# Package 'chromswitch'

October 16, 2019

**Title** An R package to detect chromatin state switches from epigenomic data

**Version** 1.6.0

**Date** 2017-09-20

**Description** Chromswitch implements a flexible method to detect chromatin state switches between samples in two biological conditions in a specific genomic region of interest given peaks or chromatin state calls from ChIP-seq data.

**Depends** R (>= 3.4), GenomicRanges (>= 1.26.4)

**Imports** cluster (>= 2.0.6), Biobase (>= 2.36.2), BiocParallel (>= 1.8.2), dplyr (>= 0.5.0), gplots(>= 3.0.1), graphics, grDevices, IRanges (>= 2.4.8), lazyeval (>= 0.2.0), matrixStats (>= 0.52), magrittr (>= 1.5), methods, NMF (>= 0.20.6), rtracklayer (>= 1.36.4), S4Vectors (>= 0.14.4), stats, tidyr (>= 0.6.3)

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**VignetteBuilder** knitr

**Suggests** BiocStyle, DescTools (>= 0.99.19), devtools (>= 1.13.3), GenomeInfoDb (>= 1.16.0), knitr, rmarkdown, mclust (>= 5.3), testthat

**RoxygenNote** 6.0.1

**URL** https://github.com/sjessa/chromswitch

**BugReports** https://github.com/sjessa/chromswitch/issues

**biocViews** ImmunoOncology, MultipleComparison, Transcription, GeneExpression, DifferentialPeakCalling, HistoneModification, Epigenetics, FunctionalGenomics, Clustering

**git_url** https://git.bioconductor.org/packages/chromswitch

**git_branch** RELEASE_3_9

**git_last_commit** 1c017d6

**git_last_commit_date** 2019-05-02

**Date/Publication** 2019-10-15

**Author** Selin Jessa [aut, cre], Claudia L. Kleinman [aut]

**Maintainer** Selin Jessa <selinjessa@gmail.com>

# R topics documented:



---

`binarizePeaks` *binarizePeaks*

---

## Description

Given peaks for a set of samples in a query region, construct a sample-by- feature matrix where each row is a binary vector which models the presence or absence of unqiue peaks in the region.

## Usage

```
binarizePeaks(localpeaks, p)
```

## Arguments

| | |
|---|---|
| `localpeaks` | LocalPeaks object storing peaks for all samples in the query region |
| `p` | Numeric value in [0, 1] giving the fraction of reciprocal overlap to require. |

## Value

A data frame where rows are samples and columns are features. The genomic ranges which give the features are returned as the `features` attribute of the data frame.

## Examples

```
samples <- c("E068", "E071", "E074", "E101", "E102", "E110")
bedfiles <- system.file("extdata", paste0(samples, ".H3K4me3.bed"),
package = "chromswitch")

metadata <- data.frame(Sample = samples,
    H3K4me3 = bedfiles,
    stringsAsFactors = FALSE)

lpk <- retrievePeaks(H3K4me3,
    metadata = metadata,
    region = GRanges(seqnames = "chr19",
    ranges = IRanges(start = 54924104, end = 54929104)))

# Get feature matrix
ft_matrix <- binarizePeaks(lpk, 0.5)

# See features
attr(ft_matrix, "features")
```

---

`callBinary` *callBinary*

---

## Description

One of two main functions in the `chromswitch` package, this function detects a switch in chromatin state in one or more regions given ChIP-seq peak calls for one mark, executing the entire algorithm from preprocessing to evaluating the clustering results, using the binary strategy.

## Usage

```
callBinary(query, metadata, peaks, filter = FALSE, filter_columns = NULL,
  filter_thresholds = NULL, reduce = TRUE, gap = 300, p = 0.4,
  n_features = FALSE, heatmap = FALSE, titles = NULL, outdir = NULL,
  optimal_clusters = TRUE, estimate_state = FALSE, test_condition = NULL,
  BPPARAM = bpparam())
```

