Package 'optima'

July 26, 2023

Title Tapestri Single-Cell Data Analysis

Version 0.1.0

Description The Tapestri platform offers DNA and protein analysis at the single cell level. Integrating both types of data is beneficial for studying multiple cell populations in heterogeneous microenvironments, such as cancer cells. We present optima, an R package for the processing and analysis of data generated from the Tapestri platform. This package provides streamlined functionality for raw data filtering, integration, normalization, transformation, and visualization. Insights gained from the optima package help users to identify unique cell populations and uncover surface protein expression patterns. The results generated by optima help researchers elucidate dynamic changes at the single cell level in heterogeneous microenvironments.

License 'Artistic-2.0'

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Imports compositions,
dbscan,
httr,
jsonlite,
methods,
pheatmap,
rhdf5,
umap

R topics documented:

`annotateVariant` *Variant annotation*

Description

This function takes variant names as input and returns annotation for annotation table for all variant IDs in a data frame.

Usage

```
annotateVariant(variant.names)
```

Arguments

`variant.names`	Input variant IDs, can be a vector.

Value

A data frame with annotation for all input variant IDs.

Examples

```
annotateVariant(variants_id)
```

`calculatePloidy` *CNV ploidy calculation function*

Description

The function uses the normalized CNV matrix to calculate the ploidy for each CNV locus.

Usage

```
calculatePloidy(optima.obj, diploid.cell)
```

Arguments

`optima.obj`	optima object.
`diploid.cell`	The cell type that should be considered as diploid cell. This cell type should be one of cell types in the cell.labels vector.

Value

optima object with normalized CNV

Examples

```
calculatePloidy(optima.obj)
```

---

drawHeatmap *Heatmap function*

Description

This function creates a heatmap using matrix data. Matrix data can be DNA or protein.

Usage

```
drawHeatmap(optima.obj, omic.type)
```

Arguments

optima.obj	optima object.
omic.type	Type of data for heat map. Potential values "dna" and "protein". If "dna" is specified, then the VAF data will be used for plot. If "protein" is specified, then protein expression values will be used.

Value

Heat map visualization.

Examples

```
drawHeatMap()
```

---

elbowPlot *Generate elbow plot*

Description

This function generate an elbow plot. The plot is useful for determining how many PCs will be used for dimension reduction The input matrix can be the DNA matrix in an optima object.

Usage

```
elbowPlot(input.mtx)
```

Arguments

input.mtx	Input matrix

Value

an elbow plot

Examples

```
elbowPlot(example.matrix)
```

`filterVariant` *DNA Variant filter function*

Description

This function uses multiple matrices imported from the h5 file to conduct quality filtering. This includes the sequencing depth matrix, genotype matrix, variant allele frequency matrix, genotype quality matrix The function returns an optima object that has been filtered with variant/cells. In addition, the returned optima object's variant.filter label is changed to "filtered". This function is usually applied before protein and CNV analysis.

Usage

```
filterVariant(
  optima.obj,
  min.dp = 10,
  min.gq = 30,
  vaf.ref = 5,
  vaf.hom = 95,
  vaf.het = 35,
  min.cell.pt = 50,
  min.mut.cell.pt = 1
)
```

Arguments

`optima.obj`	An optima object with raw data unfiltered.
`min.dp`	Minimum depth, defaults to 10. Read depth ranges from 0 to positive infinity. When to change: If your sample ended up with more cells being sequenced than planned, then you may have less reads per cell. In such cases, you could try to lower this parameter to 7, 8 or 9. It is not common to increase this parameter above 10.
`min.gq`	Minimum genotype quality. The default value is 30. The possible range of GQ score is 0-99. This GQ score is derived from GATK. A higher score means a more confident genotype call. A lower score means genotype call with low confidence. When to change: You may consider decreasing the threshold when the reads sequencing quality is low overall.
`vaf.ref`	If reference call vaf (GT=0) is larger than vaf.ref, then value in genotype call matrix is converted to GT=3.
`vaf.hom`	If homozygous call vaf (GT=2) is smaller than vaf.hom, then value in genotype call matrix is converted to GT=3.

`vaf.het` If heterozygous call vaf (GT=1) is smaller than vaf.ref, then value in genotype call matrix is converted to GT=3. If heterozygous call vaf (GT=1) is smaller than vaf.ref, then value in genotype call matrix is converted to GT=3. The default value is 35. For heterozygous cell, there should be 50/50 in two alleles, respectively. However, due to sampling error, there are chances when we sequenced more reads in one allele than the other. This parameter allows for tolerating such situations. We don't normally change this parameter.

