Package 'karyotapR'

May 8, 2026

Title DNA Copy Number Analysis for Genome-Wide Tapestri Panels

Version 1.0.2

Description Analysis of DNA copy number in single cells using custom genome-wide targeted DNA sequencing panels for the Mission Bio Tapestri platform. Users can easily parse, manipulate, and visualize datasets produced from the automated 'Tapestri Pipeline', with support for normalization, clustering, and copy number calling. Functions are also available to deconvolute multiplexed samples by genotype and parsing barcoded reads from exogenous lentiviral constructs.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

URL https://github.com/joeymays/karyotapR, http://joeymays.xyz/karyotapR/

BugReports https://github.com/joeymays/karyotapR/issues

Imports circlize, cli, ComplexHeatmap, dbscan, dplyr, fitdistrplus, GenomeInfoDb, GenomicRanges, ggplot2, gtools, IRanges, magrittr, methods, purrr, rhdf5, rlang, S4Vectors, stats, SummarizedExperiment, tibble, tidyr, umap, viridisLite

Depends R (>= 4.1), SingleCellExperiment

Suggests Biostrings, knitr, rmarkdown, Rsamtools, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/Needs/website forcats, purrr, ggdist

NeedsCompilation no

Author Joseph Mays [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-4903-938X>)

Maintainer Joseph Mays <josephcmays@gmail.com>

Repository CRAN

Date/Publication 2025-11-01 22:40:08 UTC

Contents

`assayBoxPlot` *Generate a box plot from assay data*

Description

Draws box plot of data from indicated `TapestriExperiment` `assay` slot. This is especially useful for visualizing `altExp` count data, such as counts from probes on chrY or barcode probe counts.

Usage

```
assayBoxPlot(
  TapestriExperiment,
  alt.exp = NULL,
  assay = NULL,
  log.y = TRUE,
  split.features = FALSE,
  split.x.by = NULL,
  split.y.by = NULL
)
```

Arguments

TapestriExperiment	TapestriExperiment object
alt.exp	Character, altExp to plot. NULL (default) uses the top-level experiment in TapestriExperiment.
assay	Character, assay to plot. NULL (default) selects first assay listed TapestriExperiment.
log.y	Logical, if TRUE, scales data using log1p(). Default TRUE.
split.features	Logical, if TRUE, splits plot by rowData features if slot has more than one row feature/probe. Default FALSE.
split.x.by	Character, colData column to use for X-axis categories. Default NULL.
split.y.by	Character, colData column to use for Y-axis splitting/faceting. Default NULL.

Value

ggplot object, box plot

See Also

ggplot2::geom_boxplot()

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
assayBoxPlot(tap.object, alt.exp = "chrYCounts", split.features = TRUE, split.x.by = "test.cluster")
```

---

assayHeatmap *Generate heatmap of assay data*

Description

Creates a heatmap of data from the indicated TapestriObject assay slot using the ComplexHeatmap package. Heatmaps are generated as transposed (i.e. x-y flipped) representations of the assay matrix. Additional ComplexHeatmap::Heatmap() parameters can be passed in to overwrite defaults.

Usage

```
assayHeatmap(
  TapestriExperiment,
  alt.exp = NULL,
  assay = NULL,
  split.col.by = NULL,
  split.row.by = NULL,
  annotate.row.by = NULL,
  color.preset = NULL,
  color.custom = NULL,
  ...
)
```

Arguments

TapestriExperiment	TapestriExperiment object
alt.exp	Character, altExp slot to use. NULL (default) uses top-level/main experiment.
assay	Character, assay slot to use. NULL (default) uses first-indexed assay (usually "counts").
split.col.by	Character, rowData column to split columns by, i.e. "chr" or "arm". Default NULL.
split.row.by	Character, colData column to split rows by, i.e. "cluster". Default NULL.
annotate.row.by	Character, colData column to use for block annotation. Default NULL.
color.preset	Character, color preset to use for heatmap color, either "copy.number" or "copy.number.denoise" (see Details). Overrides color.custom. NULL (default) uses default ComplexHeatmap coloring.
color.custom	Color mapping function given by circlize::colorRamp2(). color.preset must be NULL.
...	Additional parameters to pass to ComplexHeatmap::Heatmap().

Value

A ComplexHeatmap object

Options for color.preset

"copy.number":
Blue-white-red gradient from 0-2-4. 4 to 8+ is red-black gradient.

```
circlize::colorRamp2(c(0,1,2,3,4,8),
c('#2c7bb6','#abd9e9','#ffffff','#fdae61','#d7191c', "black"))
```

"copy.number.denoise":
Similar to 'copy.number' present, but white range is from 1.5-2.5 to reduce the appearance of noise around diploid cells.