## Arguments

| | |
|---|---|
| `query` | GRanges list containing one or more genomic regions of interest in which to call a switch. The output dataframe will contain one row per region in `query`. |
| `metadata` | A dataframe with at least two columns: "Sample" which stores the sample IDs, "Condition", which stores the biological condition labels of the samples |
| `peaks` | List of GRanges objects storing peak calls for each sample, where element names correspond to sample IDs |
| `filter` | (Optional) logical value, filter peaks based on thresholds on peak statistics? Default: FALSE. The filter step is described in `filterPeaks`. |
| `filter_columns` | If `filter` is TRUE, a chracter vector corresponding to names of columns in the peak metadata by which to filter peaks. If `filter` is FALSE, not used. |

filter_thresholds
: If filter is TRUE, a numeric vector corresponding to lower cutoffs applied to metadata columns in order to filter peaks. Provide one per column specified in filter_columns, in the same order. If filter is FALSE, not used.

reduce
: (Optional) logical value, if TRUE, reduce gaps between nearby peaks in the same sample. See more at reducePeaks. Default: TRUE

gap
: (Optional) If reduce is TRUE, numeric value, specifying the threshold distance for merging. Peaks in the same sample which are within this many bp of each other will be merged. Default: 300

p
: Numeric value in [0, 1] giving the fraction of reciprocal overlap to require. Default: 0.4

n_features
: (Optional) Logical value indicating whether to include a column "n_features" in the output storing the number of features in the feature matrix constructed for the region, which may be useful for understanding the behaviour of the binary strategy for constructing feature matrices. Default: FALSE

heatmap
: (Optional) Logical value, plot the heatmap corresponding to the hierarchical clustering result? Default: FALSE

titles
: (Optional) if heatmap is TRUE, a character vector of the same length as query, specifying the title to use when plotting each heatmap (e.g. a gene name), also reused as the prefix of the name of the file where the heatmap is saved. By default, the title is the genomic coordinates of the region in the form "chrN:start-end"

outdir
: (Optional) if heatmap is TRUE, the name of the directory where heatmaps should be saved

optimal_clusters
: (Optional) Logical value indicate whether to cluster samples into two groups, or to find the optimal clustering solution by choosing the set of clusters which maximizes the Average Silhouette width. Default: TRUE.

estimate_state
: (Optional) Logical value indicating whether to include a column "state" in the output specifying the estimated chromatin state of a test condition. The state will be on of "ON", "OFF", or NA, where the latter results if a binary switch between the conditions is unclear. Default: FALSE.

test_condition
: (Optional) If estimate_state is TRUE, string specifying one of the two biological condtions in metadata$Condition for which to estimate chromatin state.

BPPARAM
: (Optional) instance of BiocParallel:BiocParallelParam used to determine the back-end used for parallel computations when performing the analysis on more than one region.

## Details

This strategy constructs a sample-by-feature matrix to use as input for hierarchical clustering by first assembling the set of unique peaks observed in the region across samples. Then for each unique peak, we model the presence or absence of that peak in each sample, resulting in a binary feature matrix.

## Value

Data frame with one row per region in query. Contains the coordinates of the region, the number of inferred clusters, the computed cluster validity statistics, and the cluster assignment for each sample.

## Examples

```
samples <- c("E068", "E071", "E074", "E101", "E102", "E110")
bedfiles <- system.file("extdata", paste0(samples, ".H3K4me3.bed"),
package = "chromswitch")
Conditions <- c(rep("Brain", 3), rep("Other", 3))

metadata <- data.frame(Sample = samples,
    H3K4me3 = bedfiles,
    Condition = Conditions,
    stringsAsFactors = FALSE)

regions <- GRanges(seqnames = c("chr19", "chr19"),
    ranges = IRanges(start = c(54924104, 54874318),
                                end = c(54929104, 54877536)))

callBinary(query = regions, metadata = metadata, peaks = H3K4me3,
           BPPARAM = BiocParallel::SerialParam())
```

---

`callSummary` *callSummary*

---

## Description

One of two main functions in the `chromswitch` package, this function detects a switch in chromatin state in one or more regions given ChIP-seq peak calls for one mark, executing the entire algorithm from preprocessing to evaluating the clustering results, using the summary strategy.

