluster_variable,
  cluster_value,
  use_partition = TRUE,
  close_loop = TRUE,
  force_graph = FALSE
)
```

Arguments

`cds`	The cell data set object to calculate pseudotime upon. Does not yet accept seurat objects.
`calculate_pseudotime`	Logical, whether to calculate the pseudotime dimension. If false, will only run learn_graph, Default: TRUE

cluster_variable
: The cell metadata column from which the pseudotime = 0 cell will be selected.

cluster_value
: The value of cluster_variable that identifies a cluster. The cell closest to the root node closest to the center of this cluster will have pseudotime of 0.

use_partition
: Logical; If TRUE, learn_graph will construct trajectories within partitions. If FALSE, it will connect partitions, Default: TRUE

close_loop
: Logical; Whether learn_graph will close looping trajectories, Default: TRUE

force_graph
: Logical; If TRUE, the function will recalculate the graph., Default: FALSE

Value

A cell data set

See Also

cli_div, cli_alert learn_graph, order_cells, pseudotime SummarizedExperiment-class

---

bb_qc *A function to run qc tests on cds objects.*

Description

A function to run qc tests on cds objects.

Usage

```
bb_qc(
  cds,
  cds_name,
  genome = c("human", "mouse", "zfish"),
  nmad_mito = 2,
  nmad_detected = 2,
  max_mito = NULL,
  min_log_detected = NULL
)
```

Arguments

cds
: A cell data set object to run qc functions on

cds_name
: The name of the cds

genome
: The species to use for identifying mitochondrial genes. Choose from "human", "mouse", "zfish", "human_mouse" for pdx.

max_mito
: Manual cutoff for mitochondrial percentage. May be more strict, i.e. lower, than the automated cutoff but not less strict, Default: NULL

min_log_detected
: Manual cutoff for log detected features. May be more strict, i.e. higher, than the automated cutoff not not less strict, Default: NULL

Value

A list of qc objects

`bb_read_bam` *Read Bam Files*

Description

This function reads a single sorted, dedupliated paired end bam file and returns either a GRanges object or a GenomicAlignmentPairs object. The former requires much less memory but a the cost of retaining the outer boundaries of each read. If read 1 has start S1 and end E1 and read 2 has start S2 and end S2, the Granges object spans S1-E2.

Usage

```
bb_read_bam(
  sortedBam,
  genome = c("hg38", "danRer11"),
  return_type = c("GenomicAlignmentPairs", "GRanges")
)
```

Arguments

`sortedBam`	File path to the bam file to load.
`genome`	One of "hg38" or "danRer11". This is used to clean up the granges object if necessary.
`return_type`	Type of object to return. GRanges is smaller. GenomicAlignmentPairs retains read pair data.

Value

An object according to return_type.

`bb_read_narrowpeak` *A Function to Read In Narrow Peaks Files*

Description

This function reads the narrow peaks file (BED6 + 4 format) and turns it into a GRanges object.

Usage

```
bb_read_narrowpeak(file)
```

Arguments

file	The file path to the narrow peaks file.

Value

A GRanges object

---

bb_rejoin *Rejoin qc and doubletfinder data to a cds object*

Description

Rejoin qc and doubletfinder data to a cds object

Usage

```
bb_rejoin(cds, qc_data, doubletfinder_data)
```

Arguments

cds	A cell data set object to rejoin to
qc_data	A table of cell barcodes with qc data. Can be extracted from bb_qc with purrr::map(qc_result, 1)
doubletfinder_data	The doubletfinder result tbl

Value

A cell data set object with qc and doubletfinder data

---

bb_remove_dupes *Remove rows that have duplicates in a given column*

Description

Remove rows that have duplicates in a given column

Usage

```
bb_remove_dupes(data, column)
```

Arguments

data	A tibble.
column	A column to deduplicate

Value

A deduplicated tibble

`bb_rowmeta` *Get Feature/Gene Metadata*

Description

Take a cell_data_set or Seurat object and return the gene/feature metadata in the form of a tibble. RNA is used as the default assay.

Usage