`min.cell.pt` Minimum threshold for cell percentage that has valid variant call (GT = 0, 1 or 2) after applying the filter. Minimum threshold for cell percentage that has valid variant call (GT = 0, 1 or 2) after applying the filter. The default value is 50%. This means for one variant; we need at least 50% of cells have a valid variant call. When to change: If the variant of interest is in a high GC content region, then PCR amplification is hard. In such cases, you may choose to decrease the percent to 30 or 40 so that your interested variant could come through the filter.

`min.mut.cell.pt`
Minimum threshold for cell percentage that has mutated genotype (GT = 1 or 2) after applying the filter. The default is 1, corresponds to 1%. This filter is used to remove false positives. When to change: If you know the variant is rare in the data, then you could try lower threshold to try to keep the variant in your dataset.

Value

An optima object, The DNA data in the object is filtered, the variant.filter label is "filtered". Meanwhile, the protein matrix and CNV matrix is also updated so that only cells withstand DNA variant filter are kept.

Examples

```
filterVariant(optima.obj)
```

`findSignature` *Identify signature protein function*

Description

This function compares protein levels for a input cell type against all other cells using t test. This function returns a data frame ranked by FDR adjusted p-value. If two cell types specified, then this function compares protein expression between the two cell types.

Usage

```
findSignature(optima.obj, cell.type, cell.type.2 = "all")
```

Arguments

`optima.obj`	optima object.
`cell.type`	Input cell type to compare protein level to all other cell types.
`cell.type.2`	Second cell type to compare.

Value

Data frame of all proteins p-values comparing protein levels of input cell type to all other cell types. In addition, it provides the mean difference between Input cell type and other cells. If mean difference is positive, this mean more expression in the cell type.

Examples

```
findSignature(optima.obj, cell.type)
```

`getClones` *Clustering variant function*

Description

This function identifies cell clones based on DNA variant data.

Usage

```
getClones(optima.obj, eps = 1, minPts = 100, num.PC = 5, plot = FALSE)
```

Arguments

`optima.obj`	optima object.
`eps`	size/radius of the epsilon neighborhood. This argument will passed to dbscan function.
`minPts`	number of minimum points required in the eps neighborhood for core points, including the point itself. This argument will passed to dbscan function.
`plot`	if True, a UMAP plot will be generated based on the dimension reduction result from variant matrix. Default is FALSE.

Value

A list, the first element in the list is the optima object with labels assigned based on dbscan. the second element in the list is the dimension reduction result based on VAF matrix.

Examples

```
getClones(my.obj)
```

getCNVmtx *The getter function for CNV matrix*

Description

This function returns the CNV matrix within the optima object.

Usage

```
getCNVmtx(optima.obj)
```

Arguments

optima.obj	optima object.

Value

A matrix that contains CNV data in the optima object. The row names are cell id, the column names are CNV IDs.

Examples

```
getCNVmtx(my.obj)
```

getDNAmtx *The getter function for VAF matrix*

Description

This function returns the VAF matrix within the optima object.

Usage

```
getDNAmtx(optima.obj)
```

Arguments

optima.obj	optima object.

Value

A matrix that contains VAF data in the optima object. The row names are cell id, the column names are variant ID.

Examples

```
getDNAmtx(my.obj)
```

getInfo *Single variant ID annotation function*

Description

Returns annotation from one variant ID. This function is not visable to users.

Usage

```
getInfo(variant)
```

Arguments

variant	variant name in a string

Value

Annotation information for one specific variant ID.

Examples

```
getInfo(variant_id)
```

getPloidyMtx *The getter function for Ploidy matrix*

Description

This function returns the Ploidy matrix within the optima object.

Usage

```
getPloidyMtx(optima.obj)
```

Arguments

optima.obj	optima object.

Value

A matrix that contains Ploidy data in the optima object. The row names are cell id, the column names are CNV IDs.

Examples

```
getPloidyMtx(my.obj)
```

getProteinMtx *The getter function for Protein matrix*

Description

This function returns the Protein matrix within the optima object.

Usage

```
getProteinMtx(optima.obj)
```

Arguments

optima.obj	optima object.

Value

A matrix that contains Protein data in the optima object. The row names are cell IDs, the column names are protein IDs.

Examples

```
getProteinMtx(my.obj)
```

normalizeCNV *CNV normalization function*

Description

The function normalizes the CNV matrix to correct for column-wise and row-wise variation and updates the optima object amp.normalize.method from "unnormalized" to "normalized".