## Usage

```
callSummary(query, metadata, peaks, mark, filter = FALSE,
  filter_columns = summarize_columns, filter_thresholds = NULL,
  summarize_columns = NULL, normalize_columns = summarize_columns,
  tail = 0.005, normalize = ifelse(is.null(normalize_columns) &&
  is.null(summarize_columns), FALSE, TRUE), fraction = TRUE, n = FALSE,
  heatmap = FALSE, titles = NULL, outdir = NULL,
  optimal_clusters = TRUE, estimate_state = FALSE, signal_col = NULL,
  test_condition = NULL, BPPARAM = bpparam())
```

## Arguments

| | |
|---|---|
| `query` | GRanges list containing one or more genomic regions of interest in which to call a switch. The output dataframe will contain one row per region in `query`. |
| `metadata` | A dataframe with at least two columns: "Sample" which stores the sample IDs, "Condition", which stores the biological condition labels of the samples |
| `peaks` | List of GRanges objects storing peak calls for each sample, where element names correspond to sample IDs |
| `mark` | Character specifying the histone mark or ChIP-target, for example, "H3K4me3" |
| `filter` | (Optional) logical value, filter peaks based on thresholds on peak statistics? Default: FALSE. The filter step is described in `filterPeaks`. |

`filter_columns` If `filter` is TRUE, a chracter vector corresponding to names of columns in the peak metadata by which to filter peaks. If `filter` is FALSE, not used.

`filter_thresholds`
If `filter` is TRUE, a numeric vector corresponding to lower cutoffs applied to metadata columns in order to filter peaks. Provide one per column specified in `filter_columns`, in the same order. If `filter` is FALSE, not used.

`summarize_columns`
Character vector of column names on which to compute summary statistics during feature matrix construction. These statistics become the features of the matrix.

`normalize_columns`
If `normalize` is TRUE, a character vector corresponding to names of columns in the peak metadata to normalize genome-wide for each sample. If `normalize` is FALSE, not used.

`tail` (Optional) if `normalize` is TRUE, specifies the fraction of extreme values in each tail to bound during normalization. More details at `normalizePeaks`.

`normalize` (Optional) logical value, normalize peak statistics genome-wide for each sample? Default: TRUE if `summarize_columns` or `normalize_columns` is specified, FALSE, otherwise.

`fraction` (Optional) Logical value, during feature matrix construction, compute the fraction of the region overlapped by peaks? Default: TRUE

`n` (Optional) Logical value, during feature matrix construction, compute the number of peaks in the region? Default: FALSE

`heatmap` (Optional) Logical value, plot the heatmap corresponding to the hierarchical clustering result? Default: FALSE

`titles` (Optional) if `heatmap` is TRUE, a character vector of the same length as `query`, specifying the title to use when plotting each heatmap (e.g. a gene name), also reused as the prefix of the name of the file where the heatmap is saved. By default, the title is the genomic coordinates of the region in the form "chrN:start-end"

`outdir` (Optional) if `heatmap` is TRUE, the name of the directory where heatmaps should be saved

`optimal_clusters`
(Optional) Logical value indicate whether to cluster samples into two groups, or to find the optimal clustering solution by choosing the set of clusters which maximizes the Average Silhouette width. Default: TRUE

`estimate_state` (Optional) Logical value indicating whether to include a column "state" in the output specifying the estimated chromatin state of a test condition. The state will be on of "ON", "OFF", or NA, where the latter results if a binary switch between the conditions is unclear. Default: FALSE.

`signal_col` (Optional) If `estimate_state` is TRUE, string specifying the name of the column in the original peak files which corresponds to the level of enrichment in the region, e.g. fold change

`test_condition` (Optional) If `estimate_state` is TRUE, string specifying one of the two biological condtions in `metadata$Condition` for which to estimate chromatin state.

`BPPARAM` (Optional) instance of `BiocParallel:BiocParallelParam` used to determine the back-end used for parallel computations when performing the analysis on more than one region.

## Details

This strategy constructs a sample-by-feature matrix to use as input for hierarchical clustering by computing, for each sample, a vector of summary statistics based on that sample's peaks in the query region. The summary statistics are generally based on the enrichment statistics associated with each peak as returned by the peak calling too, which might include, for example, a p value and fold change.

## Value

Data frame with one row per region in `query`. Contains the coordinates of the region, the number of inferred clusters, the computed cluster validity statistics, and the cluster assignment for each sample.