```
circlize::colorRamp2(c(0,1,1.5,2,2.5,3,4,8),
c('#2c7bb6','#abd9e9','#ffffff','#ffffff','#ffffff','#fdae61','#d7191c', "black"))
```

See Also

Heatmap

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
assayHeatmap(tap.object,
  assay = "counts", split.row.by = "test.cluster",
  annotate.row.by = "test.cluster", split.col.by = "chr"
)
```

calcCopyNumber *Calculate relative copy number value for each cell-probe unit using reference sample*

Description

`calcCopyNumber()` transforms the normalized count matrix `normcounts` of a `TapestriExperiment` object into copy number values based on a set of reference cell barcodes and given copy number value (e.g. 2 for diploid). This is practically used to set the median copy number of a usually diploid reference cell population to a known copy number value, e.g. 2, and then calculate the copy number for all the cells relative to that reference population. This occurs individually for each probe, such that the result is one copy number value per cell barcode per probe (cell-probe unit). `control.copy.number` is a `data.frame` lookup table used to indicate the copy number value and cell barcodes to use as the reference. A template for `control.copy.number` can be generated using `generateControlCopyNumberTemplate()`, which will have a row for each chromosome arm represented in `TapestriExperiment`.

The `control.copy.number` data.frame should include 3 columns named `arm`, `copy.number`, and `sample.label`. `arm` is chromosome arm names from chr1p through chrXq, `copy.number` is the reference copy number value (2 = diploid), and `sample.label` is the value corresponding to the `colData` column given in `sample.feature` to indicate the set of reference cell barcodes to use to set the copy number. This is best used in a workflow where the cells are clustered first into their respective samples, and then one cluster is used as the reference population the other clusters. This also allows for the baseline copy number to be set for each chromosome arm individually in the case where the reference population is not completely diploid.

Usage

```
calcCopyNumber(
  TapestriExperiment,
  control.copy.number,
  sample.feature = "cluster",
  remove.bad.probes = FALSE
)

generateControlCopyNumberTemplate(
  TapestriExperiment,
  copy.number = 2,
  sample.feature.label = NA
)
```

Arguments

`TapestriExperiment`
: `TapestriExperiment` object.

`control.copy.number`
: `data.frame` with columns `arm`, `copy.number`, and `sample.label`. See details.

`sample.feature` Character, `colData` column to use for subsetting cell.barcodes. Default "cluster".

`remove.bad.probes` Logical, if `TRUE`, probes with median normalized counts = 0 are removed from the returned `TapestriExperiment`. If FALSE (default), probes with median normalized counts = 0 throw error and stop function.

`copy.number` Numeric, sets all entries of `copy.number` column in output. Default 2 (diploid).

`sample.feature.label` Character, sets all entries of `sample.label` column in output.

Value

`TapestriExperiment` object with cell-probe copy number values in `copyNumber` assay slot.

`data.frame` with 3 columns named `arm`, `copy.number`, and `sample.label`

Functions

- `generateControlCopyNumberTemplate()`: generates a `data.frame` template for `control.copy.number` in `calcCopyNumber()`.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- calcNormCounts(tap.object)
control.copy.number <- generateControlCopyNumberTemplate(tap.object,
  copy.number = 2,
  sample.feature.label = "cellline1"
)
tap.object <- calcCopyNumber(tap.object,
  control.copy.number,
  sample.feature = "test.cluster"
)
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
control.copy.number <- generateControlCopyNumberTemplate(tap.object,
  copy.number = 2,
  sample.feature.label = "cellline1"
)
```

---

calcGMMCopyNumber *Call copy number for each cell-chromosome using Gaussian mixture models*

Description

Uses control cells to simulate expected smoothed copy number distributions for all chromosomes across each of `model.components` (copy number level). Then uses the distributions to calculate posterior probabilities for each cell-chromosome belonging to each of copy number level. Each cell-chromosome is assigned the copy number value for which its posterior probability is highest. This is done for both whole chromosomes and chromosome arms.

Usage

```
calcGMMCopyNumber(
  TapestriExperiment,
  cell.barcodes,
  control.copy.number,
  model.components = 1:5,
  model.priors = NULL,
  ...
)
```

Arguments

TapestriExperiment	TapestriExperiment object.
cell.barcodes	character, vector of cell barcodes to fit GMM. Usually corresponds to diploid control.
control.copy.number	data.frame with columns arm and copy.number, indicating of known copy number of cells in cell.barcodes.
model.components	numeric, vector of copy number GMM components to calculate, default 1:5 (for copy number = 1, 2, 3, 4, 5).
model.priors	numeric, relative prior probabilities for each GMM component. If NULL (default), assumes equal priors.
...	Additional parameters to be passed to internal functions.

Value

TapestriExperiment object with copy number calls based on the calculated GMMs, saved to gmmCopyNumber slot of smoothedCopyNumberByChr and smoothedCopyNumberByArm altExps. GMM parameters for each feature.id are saved to the metadata slot.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- calcNormCounts(tap.object)
control.copy.number <- generateControlCopyNumberTemplate(tap.object,
  copy.number = 2,
  sample.feature.label = "cellline1"
)
tap.object <- calcCopyNumber(tap.object,
  control.copy.number,
  sample.feature = "test.cluster"
)
tap.object <- calcSmoothCopyNumber(tap.object)
tap.object <- calcGMMCopyNumber(tap.object,
  cell.barcodes = colnames(tap.object),
  control.copy.number = control.copy.number,
  model.components = 1:5
```

```
)
```