Usage

```
normalizeCNV(optima.obj)
```

Arguments

optima.obj	optima object.

Value

optima object with normalized CNV and amp.normalize.method updated to "normalized".

Examples

```
normalizeCNV(optima.obj)
```

normalizeProtein *Protein matrix normalization*

Description

The function normalizes protein matrix within an optima object using CLR method.

Usage

```
normalizeProtein(optima.object)
```

Arguments

optima.obj	optima object.

Value

An optima object with protein matrix being normalized and protein.normalize.method label updated to "normalized".

Examples

```
normalizeProtein(optima.object)
```

optima-class *optima object*

Description

An optima object contains DNA, protein and CNV for Tapestri platform single cell sequencing data.

Value

Object containing DNA, protein and CNV single cell sequencing data.

Slots

`meta.data` User-defined metadata can be kept with the object.

`cell.ids` A vector of cell IDs/barcodes from Tapestri. This vector should contain unique IDs.

`cell.labels` A vector that is used to store the cell type information for each cell.

`variants` A vector of variant IDs.

`variant.filter` A string that keeps track of if optima object is being QC filtered on its variant matrix.

`vaf.mtx` Variant matrix.

`gt.mtx` Genotype matrix, The integers within matrix are reference call GT=0, heterozygous call GT=1, homozygous call GT=2, no calls GT = 3.

`dp.mtx` Sequencing depth matrix.

`gq.mtx` Genotype quality.

`amps` A vector of CNV locus.

`amp.normalized.method` A string that keeps track of if optima object is being normalized on its CNV matrix.

`amp.mtx` CNV matrix.

`ploidy.mtx` Ploidy matrix.

`proteins` A vector of surface protein ID.

`protein.normalize.method` A string that keeps track of if optima object is being normalized on its protein matrix.

`protein.mtx` Protein matrix

Examples

```
setClass()
```

`plotPloidy` *Ploidy scatter plot function*

Description

For a specified cell type, this function creates a scatter plot indicating ploidy for different CNV loci.

Usage

```
plotPloidy(optima.obj, cell.type)
```

Arguments

`optima.obj`	optima object.
`cell.type`	String that indicates which cell type,

Value

optima object with normalized CNV.

Examples

```
plotPloidy()
```

plotProteinFeature *Plot specific protein levels*

Description

This function create a plot based on protein level dimension reduction result. Within the plot, Each cell was colored based on the protein level

Usage

```
plotProteinFeature(optima.obj, protein.name, reduceDim.obj)
```

Arguments

optima.obj	optima object.
protein.name	the specific protein user interested
reduceDim.obj	dimension reduction result returned by reduceDim() function.

Value

A scatter plot based on dimension reduction result. Each cell is colored based on the protein level. The more expression.

Examples

```
plotProteinFeature(my.obj, "CD11b", protein.reduceDim)
```

plotVariantFeature *Plot specific variant VAF levels*

Description

This function create a plot based on VAF dimension reduction result. Within the plot, Each cell was colored based on the VAF

Usage

```
plotVariantFeature(optima.obj, vaf.name, reduceDim.obj)
```

Arguments

optima.obj	optima object.
vaf.name	the specific variant name user interested
reduceDim.obj	dimension reduction result returned by reduceDim() function.

Value

A scatter plot based on dimension reduction result. Each cell is colored based on the protein level. The more expression.

Examples

```
plotVariantFeature(my.obj, "chr4:106190862:T/C", vaf.reduceDim)
```

`readHdf5` *H5 file to optima object function*

Description

This function read in a h5 file and return one optima object. The h5 file can be found in the Tapestri pipeline software output. The .h5 file contains all necessary data needed for single cell DNA and protein analysis.

Usage

```
readHdf5(directory, sample.name, omic.type = "DNA+protein")
```

Arguments

`directory`	Directory for the input h5 file.
`sample.name`	A sample name that will be used naming visualizations.
`omic.type`	This parameter indicates if the data set is DNA only or DNA+protein. It has two possible values: "DNA+protein" or "DNA". The default value is "DNA+protein".

Value

optima object

Examples

```
readHdf5("path/to/my/file.h5")
```

`reduceDim` *Dimension reduction function*

Description

This function reduces dimensions for a data matrix, such data matrix can be protein or DNA matrix in an optima object.

Usage

```
reduceDim(input.mtx, num.PC = 5)
```

Arguments

`input.mtx`	Input optima object.
`num.PC`	The number of PCs being used for preserving variation. Default is 5

Value

List containing PCA result and umap result derived from first 5 PCs

Examples

```
reduceDim(example.matrix)
```

Index