## Examples

```
samples <- c("E068", "E071", "E074", "E101", "E102", "E110")
bedfiles <- system.file("extdata", paste0(samples, ".H3K4me3.bed"),
package = "chromswitch")
Conditions <- c(rep("Brain", 3), rep("Other", 3))

metadata <- data.frame(Sample = samples,
    H3K4me3 = bedfiles,
    Condition = Conditions,
    stringsAsFactors = FALSE)

regions <- GRanges(seqnames = c("chr19", "chr19"),
    ranges = IRanges(start = c(54924104, 54874318),
                                end = c(54929104, 54877536)))

callSummary(query = regions,
                metadata = metadata,
                peaks = H3K4me3,
                normalize_columns = c("qValue", "pValue", "signalValue"),
                mark = "H3K4me3",
                summarize_columns = c("pValue", "qValue", "signalValue"),
                heatmap = FALSE,
                BPPARAM = BiocParallel::SerialParam())
```

---

`chromswitch` *chromswitch: An R package for detecting chromatin state switches*

---

## Description

chromswitch implements a flexible method to detect chromatin state switches between samples in two biological conditions in a specific genomic region of interest given peaks called from ChIP-seq data.

## classEntropy *classEntropy*

### Description

Computes the entropy of a set of classes, as defined in https://aclweb.org/anthology/D/D07/D07-1043.pdf

### Usage

```
classEntropy(contingency, c, k)
```

### Arguments

| | |
|---|---|
| contingency | Table, contingency table between clusters and conditions as returned by the table function |
| c | Vector of classes |
| k | Vector of clusters |

### Value

Numeric

### Examples

```
clusters <- c(0, 0, 2, 1, 1, 0, 1)
classes <- c("A", "A", "A", "B", "B", "A", "B")
ct <- table(classes, clusters)
classEntropy(contingency = ct)
```

## cluster *cluster*

### Description

Given a sample-by-feature matrix and sample-associated metadata including their biological condition groupings, cluster samples hierarchically and use external cluster validity measures (Adjusted Rand Index, Normalized Mutual Information, and V measure) to assess the agreement between the inferred clusters and the biological conditions. Optionally, produce a heatmap reflecting the hierarchical clustering result.

### Usage

```
cluster(ft_mat, metadata, query, heatmap = FALSE, title = NULL,
  outdir = NULL, optimal_clusters = TRUE, n_features = FALSE,
  estimate_state = FALSE, method = NULL, test_condition = NULL,
  signal_col = NULL, mark = NULL)
```

## Arguments

| | |
|---|---|
| ft_mat | matrix where columns are features and rows are samples as returned by summarizePeaks or binarizePeaks |
| metadata | A dataframe with a column "Sample" which stores the sample identifiers, and a column "Condition", which stores the biological condition labels of the samples |
| query | GRanges object specifying the query region |
| heatmap | (Optional) Logical value indicating whether to plot the heatmap for hierarchical clustering. Default: FALSE |
| title | (Optional) If heatmap is TRUE, specify the title of the plot, which will also be used for the output file name in PDF format |
| outdir | (Optional) String specifying the name of the directory where PDF of heatmaps should be saved |
| optimal_clusters | (Optional) Logical value indicate whether to cluster samples into two groups, or to find the optimal clustering solution by choosing the set of clusters which maximizes the Average Silhouette width. Default: TRUE |
| n_features | (Optional) Logical value indicating whether to include a column "n_features" in the output storing the number of features in the feature matrix constructed for the region, which may be useful for understanding the behaviour of the binary strategy for constructing feature matrices. Default: FALSE |
| estimate_state | (Optional) Logical value indicating whether to include a column "state" in the output specifying the estimated chromatin state of a test condition. The state will be on of "ON", "OFF", or NA, where the latter results if a binary switch between the conditions is unclear. Default: FALSE. |
| method | (Optional) If estimate_state is TRUE, one of "summary" or "binary", specifying which method was used to construct the feature matrix in ft_mat |
| test_condition | (Optional) If estimate_state is TRUE, string specifying one of the two biological condtions in metadata$Condition for which to estimate chromatin state. |
| signal_col | (Optional) If estimate_state is TRUE, and method is "summary", string specifying the name of the column in the original peak files which corresponds to the level of enrichment in the region, e.g. fold change |
| mark | (Optional) If estimate_state is TRUE, and method is "summary",string specifying the name of the mark for which ft_mat was constructed |