```
bb_rowmeta(
  obj,
  row_name = "feature_id",
  experiment_type = "Gene Expression",
  assay = "RNA",
  cds = NULL
)
```

Arguments

`obj`	A cell_data_set or Seurat object
`row_name`	Optional name to provide for feature unique identifier, Default: 'feature_id'
`experiment_type`	The experiment type to display. Applies only to cds objects. Commonly will be either "Gene Expression" or "Antibody Capture", Default: 'Gene Expression'
`assay`	For a Seurat object, th feature assay to return. CDS objects with alternative experiments are not supported, Default: 'RNA'
`cds`	Provided for compatibility with prior versions, Default: NULL

Details

If a value is supplied for cds, a warning will be issued and the function will pass the value of cds to obj.

Value

At tibble.

bb_seurat_anno *A function to generate automated cell labelings with Seurat*

Description

A function to generate automated cell labelings with Seurat

Usage

```
bb_seurat_anno(cds, reference)
```

Arguments

cds	A cell data set object
reference	Seurat reference data.

Value

A modified cds with Seurat cell assignments.

bb_split_atac *Split Out Peaks Data*

Description

Extracts Peaks data from 10X counts matrix and feature metadata and saves it as an alternate experiment in the assigned CDS.

Usage

```
bb_split_atac(cds)
```

Arguments

cds	CDS to split the Peaks data from

Value

A cell data set

bb_split_citeseq *Split Antibody Capture Data into Alt Experiment*

Description

If you have cite-seq data together with gene expression data, this function will move the cite seq data to a new separate experiment. It will use Seurat to normalize these data using the CLR method and store them in a new assay.

Usage

```
bb_split_citeseq(cds)
```

Arguments

cds	the cell data set to split

Value

a new CDS

See Also

SummarizedExperiment-class

bb_tbl_to_coldata *A Function To Add Tibble Columns To Cell Metadata*

Description

A Function To Add Tibble Columns To Cell Metadata

Usage

```
bb_tbl_to_coldata(obj, min_tbl, join_col = "cell_id", cds = NULL)
```

Arguments

obj	A Seurat or cell data set object
min_tbl	A tibble containing only the columns you want to add plus one column for joining. Cell IDs may not be duplicated but missing cells are ok; values will be replaced by NA.
join_col	The column in min_tbl containing the join information for the cds rowData. Defaults to "cell_id".
cds	Retained for backwards compatibility. If supplied, will generate a warning and pass argument to obj. Default = NULL

Value

An object of the same class

`bb_tbl_to_matrix` *Convert a wide-form tibble a matrix*

Description

Convert a wide-form tibble a matrix

Usage

```
bb_tbl_to_matrix(data)
```

Arguments

`data`	A wide form tibble to convert to a matrix. The first column will become the rownames.

Value

A matrix

`bb_tbl_to_rowdata` *A Function To Add Tibble Columns To Feature Metadata*

Description

A Function To Add Tibble Columns To Feature Metadata

Usage

```
bb_tbl_to_rowdata(
  obj,
  assay = "RNA",
  min_tbl,
  join_col = "feature_id",
  cds = NULL
)
```

Arguments

obj	A Seurat or cell data set object
assay	The assay to which to add the metadata column, Default = RNA
min_tbl	A tibble containing only the columns you want to add plus one column for joining. Features cannot be duplicated but missing features are ok and will be replaced by NA.
join_col	The column in min_tbl containing the join information for the cds rowData. Defaults to "feature_id".
cds	Retained for backwards compatibility. If supplied, will generate a warning and pass argument to obj. Default = NULL

Value

An object of the same class

---

bb_triplecluster *A function to generate clusters from scRNA-seq data*

Description

Based on Monocle3's Partitions, Leiden, and Louvain clustering methods. Implemented mostly with default values. Seurat objects will be converted to cell_data_set objects for the clustering. The function produces a list of top markers for each cluster type and returns these assignments to the original object as new cell metadata columnts.

Usage