`calcNormCounts` *Normalize raw counts*

Description

Normalizes raw counts from `counts` slot in `TapestriExperiment` and returns the object with normalized counts in the `normcounts` slot. Also calculates the standard deviation for each probe using normalized counts and adds it to `rowData`.

Usage

```
calcNormCounts(TapestriExperiment, method = "mb", scaling.factor = NULL)
```

Arguments

`TapestriExperiment`	`TapestriExperiment` object.
`method`	Character, normalization method. Default "mb".
`scaling.factor`	Numeric, optional number to scale normalized counts if `method == "libNorm"`. Default `NULL`.

Details

"mb" method performs the same normalization scheme as in Mission Bio's mosaic package for python: Counts for each barcode are normalized relative to their barcode's mean and probe counts are normalized relative to their probe's median. "libNorm" method preforms library size normalization, returning the proportion of counts of each probe within a cell. The proportion is multiplied by `scaling.factor` if provided.

Value

`TapestriExperiment` object with normalized counts added to `normcounts` slot.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- calcNormCounts(tap.object)
```

`calcSmoothCopyNumber` *Smooth copy number values across chromosomes and chromosome arms*

Description

`calcSmoothCopyNumber()` takes `copyNumber` slot values for probes on a chromosome and smooths them by median (default) for each chromosome and chromosome arm, resulting in one copy number value per chromosome and chromosome arm for each cell barcode. Cell-chromosome values are then discretized into integers by conventional rounding (1.5 <= x < 2.5 rounds to 2). Smoothed copy number and discretized smoothed copy number values are stored as `smoothedCopyNumber` and `discreteCopyNumber` assays, in `altExp` slots `smoothedCopyNumberByChr` for chromosome-level smoothing, and `smoothedCopyNumberByArm` for chromosome arm-level smoothing.

Usage

```
calcSmoothCopyNumber(TapestriExperiment, method = "median")
```

Arguments

`TapestriExperiment`	`TapestriExperiment` object.
`method`	Character, smoothing method: median (default) or mean.

Value

`TapestriExperiment` with `smoothedCopyNumber` and `discreteCopyNumber` assays in `altExp` slots `smoothedCopyNumberByChr` and `smoothedCopyNumberByArm`.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- calcNormCounts(tap.object)
control.copy.number <- generateControlCopyNumberTemplate(tap.object,
  copy.number = 2,
  sample.feature.label = "cellline1"
)
tap.object <- calcCopyNumber(tap.object,
  control.copy.number,
  sample.feature = "test.cluster"
)
tap.object <- calcSmoothCopyNumber(tap.object)
```

callSampleLables *Call sample labels based on feature counts*

Description

callSampleLables() assigns labels (stored as colData column) to cells using feature count data in colData. This is most useful for assigning barcode labels based on barcoded reads (see countBarcodedReads). For method = max, labels are dictated by whichever input.features column has the highest number of counts. By default, ties are broken by choosing whichever label has the lowest index position (ties.method = "first"). Samples with 0 counts for all input.features columns are labeled according to neg.label. If only one feature column is used, labels are assigned to cells with counts > min.count.threshold, and neg.label otherwise.

Usage

```
callSampleLables(
  TapestriExperiment,
  input.features,
  output.feature = "sample.call",
  return.table = FALSE,
  neg.label = NA,
  method = "max",
  ties.method = "first",
  min.count.threshold = 1
)
```

Arguments

TapestriExperiment	A TapestriExperiment object.
input.features	Character vector, column names in colData to evaluate.
output.feature	Character, column name to use for the call output. Default "sample.call".
return.table	Logical, if TRUE, returns a data.frame of the sample.calls. If FALSE (default), returns updated TapestriExperiment object.
neg.label	Character, label for samples with no counts. Default NA.
method	Character, call method. Only "max" currently supported, calls based on whichever input.features column has the most counts.
ties.method	Character, passed to max.col() indicating how to break ties. Default "first".
min.count.threshold	Numeric, minimum number of counts per cell to use for call. Default 1.