## Value

A dataframe with the region, the number of clusters inferred, the cluster validity statistics, and the cluster assignments for each sample

## Examples

```
samples <- c("E068", "E071", "E074", "E101", "E102", "E110")
bedfiles <- system.file("extdata", paste0(samples, ".H3K4me3.bed"),
package = "chromswitch")
Conditions <- c(rep("Brain", 3), rep("Other", 3))

metadata <- data.frame(Sample = samples,
    H3K4me3 = bedfiles,
    Condition = Conditions,
```

```
    stringsAsFactors = FALSE)

region <- GRanges(seqnames = "chr19",
    ranges = IRanges(start = 54924104, end = 54929104))

lpk <- retrievePeaks(H3K4me3,
    metadata = metadata,
    region = region)

ft_mat <- summarizePeaks(lpk, mark = "H3K4me3",
cols = c("qValue", "signalValue"))

cluster(ft_mat, metadata, region)

# Estimate the state of the test condition, "Brain"
cluster(ft_mat, metadata, region,
    estimate_state = TRUE,
    method = "summary",
    signal_col = "signalValue",
    mark = "H3K4me3",
    test_condition = "Brain")
```

---

clusterEntropy *clusterEntropy*

---

## Description

Computes the entropy of a set of clusters, as defined in https://aclweb.org/anthology/D/D07/D07-1043.pdf

## Usage

```
clusterEntropy(contingency, c, k)
```

## Arguments

| | |
|---|---|
| contingency | Table, contingency table between clusters and conditions as returned by the table function |
| c | Vector of classes |
| k | Vector of clusters |

## Value

Numeric

## Examples

```
clusters <- c(0, 0, 2, 1, 1, 0, 1)
classes <- c("A", "A", "A", "B", "B", "A", "B")
ct <- table(classes, clusters)
clusterEntropy(contingency = ct)
```

completeness *completeness*

## Description

Computes the completeness of a set of clusters given ground-truth classes, as defined in https://aclweb.org/anthology/D/D
1043.pdf

## Usage

```
completeness(contingency, c, k)
```

## Arguments

| | |
|---|---|
| contingency | Table, contingency table between clusters and conditions as returned by the table function |
| c | Vector of classes |
| k | Vector of clusters |

## Value

Numeric

## Examples

```
clusters <- c(0, 0, 2, 1, 1, 0, 1)
classes <- c("A", "A", "A", "B", "B", "A", "B")
ct <- table(classes, clusters)
completeness(contingency = ct)
```

conditionalClassEntropy *classEntropyGivenClusters*

## Description

Computes the conditional entropy of a set of classes, given the cluster assignments, as defined in https://aclweb.org/anthology/D/D07/D07-1043.pdf

## Usage

```
conditionalClassEntropy(contingency, c, k)
```

## Arguments

| | |
|---|---|
| contingency | Table, contingency table between clusters and conditions as returned by the table function |
| c | Vector of classes |
| k | Vector of clusters |

## Value

Numeric

## Examples

```
clusters <- c(0, 0, 2, 1, 1, 0, 1)
classes <- c("A", "A", "A", "B", "B", "A", "B")
ct <- table(classes, clusters)
conditionalClassEntropy(contingency = ct)
```

---

`conditionalClusterEntropy` *clusterEntropyGivenClasses*

---

## Description

Computes the conditional entropy of a set of clusters, given the true classes, as defined in https://aclweb.org/anthology/D/D
1043.pdf

## Usage

```
conditionalClusterEntropy(contingency, c, k)
```

## Arguments

| | |
|---|---|
| `contingency` | Table, contingency table between clusters and conditions as returned by the `table` function |
| `c` | Vector of classes |
| `k` | Vector of clusters |

## Value

Numeric

## Examples

```
clusters <- c(0, 0, 2, 1, 1, 0, 1)
classes <- c("A", "A", "A", "B", "B", "A", "B")
ct <- table(classes, clusters)
conditionalClusterEntropy(contingency = ct)
```

coordToGRanges *coordToGRanges*

## Description

Convert a string of genomic coordinates to a GRanges object

## Usage

```
coordToGRanges(coord)
```

## Arguments

| | |
|---|---|
| coord | String coordinate in genome browser-friendly format to convert to a GRanges object |