```
bb_triplecluster(
  obj,
  n_top_markers = 50,
  outfile = NULL,
  n_cores = 8,
  cds = NULL
)
```

Arguments

obj	A Seurat or cell_data_set object
n_top_markers	Number of top markers to identify per cell group, Default: 50
outfile	Name of a csv file to hold the top marker results. If null, will place "top_markers.csv" in the working directory, Default: NULL
n_cores	Number of processor cores to use, Default: 8
cds	Provided for backwards compatibility for existing code. If a value is supplied it will be transferred to obj and a warning message will be emitted, Default: NULL

Value

A modified Seurat or cell_data_set object

`bb_unblind_images` *Rejoin Blinded Scores to Original File Names*

Description

Will rejoin scoresheet and blinded key to produce unblinded results. If you change the names of either of those files, they have to be provided as arguments to the function. Otherwise keyfile and scorefile are optional.

Usage

```
bb_unblind_images(
  directory,
  keyfile = "blinding_key.csv",
  scorefile = "scoresheet.csv",
  analysis_file,
  file_column
)
```

Arguments

`directory`	The linux-style filepath of the folder containing the scoresheet and blinded key.
`keyfile`	Optional: filename of the key file. Defaults to "blinding_key.csv".
`scorefile`	Optional: filename of the score file. Defaults to "scoresheet.csv".
`analysis_file`	Complete file path to the the unblinded main analysis sheet. The function will will left_join analysis_file and unblinded results. In the process, it will necessarily convert windows file paths to linux-style file paths. Samples not included in the blinding should return with NA values for the added columns. New data columns being added on from scoresheet should be unique relative to analysis_file.
`file_column`	The column in analysis_file containing file paths for the files that were blinded.

Value

nothing

bb_var_umap *A function to generate a UMAP with colors mapped to colData variables*

Description

A function to generate a UMAP with colors mapped to colData variables

Usage

```
bb_var_umap(
  obj,
  var,
  assay = "RNA",
  value_to_highlight = NULL,
  foreground_alpha = 1,
  legend_pos = "right",
  cell_size = 0.5,
  alt_stroke_color = NULL,
  legend_title = NULL,
  plot_title = NULL,
  palette = NULL,
  alt_dim_x = NULL,
  alt_dim_y = NULL,
  overwrite_labels = FALSE,
  group_label_size = 3,
  alt_label_col = NULL,
  shape = 21,
  nbin = 100,
  facet_by = NULL,
  sample_equally = FALSE,
  rasterize = FALSE,
  raster_dpi = 300,
  show_trajectory_graph = FALSE,
  trajectory_graph_color = "grey28",
  trajectory_graph_segment_size = 0.75,
  label_root_node = FALSE,
  pseudotime_dim = var,
  label_principal_points = FALSE,
  graph_label_size = 2,
  cds = NULL,
  outline_cluster = FALSE,
  outline_color = "black",
  outline_size = 1,
  outline_type = "solid",
  outline_alpha = 1,
  ...,
```