Value

A TapestriExperiment object with sample calls added to colData column sample.name. If return.table == TRUE, a data.frame of sample calls.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
colData(tap.object)$gRNA1 <- 2 # example barcode counts
colData(tap.object)$gRNA2 <- 10 # example barcode counts
tap.object <- callSampleLables(tap.object,
  input.features = c("gRNA1", "gRNA2"),
  output.feature = "sample.grna"
)
```

---

corner *Print the top-left corner of a matrix*

Description

Outputs up to 5 rows and columns of the input matrix object (with `rownames` and `colnames`) to get a quick look without filling the console.

Usage

```
corner(input.mat)
```

Arguments

`input.mat`	A matrix-like object.

Value

A matrix-like object matching input class, subset to a maximum of 5 rows and columns.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
corner(assay(tap.object, "counts"))
```

---

countBarcodedReads *Get read counts from barcoded reads*

Description

`countBarcodedReads()` and `countBarcodedReadsFromContig()` match exogenous DNA barcode sequences to their associated cell barcodes and saves them to the `colData` (cell barcode metadata) of `TapestriExperiment`. `countBarcodedReads()` is a shortcut for `countBarcodedReadsFromContig()`, allowing the user to specify 'gRNA' or 'barcode' to use the `grnaCounts` or `barcodeCounts` `altExp` slots. The entries in the `barcode.lookup` table do not have to be present in the sample, allowing users to keep one master table/file of available barcode sequences for use in all experiments. The `Rsamtools` and `Biostrings` packages must be installed to use these functions.

Usage

```
countBarcodedReads(
  TapestriExperiment,
  bam.file,
  barcode.lookup,
  probe,
  return.table = FALSE,
  max.mismatch = 2,
  with.indels = FALSE,
  ...
)

countBarcodedReadsFromContig(
  bam.file,
  barcode.lookup,
  contig,
  cell.barcode.tag = "RG",
  max.mismatch = 2,
  with.indels = FALSE
)
```

Arguments

TapestriExperiment	TapestriExperiment object
bam.file	File path of BAM file. .bai BAM index file must be in the same location (can be generated using Rsamtools::indexBam()).
barcode.lookup	data.frame where the first column is the barcode identifier/name and the second column is the DNA sequence. Headers are ignored.
probe	Character, either "gRNA" or "barcode" to parse counts from grnaCounts or barcodeCounts altExp slots, respectively.
return.table	Logical, if TRUE, returns table of read counts per barcode. If FALSE, returns TapestriExperiment. Default FALSE.
max.mismatch	Numeric, the maximum and minimum number of mismatching letters allowed. Default 2.
with.indels	If TRUE, then indels are allowed. Default FALSE.
...	Arguments to pass on to countBarcodedReadsFromContig().
contig	Character, contig or chromosome name to search for barcodes in. Can be a vector of more than one contig to expand search space.
cell.barcode.tag	Character of length 2, indicates cell barcode field in BAM, specified by Tapestri pipeline (currently "RG"). Default "RG".

Value

TapestriExperiment with barcoded read counts added to colData.

A data.frame of read counts for each specified barcode.

See Also

Rsamtools::indexBam()

Biostrings::matchPattern()

Examples

```
## Not run:
counts <- countBarcodedReads(
  TapestriExperiment,
  bam.file, barcode.lookup, "gRNA"
)

## End(Not run)
## Not run:
counts <- countBarcodedReadsFromContig(bam.file, barcode.lookup, "virus_ref2")

## End(Not run)
```

---

createTapestriExperiment

Create `TapestriExperiment` *object from Tapestri Pipeline output*

Description

`createTapestriExperiment()` constructs a `TapestriExperiment` container object from data stored in the `.h5` file output by the Tapestri Pipeline. Read count matrix (probe x cell barcode) is stored in the "counts" `assay` slot of the top-level experiment. Allele frequency matrix (variant x cell barcode) is stored in the "alleleFrequency" `assay` slot of the "alleleFrequency" `altExp` (alternative experiment) slot. `panel.id` is an optional shortcut to set special probe identities for specific custom panels.

Usage

```
createTapestriExperiment(
  h5.filename,
  panel.id = NULL,
  get.cytobands = TRUE,
  genome = "hg19",
  move.non.genome.probes = TRUE,
  filter.variants = TRUE,
  verbose = TRUE
)
```

Arguments

`h5.filename`	File path for `.h5` file from Tapestri Pipeline output.

panel.id	Character, Tapestri panel ID, either CO261, CO293, CO610, or NULL. Initializes barcodeProbe and grnaProbe slots. Default NULL.
get.cytobands	Logical, if TRUE (default), retrieve and add chromosome cytobands and chromosome arms to rowData (probe metadata).
genome	Character, reference genome for pulling cytoband coordinates and chromosome arm labels (see getCytobands()). Only "hg19" (default) is currently supported.
move.non.genome.probes	Logical, if TRUE (default), move counts and metadata from non-genomic probes to altExp slots (see moveNonGenomeProbes()).
filter.variants	Logical, if TRUE (default), only stores variants that have passed Tapestri Pipeline filters.
verbose	Logical, if TRUE (default), metadata is output in message text.

Value

TapestriExperiment object containing data from Tapestri Pipeline output.

Panel ID Shortcuts

panel.id is an optional shortcut to set the barcodeProbe and grnaProbe slots in TapestriExperiment for specific custom Tapestri panels.

CO261:

- barcodeProbe = "not specified"
- grnaProbe = "not specified"

CO293:

- barcodeProbe = "AMPL205334"
- grnaProbe = "AMPL205666"

CO610:

- barcodeProbe = "CO610_AMP351"
- grnaProbe = "CO610_AMP350"

Automatic Operations

Raw Data:

Read count and allele frequency matrices are imported to their appropriate slots as described above. filter.variants == TRUE (default) only loads allele frequency variants that have passed internal filters in the Tapestri Pipeline. This greatly reduces the number of variants from tens of thousands to hundreds of likely more consequential variants, saving RAM and reducing operation time.

Metadata:

Several metadata sets are copied or generated and then stored in the appropriate TapestriExperiment slot during construction.

- Probe panel metadata stored in the `.h5` file are copied to `rowData`.
- Basic QC stats (e.g. total number of reads per probe) are added to `rowData`.
- Basic QC stats (e.g. total number of reads per cell barcode) are added to `colData`.
- Experiment-level metadata is stored in `metadata`.

Optional Operations

Two additional major operations are called by default during `TapestriExperiment` construction for convenience. `get.cytobands == TRUE` (default) calls `getCytobands()`, which retrieves the chromosome arm and cytoband for each probe based on stored positional data and saves them in `rowData`. Some downstream smoothing and plotting functions may fail if chromosome arms are not present in `rowData`, so this generally should always be run. `move.non.genome.probes` calls `moveNonGenomeProbes()`, which moves probes corresponding to the specified tags to `altExp` (alternative experiment) slots in the `TapestriExperiment` object. The exception is probes on chromosome Y; CNVs of chrY are more rare, so we move it to an `altExp` for separate analysis. Probes corresponding to the `barcodeProbe` and `grnaProbe` slots, which are specified by the `panel.id` shortcut or manually (see Custom Slot Getters and Setters), are automatically moved to `altExp` by this operation as well. If such probes are not present, the function will only generate a warning message, so it is always safe (and recommended) to run by default. Any remaining probes that are not targeting a human chromosome and are not specified by the shortcut tags are moved to the `otherProbeCounts` slot.

See Also

`moveNonGenomeProbes()`, `getCytobands()`, which are run as part of this function by default.

Examples

```
## Not run:
tapExperiment <- createTapestriExperiment("myh5file.h5", "CO293")

## End(Not run)
```

Custom Slot Getters and Setters

Getter and Setter functions for `TapestriExperiment` *slots*

Description

Get and set custom slots in `TapestriExperiment`. Slots include `barcodeProbe` for a sample barcode probe ID and `grnaProbe` for a gRNA-associated probe ID. These are used as shortcuts for `moveNonGenomeProbes()` and `countBarcodedReads()`. `gmmParams` holds parameters and metadata for GMM copy number calling models.

Usage

```
barcodeProbe(x)

## S4 method for signature 'TapestriExperiment'
barcodeProbe(x)

barcodeProbe(x) <- value

## S4 replacement method for signature 'TapestriExperiment'
barcodeProbe(x) <- value

grnaProbe(x)

## S4 method for signature 'TapestriExperiment'
grnaProbe(x)

grnaProbe(x) <- value

## S4 replacement method for signature 'TapestriExperiment'
grnaProbe(x) <- value

gmmParams(x)

## S4 method for signature 'TapestriExperiment'
gmmParams(x)
```

Arguments

x	A TapestriExperiment object
value	Character, probe ID to assign to slot
TapestriExperiment	A TapestriExperiment object

Value

For the getter methods barcodeProbe, grnaProbe, and gmmParams, the value of the given slot is returned. For the setter methods barcodeProbe and grnaProbe, a TapestriExperiment object is returned with modifications made to the given slot.