```
  man_text_df = NULL,
  text_geom = "text",
  minimum_segment_length = 1
)
```

Arguments

obj	A Seurat or cell data set object
var	The variable to map colors to. Special exceptions are "density", "local_n" and "log_local_n" which calculate the 2 d kernel density estimate or binned cell counts and maps to color scale.
assay	The gene expression assay to draw reduced dimensions from. Default is "RNA". Does not do anything with cell_data_set objects.
value_to_highlight	Option to highlight a single value
foreground_alpha	Alpha value for foreground points
legend_pos	Legend position
cell_size	Cell point size
alt_stroke_color	Alternative color for the data point stroke
legend_title	Title for the legend
plot_title	Main title for the plot
palette	Color palette to use. "Rcolorbrewer", "Viridis" are builtin options. Otherwise provide manual values.
alt_dim_x	Alternate/reference dimensions to plot by.
alt_dim_y	Alternate/reference dimensions to plot by.
overwrite_labels	Whether to overwrite the variable value labels
group_label_size	Size of the overwritten labels
alt_label_col	Alternate column to label cells by
shape	Shape for data points
nbin	Number of bins if using var %in% c("density". "local_n", "log_local_n")
facet_by	Variable or variables to facet by.
sample_equally	Whether or not you should downsample to the same number of cells in each plot. Default is FALSE or no.
rasterize	Whether to render the graphical layer as a raster image. Default is FALSE.
raster_dpi	If rasterize then this is the DPI used. Default = 300.
show_trajectory_graph	Whether to render the principal graph for the trajectory. Requires that learn_graph() has been called on cds.

`trajectory_graph_color`
: The color to be used for plotting the trajectory graph.

`trajectory_graph_segment_size`
: The size of the line segments used for plotting the trajectory graph.

`label_root_node`
: Logical; whether to label the root node for the selected pseudotime trajectory. The function will requires that a valid pseudotime column be identified, usually as the value of the "var" argument in the form of "pseudotime_cluster_value". If you wish to use var to color the cells in some other way, the pseudotime_dim argument needs to be supplied with the correct pseudotime dimension to pick the root node from.

`pseudotime_dim`
: An alternative column to pick the pseudoetime root node from, if not supplied to var.

`label_principal_points`
: Logical indicating whether to label roots, leaves, and branch points with principal point names. This is useful for order_cells and choose_graph_segments in non-interactive mode.

`graph_label_size`
: How large to make the branch, root, and leaf labels.

`cds`
: Provided for backward compatibility with prior versions. If a value is supplied, a warning will be emitted and the value will be transferred to the obj argument, Default: NULL

`...`
: Additional params for facetting.

`man_text_df`
: A data frame in the form of text_x = numeric_vector, text_y = numeric_vector, label = character_vector for manually placing text labels.

`minimum_segment_length`
: Minimum length of a line to draw from label to centroid.

Value

a ggplot

`COMMENTS` *Get the Comments Slot from an Ape Object*

Description

This function gets the comments slot from an Ape object.

Usage

```
COMMENTS(ape)
```

data_median_se *Generate median +/- se stat object for jitter ggplot*

Description

Generate median +/- se stat object for jitter ggplot

Usage

```
data_median_se(x)
```

Arguments

x	A numeric vector

data_summary_mean_sd *Generate mean +/- sd stat object for jitter ggplot*

Description

Generate mean +/- sd stat object for jitter ggplot

Usage

```
data_summary_mean_sd(x)
```

Arguments

x	A numeric vector

data_summary_mean_se *Generate mean +/- se stat object for jitter ggplot*

Description

Generate mean +/- se stat object for jitter ggplot

Usage

```
data_summary_mean_se(x)
```

Arguments

x	A numeric vector

`data_summary_median_iqr` *Generate median +/- iqr stat object for jitter ggplot*

Description

Generate median +/- iqr stat object for jitter ggplot

Usage

```
data_summary_median_iqr(x)
```

Arguments

x	A numeric vector

`data_summary_median_mad` *Generate median +/- mad stat object for jitter ggplot*

Description

Generate median +/- mad stat object for jitter ggplot

Usage

```
data_summary_median_mad(x)
```

Arguments

x	A numeric vector

dMcast *Casts or pivots a long* data frame *into a wide sparse matrix*

Description

Similar in function to dcast, but produces a sparse Matrix as an output. Sparse matrices are beneficial for this application because such outputs are often very wide and sparse. Conceptually similar to a pivot operation.

Usage

```
dMcast(
  data,
  formula,
  fun.aggregate = "sum",
  value.var = NULL,
  as.factors = FALSE,
  factor.nas = TRUE,
  drop.unused.levels = TRUE
)
```

Arguments

data	a data frame
formula	casting formula, see details for specifics.
fun.aggregate	name of aggregation function. Defaults to 'sum'
value.var	name of column that stores values to be aggregated numerics
as.factors	if TRUE, treat all columns as factors, including
factor.nas	if TRUE, treat factors with NAs as new levels. Otherwise, rows with NAs will receive zeroes in all columns for that factor
drop.unused.levels	should factors have unused levels dropped? Defaults to TRUE, in contrast to model.matrix

Details

Casting formulas are slightly different than those in dcast and follow the conventions of model.matrix. See formula for details. Briefly, the left hand side of the ~ will be used as the grouping criteria. This can either be a single variable, or a group of variables linked using :. The right hand side specifies what the columns will be. Unlike dcast, using the + operator will append the values for each variable as additional columns. This is useful for things such as one-hot encoding. Using : will combine the columns as interactions.

Value

a sparse Matrix

See Also

cast

dcast

FEATURES *Get the Features Slot from an Ape Object*

Description

This function cats the features slot from an Ape object.

Usage

```
FEATURES(ape)
```

---

filter_cds *Filter a CDS*

Description

This function provides a pipe-friendly method to filter cds objects.

Usage

```
filter_cds(cds, cells = "all", genes = "all")
```

Arguments

cds	The CDS to filter.
cells	Optional: a tibble of cell metadata for the cells you wish to keep. Use bb_cellmeta(). Default: 'all'
genes	Optional: a tibble of gene metadata for the genes you wish to keep. Use bb_rowmeta(). Default: 'all'

Value

A filtered CDS

geom_split_violin *geom_split_violin*

Description

create a split violin geom

Usage

```
geom_split_violin(
  mapping = NULL,
  data = NULL,
  stat = "ydensity",
  position = "identity",
  ...,
  draw_quantiles = NULL,
  trim = TRUE,
  scale = "area",
  na.rm = FALSE,
  show.legend = NA,
  inherit.aes = TRUE
)
```

Arguments

mapping	Set of aesthetitc mappings, Default: NULL
data	Data to display, Default: NULL
stat	Statistic to plot, Default: 'ydensity'
position	Position, Default: 'identity'
...	extra arguments
draw_quantiles	Quantiles to draw, Default: NULL
trim	Trim ends?, Default: TRUE
scale	Analogous to violin scale, Default: 'area'
na.rm	Default: FALSE
show.legend	Default: NA
inherit.aes	Default: TRUE

granges_to_features — *Format a GRanges Object as a Character Vector For Inclusion in Genebank File*

Description

This function takes a GRanges object and returns a character vector. Only 4 metadata fields from the GRanges object will be included: locus_tag, type, fwdcolor and revcolor. Locus_tag must be unique. This will be checked by the Ape constructor. This function should mostly be used internally in the construction and FEATURE-setting of instances of the Ape class.

Usage

```
granges_to_features(gr)
```

Arguments

gr	A GRanges object.

Value

A character vector.

LearnGraph — *Run* link[monocle3]{learn_graph} *on a* Seurat *object*

Description

Run link[monocle3]{learn_graph} on a Seurat object

Usage

```
LearnGraph(object, reduction = DefaultDimReduc(object = object), ...)
```

Arguments

object	A Seurat object
reduction	Name of reduction to use for learning the pseudotime graph
...	Arguments passed to learn_graph

Value

A cell_data_set object with the pseudotime graph

See Also

learn_graph cell_data_set

LOCUS *Get the Locus Slot from an Ape Object*

Description

This function cats the Locus slot from an Ape object.

Usage

```
LOCUS(ape)
```

merge.Matrix *Merges two Matrices or matrix-like objects*

Description

Implementation of `merge` for `Matrix`. By explicitly calling `merge.Matrix` it will also work for `matrix`, for `data.frame`, and `vector` objects as a much faster alternative to the built-in `merge`.

Usage