Functions

- barcodeProbe(TapestriExperiment): barcodeProbe getter
- barcodeProbe(TapestriExperiment) <- value: barcodeProbe setter
- grnaProbe(TapestriExperiment): grnaProbe getter
- grnaProbe(TapestriExperiment) <- value: grnaProbe setter
- gmmParams(TapestriExperiment): gmmParams getter

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
barcodeProbe(tap.object) <- "Probe01"
barcodeProbe(tap.object)

grnaProbe(tap.object) <- "Probe02"
grnaProbe(tap.object)

gmmParams(tap.object)
```

`getChrOrder` *Get chromosome order from a string of chromosome/contig names*

Description

`getChrOrder()` takes a string of chromosome or contig names and returns the indices of the string in typical chromosome order, i.e. 1 through 22, X, Y. Contig names that do not match 1:22, X, or Y are sorted numerically and alphabetically (with numbers coming first), and added to the end of the order. The output string can then be used to sort the input string into typical chromosome order.

Usage

```
getChrOrder(chr.vector)
```

Arguments

`chr.vector`	Character vector of chromosome or contig names.

Value

A numerical vector of the input vectors indices in chromosome order.

Examples

```
chr.order <- getChrOrder(c(1, "virus", 5, "X", 22, "plasmid", "Y"))
ordered.vector <- c(1, "virus", 5, "X", 22, "plasmid", "Y")[chr.order]
```

getCytobands *Add chromosome cytobands and chromosome arms to TapestriExperiment*

Description

getCytobands() retrieves the chromosome arm and cytoband for each probe based on stored positional data and saves them in rowData. This is run automatically as part of createTapestriExperiment(). Note: Some downstream smoothing and plotting functions may fail if chromosome arms are not present in rowData.

Usage

```
getCytobands(TapestriExperiment, genome = "hg19", verbose = TRUE)
```

Arguments

TapestriExperiment	TapestriExperiment object.
genome	Character, reference genome to use. Only hg19 is currently supported.
verbose	Logical, if TRUE (default), progress is output as message text.

Value

TapestriExperiment object with rowData updated to include chromosome arms and cytobands.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- getCytobands(tap.object, genome = "hg19")
```

getGMMBoundaries *Calculate decision boundaries between components of copy number GMMs*

Description

Calculate decision boundaries between components of copy number GMMs

Usage

```
getGMMBoundaries(TapestriExperiment, chromosome.scope = "chr")
```

Arguments

TapestriExperiment
: TapestriExperiment object.

chromosome.scope
: "chr" or "arm", for using models for either whole chromosomes or chromosome arms. Default "chr".

Value

tibble containing boundary values of GMMs for each feature.id.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- calcNormCounts(tap.object)
control.copy.number <- generateControlCopyNumberTemplate(tap.object,
  copy.number = 2,
  sample.feature.label = "cellline1"
)
tap.object <- calcCopyNumber(tap.object,
  control.copy.number,
  sample.feature = "test.cluster"
)
tap.object <- calcSmoothCopyNumber(tap.object)
tap.object <- calcGMMCopyNumber(tap.object,
  cell.barcodes = colnames(tap.object),
  control.copy.number = control.copy.number,
  model.components = 1:5
)

boundaries <- getGMMBoundaries(tap.object,
  chromosome.scope = "chr"
)
```

---

getTidyData *Get tidy-style data from* TapestriExperiment *objects*

Description

getTidyData() pulls data from the indicated assay and/or altExp slot(s), and rearranges it into tidy format. colData (cell metadata) from the top-level/main experiment is included. rowData (probe metadata) from the indicated assay and/or altExp slot(s) is included. Attempts are made to sort by "chr" and "start.pos" columns if they are present to simplify plotting and other downstream operations.

Usage

```
getTidyData(
  TapestriExperiment,
  alt.exp = NULL,
  assay = NULL,
  feature.id.as.factor = TRUE
)
```

Arguments

TapestriExperiment	TapestriExperiment object.
alt.exp	Character, altExp slot to use. NULL (default) uses top-level/main experiment.
assay	Character, assay slot to use. NULL (default) uses first-indexed assay (often "counts").
feature.id.as.factor	Logical, if TRUE (default), the feature.id column is returned as a factor.

Value

A tibble of tidy data with corresponding metadata from colData and rowData.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tidy.data <- getTidyData(tap.object, alt.exp = "alleleFrequency")
```

---

moveNonGenomeProbes *Move non-genome probes counts and metadata to altExp slots*

Description

moveNonGenomeProbes() takes the probe IDs corresponding to grnaProbe and barcodeProbe slots of the TapestriExperiment object, as well as probes on chrY, and moves them to their own altExp slots in the object. This allows those counts and associated metadata to be manipulated separately without interfering with the probes used for CNV measurements which target the endogenous genome. SingleCellExperiment::splitAltExps() can be used for manual specification of probes to move to altExp slots if the shortcut slots are not used.

Usage

```
moveNonGenomeProbes(TapestriExperiment)
```

Arguments

TapestriExperiment	TapestriExperiment object.