```
merge.Matrix(
  x,
  y,
  by.x,
  by.y,
  all.x = TRUE,
  all.y = TRUE,
  out.class = class(x)[1],
  fill.x = ifelse(is(x, "sparseMatrix"), FALSE, NA),
  fill.y = fill.x,
  ...
)

join.Matrix(
  x,
  y,
  by.x,
  by.y,
  all.x = TRUE,
  all.y = TRUE,
  out.class = class(x)[1],
  fill.x = ifelse(is(x, "sparseMatrix"), FALSE, NA),
  fill.y = fill.x,
  ...
)
```

Arguments

x, y	Matrix or matrix-like object
by.x	vector indicating the names to match from Matrix x
by.y	vector indicating the names to match from Matrix y
all.x	logical; if TRUE, then each value in x will be included even if it has no matching values in y
all.y	logical; if TRUE, then each value in y will be included even if it has no matching values in x
out.class	the class of the output object. Defaults to the class of x. Note that some output classes are not possible due to R coercion capabilities, such as converting a character matrix to a Matrix.
fill.x, fill.y	the value to put in merged columns where there is no match. Defaults to 0/FALSE for sparse matrices in order to preserve sparsity, NA for all other classes
...	arguments to be passed to or from methods. Currently ignored

Details

#' all.x/all.y correspond to the four types of database joins in the following way:

left all.x=TRUE, all.y=FALSE

right all.x=FALSE, all.y=TRUE

inner all.x=FALSE, all.y=FALSE

full all.x=TRUE, all.y=TRUE

Note that NA values will match other NA values.

---

normalize_batch *Noramalize a Data Table by Group and Batch*

Description

Often you will have a data table with repeats or batches of the same experiment. An effective way to control for batch effects is to normalize the data from each batch to a control group present in all of the experiments. To use this function, provide such a data table, identify the column holding the experimental group data, the identity of the control group to normalize by, the column holding the batch data, and the column holding the numerical data to normalize. Also select the function to average by (mean or median). The function will return the data table with three new columns: the average of the control group by batch, fold change of each observation relative to the batch average and the log2-transformed fold change. This function is pipe-friendly.

Usage

```
normalize_batch(
  data,
  group_col,
  norm_group,
  batch_col,
  data_col,
  fun = c("mean", "median")
)
```

Arguments

data	a tibble
group_col	the column containing the experimental group identifier
norm_group	the experimental group you want to normalize to across batches
batch_col	the column containing the batch identifier
data_col	the column with your data
fun	averaging function to use, Default: c("mean", "median")

Value

A tibble with new columns indicating batch normalization group average, fold change for each observation relative to the batch average and log2 fold change

See Also

arg_match cli_abort filter, group_by, summarise, select, mutate-joins, mutate

---

rBind.fill *Combine matrixes by row, fill in missing columns*

Description

rbinds a list of Matrix or matrix like objects, filling in missing columns.

Usage

```
rBind.fill(x, ..., fill = NULL, out.class = class(rbind(x, x))[1])
```

Arguments

x, ...	Objects to combine. If the first argument is a list and .. is unpopulated, the objects in that list will be combined.
fill	value with which to fill unmatched columns
out.class	the class of the output object. Defaults to the class of x. Note that some output classes are not possible due to R coercion capabilities, such as converting a character matrix to a Matrix.

Details

Similar to rbind.fill.matrix, but works for Matrix as well as all other R objects. It is completely agnostic to class, and will produce an object of the class of the first input (or of class matrix if the first object is one dimensional).

The implementation is recursive, so it can handle an arbitrary number of inputs, albeit inefficiently for large numbers of inputs.

This method is still experimental, but should work in most cases. If the data sets consist solely of data frames, rbind.fill is preferred.

Value

a single object of the same class as the first input, or of class matrix if the first object is one dimensional

See Also

rbind.fill

rbind.fill.matrix

---

se *Calculate standard error of the mean*

Description

Calculate standard error of the mean

Usage