Details

moveNonGenomeProbes() moves probes corresponding to the specified tags to altExp (alternative experiment) slots in the TapestriExperiment object. These probes should be those which do not correspond to a chromosome and therefore would not be used to call copy number variants. The exception is probes on chromosome Y; CNVs of chrY are more rare, so we move it to an altExp for separate analysis. Probes corresponding to the barcodeProbe and grnaProbe slots, which are specified by the panel.id shortcut or manually (see Custom Slot Getters and Setters), are automatically moved to altExp by this operation as well. If such probes are not present, the function will only generate a warning message, so it is always safe (and recommended) to run by default. Any remaining probes that are not targeting a human chromosome and are not specified by the shortcut tags are moved to the otherProbeCounts slot. This function is run automatically by default and with default behavior as part of createTapestriExperiment().

Value

TapestriExperiment with altExp slots filled with counts and metadata for non-genomic probes.

See Also

SingleCellExperiment::splitAltExps() for manual specification of probes to move to altExp slots.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment
tap.object <- moveNonGenomeProbes(tap.object)
```

---

newTapestriExperimentExample

Create Example TapestriExperiment

Description

Creates a TapestriExperiment object for demonstration purposes, which includes 240 probes across the genome, and 300 cells of 3 types. Raw counts are generated randomly. Type 1 has 75 cells, all XY, all diploid. Type 2 has 100 cells, all XX, with 3 copies of chr 7, otherwise diploid. Type 3 has 125 cells, all XY, with 1 copy of chr 1p, otherwise diploid.

Usage

```
newTapestriExperimentExample()
```

Value

TapestriExperiment object with demo data.

Examples

```
tapExperiment <- newTapestriExperimentExample()
```

PCAKneePlot *Plot of PCA proportion of variance explained*

Description

Draws "knee plot" of PCA proportion of variance explained to determine which principal components (PCs) to include for downstream applications e.g. clustering. Variance explained for each PC is indicated by the line. Cumulative variance explained is indicated by the bars.

Usage

```
PCAKneePlot(TapestriExperiment, alt.exp = "alleleFrequency", n.pcs = 10)
```

Arguments

TapestriExperiment	TapestriExperiment object
alt.exp	Character, altExp to use, NULL uses top-level/main experiment. Default "alleleFrequency".
n.pcs	Numeric, number of PCs to plot, starting at 1. Default 10.

Value

ggplot2 object, combined line plot and bar graph

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- runPCA(tap.object, alt.exp = "alleleFrequency")
PCAKneePlot(tap.object, n.pcs = 5)
```

plotCopyNumberGMM *Plot copy number GMM components*

Description

Plots the probability densities of GMM components for given chromosome or chromosome arm, store in a TapestriExperiment. calcGMMCopyNumber() must be run first.

Usage

```
plotCopyNumberGMM(
  TapestriExperiment,
  feature.id = 1,
  chromosome.scope = "chr",
  draw.boundaries = FALSE
)
```

Arguments

TapestriExperiment	TapestriExperiment object.
feature.id	chromosome or chromosome arm to plot.
chromosome.scope	"chr" or "arm", for plotting models for either whole chromosomes or chromosome arms.
draw.boundaries	logical, if TRUE, draw decision boundaries between each Gaussian component.

Value

ggplot object, density plot

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- calcNormCounts(tap.object)
control.copy.number <- generateControlCopyNumberTemplate(tap.object,
  copy.number = 2,
  sample.feature.label = "cellline1"
)
tap.object <- calcCopyNumber(tap.object,
  control.copy.number,
  sample.feature = "test.cluster"
)
tap.object <- calcSmoothCopyNumber(tap.object)
tap.object <- calcGMMCopyNumber(tap.object,
  cell.barcodes = colnames(tap.object),
  control.copy.number = control.copy.number,
  model.components = 1:5
)

tap.object <- plotCopyNumberGMM(tap.object,
  feature.id = 7,
  chromosome.scope = "chr",
  draw.boundaries = TRUE
)
```

---

reducedDimPlot *Scatter plot for dimensional reduction results*

Description

Plots a scatter plot of the indicated dimensional reduction results.

Usage

```
reducedDimPlot(
  TapestriExperiment,
  alt.exp = "alleleFrequency",
  dim.reduction,
  dim.x = 1,
  dim.y = 2,
  group.label = NULL
)
```

Arguments

TapestriExperiment	TapestriExperiment object
alt.exp	Character, altExp to use, NULL uses top-level/main experiment. Default "alleleFrequency".
dim.reduction	Character, dimension reduction to plot, either "PCA" or "UMAP".
dim.x	Numeric, index of dimensional reduction data to plot on X axis. Default 1.
dim.y	Numeric, index of dimensional reduction data to plot on Y axis. Default 2.
group.label	Character, colData column for grouping samples by color. Default NULL.

Value

ggplot2 object, scatter plot

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- runPCA(tap.object, alt.exp = "alleleFrequency")
reducedDimPlot(tap.object, dim.reduction = "pca")
```

---

runClustering *Cluster 2D data*

Description

Clusters data using dbscan method and saves cluster assignments for each cell barcode to colData. Generally used to assign clusters to UMAP projection after PCA and UMAP dimensional reduction.

Usage

```
runClustering(
  TapestriExperiment,
  alt.exp = "alleleFrequency",
  dim.reduction = "UMAP",
  eps = 0.8,
```

```
  dim.1 = 1,
  dim.2 = 2,
  ...
)
```

Arguments

TapestriExperiment	TapestriExperiment object
alt.exp	Character, altExp slot to use. NULL uses top-level/main experiment. Default "alleleFrequency".
dim.reduction	Character, reduced dimension data to use. Default "UMAP".
eps	Numeric, dbscan eps parameter. Lower to increase cluster granularity. See dbscan::dbscan(). Default 0.8.
dim.1	Numeric, index of data dimension to use. Default 1.
dim.2	Numeric, index of data dimension to use. Default 2.
...	Additional parameters to pass to dbscan::dbscan().

Value

TapestriExperiment object with updated colData containing cluster assignments.

See Also

dbscan::dbscan()

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- runPCA(tap.object, alt.exp = "alleleFrequency")
tap.object <- runUMAP(tap.object, pca.dims = 1:3)
tap.object <- runClustering(tap.object, dim.reduction = "UMAP", eps = 0.8)
```

---

runPCA *Cluster assay data by Principal Components Analysis*

Description

Analyzes assay data by Principal Components Analysis (PCA) and saves results to reducedDims slot of TapestriObject.

Usage

```
runPCA(
  TapestriExperiment,
  alt.exp = "alleleFrequency",
  assay = NULL,
  sd.min.threshold = NULL,
  center = TRUE,
  scale. = TRUE
)
```

Arguments

TapestriExperiment	TapestriExperiment object
alt.exp	Character, altExp to use, NULL uses top-level/main experiment. Default "alleleFrequency".
assay	Character, assay to use. NULL (default) uses first-indexed assay.
sd.min.threshold	Numeric, minimum threshold for allelefreq.sd. Increase to run PCA on fewer, more variable dimensions. Set to NULL if not using for alleleFrequency slot. Default NULL.
center	Logical, if TRUE (default), variables are shifted to be zero centered. See stats::prcomp().
scale.	Logical,if TRUE (default), variables are scaled to have unit variance prior to PCA. See stats::prcomp().

Value

TapestriExperiment with PCA results saved to reducedDims slot of altExp, and proportion of variance explained by each PC saved to metadata slot of altExp.

See Also

stats::prcomp() for PCA method details.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment
tap.object <- runPCA(tap.object, alt.exp = "alleleFrequency")
```

runUMAP *Cluster matrix data by UMAP*

Description

Analyzes matrix data by UMAP and saves results to reducedDims slot of TapestriObject.

Usage

```
runUMAP(
  TapestriExperiment,
  alt.exp = "alleleFrequency",
  assay = NULL,
  use.pca.dims = TRUE,
  pca.dims = NULL,
  ...
)
```

Arguments

TapestriExperiment	TapestriExperiment object
alt.exp	Character, altExp to use, NULL uses top-level/main experiment. Default "alleleFrequency".
assay	Character, assay to use. NULL (default) uses first-indexed assay. Not used when use.pca.dims = TRUE.
use.pca.dims	Logical, if TRUE, uses experiment PCA, otherwise uses assay data. Default TRUE.
pca.dims	Numeric, indices of PCs to use in UMAP. Default NULL.
...	Additional parameters to pass to umap::umap(), e.g. for configuration (see umap::umap.defaults()).

Value

TapestriExperiment with UMAP embeddings saved to reducedDims slot of altExp.

Examples

```
tap.object <- newTapestriExperimentExample() # example TapestriExperiment object
tap.object <- runPCA(tap.object, alt.exp = "alleleFrequency")
tap.object <- runUMAP(tap.object, pca.dims = 1:3)
```

TapestriExperiment-class
TapestriExperiment Class Definition

Description

TapestriExperiment Class Definition

Usage

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

Arguments

object	An R object
TapestriExperiment	A TapestriExperiment object

Value

TapestriExperiment object

Methods (by generic)

- show(TapestriExperiment): Show method for TapestriExperiment

Slots

barcodeProbe character.

grnaProbe character.

gmmParams list.

Examples

```
tapExpObject <- new("TapestriExperiment")
```

Index