```
se(x)
```

Arguments

x	A numeric vector

---

show,Ape-method *Show an Ape Object*

Description

Show an Ape Object

Usage

```
## S4 method for signature 'Ape'
show(object)
```

show,Trace-method *Show a Trace Object*

Description

Show a Trace Object

Usage

```
## S4 method for signature 'Trace'
show(object)
```

Trace-class *An S4 class to Hold Genome Track Data*

Description

An instance of this class is created by calling "bb_makeTrace". All slots in this object are GRanges objects. Validation checks will make sure data, peaks, links, and gene models all are bound by the same plot range on the same chromosome of the same genome. This is different from the standard GRangesList object in that each slot can have it's own metadata columns. Currently, hg38 and danRer11 are the supported genomes. Use this class to plot coverage tracks from bulk or single cell ATAC, CHIP, or similar experiments. Link plotting is available for cicero-style links.

Slots

`trace_data` A GRanges object with a metadata column for "score" or "coverage" intended to be plotted as a y-variable. Metadata variables may be included to annotate color and and facets in the final plotting. This is particularly important for single cell data or when combining bulk tracks from different samples. Whole sample bigwig files can be converted to GRanges objects using import.bw from rtracklayer. All track data is trimmed during import but pre-trimming to the approximate range desired will significantly speed up processing. The plyranges package is recommended for granges manipulations such as filtering by chromosome, adding metadata and binding GRanges objects together. Bulk tracks from different samples should be pre-normalized before importing.

`peaks` A GRanges object containing peaks to plot in a track-style format.

`links` A GRanges object with Cicero-style links.

`gene_model` The gene model for plotting. Will be automatically generated by bb_makeTrace.

`plot_range` The master GRange for the whole object. Validity checks and/or constructors ensure all other ranges are contained within.

Trace.data *Get the Trace Data Slot from a Trace Object*

Description

Get the Trace Data Slot from a Trace Object

Usage

```
Trace.data(trace)
```

Trace.gene_model *Get the gene_model Slot from a Trace Object*

Description

Get the gene_model Slot from a Trace Object

Usage

```
Trace.gene_model(trace)
```

Trace.links *Get the Links Slot from a Trace Object*

Description

Get the Links Slot from a Trace Object

Usage

```
Trace.links(trace)
```

Trace.peaks *Get the peaks Slot from a Trace Object*

Description

Get the peaks Slot from a Trace Object

Usage

```
Trace.peaks(trace)
```

Trace.plot_range *Get the plot_range Slot from a Trace Object*

Description

Get the plot_range Slot from a Trace Object

Usage

```
Trace.plot_range(trace)
```

Trace.setData *Set the Trace Data Slot of a GRanges Object*

Description

Set the Trace Data Slot of a GRanges Object

Usage

```
Trace.setData(trace, gr)
```

Arguments

trace	A trace object
gr	A GRanges object. This object will become the new trace_data. If the range is smaller, it will trim the other slots to match. Usually this is used to change range metadata only.

Trace.setLinks *Set the Links Slot of a GRanges Object*

Description

Set the Links Slot of a GRanges Object

Usage

```
Trace.setLinks(trace, gr)
```

Arguments

trace	A trace object
gr	A GRanges object. This object will become the new plot links.

Trace.setpeaks *Set the peaks Slot of a GRanges Object*

Description

Set the peaks Slot of a GRanges Object

Usage

```
Trace.setpeaks(trace, gr)
```

Arguments

trace	A trace object
gr	A GRanges object. This object will become the new plot peaks.

Trace.setRange *Set the Plot Range Slot of a GRanges Object*

Description

Set the Plot Range Slot of a GRanges Object

Usage

```
Trace.setRange(trace, gr)
```

Arguments

trace	A trace object
gr	A GRanges object. This object will become the new plot range.

%notin% *Operators*

Description

Operators

Usage

```
... %notin% NA
```

Index