## Value

GRanges object

## Examples

```
string <- "chr1:1000-2000"
coordToGRanges(string)
```

filterPeaks *filterPeaks*

## Description

Given a set of peak calls for different marks and samples, filter peaks according to values in numeric

## Usage

```
filterPeaks(peaks, columns, thresholds)
```

## Arguments

| | |
|---|---|
| peaks | List of GRanges objects storing peak calls for each sample, where element names correspond to sample IDs |
| columns | Character vector of column names containing stats by which to filter peaks |
| thresholds | Vector of numeric values giving the lower thresholds to use for each of the columns specified, in the same order as columns |

## Value

A list of GRanges objects storing peak calls for each sample, with peaks filtered according to the columns and thresholds specified.

## Examples

```
filterPeaks(peaks = H3K4me3,
    columns = c("signalValue", "pValue"),
    thresholds = c(4, 10))
```

---

`GRangesToCoord` *GRangesToCoord*

---

## Description

Convert a GRanges object for one region to a genome browser-friendly string

## Usage

```
GRangesToCoord(gr)
```

## Arguments

| | |
|---|---|
| `gr` | GRanges object specifying region to convert to a string |

## Value

String

## Examples

```
gr <- GRanges(seqnames = "chr1",
              ranges = IRanges(start = 1000, end = 2000))

GRangesToCoord(gr)
```

---

`H3K4me3` *H3K4me3 peak calls in a short region for six adult tissues*

---

## Description

A toy dataset containing MACS2 narrow peak calls for 3 brain tissues and 3 other adult tissues from the Roadmap Epigenomics Project, restricted to a short region on chromosome 19. The generation of this dataset is executed by the script in the "data-raw" directory of this package, which can be viewed at https://github.com/selinj/chromswitch/tree/master/data-raw.

## Usage

```
H3K4me3
```

## Format

A list with six entries, named according to IDs of the samples. Each element contains a GRanges object with peak calls and associated statistics which are computed by MACS2. This is the format expected by the `peaks` argument in functions in chromswitch.

### Source

egg2.wustl.edu/roadmap/web_portal/

---

`homogeneity` *homogeneity*

---

### Description

Computes the homogeneity of a set of clusters given ground-truth classes, as defined in https://aclweb.org/anthology/D/D0
1043.pdf

### Usage

```
homogeneity(contingency, c, k)
```

### Arguments

| | |
|---|---|
| `contingency` | Table, contingency table between clusters and conditions as returned by the `table` function |
| `c` | Vector of classes |
| `k` | Vector of clusters |

### Value

Numeric

### Examples

```
clusters <- c(0, 0, 2, 1, 1, 0, 1)
classes <- c("A", "A", "A", "B", "B", "A", "B")
ct <- table(classes, clusters)
homogeneity(contingency = ct)
```

---

`LocalPeaks-class` *LocalPeaks*

---

### Description

The LocalPeaks class is a container for the peaks for one or more marks for a set of samples in a specific genomic region of interest, as well as the genomic region itself, and the sample IDs. These components are needed to convert sets of peaks into rectangular feature-by-sample matrices which we can then use for downstream analysis - and in particular, as input to a clustering algorithm in order to call a chromatin state switch.

## Usage

```
## S4 method for signature 'LocalPeaks'
region(x)
## S4 method for signature 'LocalPeaks'
samples(object)
## S4 method for signature 'LocalPeaks'
peaks(x)
```

## Arguments

| | |
|---|---|
| x | LocalPeaks object, as returned by retrievePeaks |
| object | LocalPeaks object, as returned by retrievePeaks |

## Value

LocalPeaks object

## Slots

region A GRanges object specifying one genomic region, the query region

peaks List of lists of GRanges objects. Each outer list stores peaks for each sample for one mark in region.

samples Character vector with sample identifiers.

## Methods

region: Access region slot of LocalPeaks object.

samples: Access samples slot of LocalPeaks object.

peaks: Access peaks slot of LocalPeaks object.

## Examples

```
# Assemble dataset
samples <- c("E068", "E071", "E074", "E101", "E102", "E110")
bedfiles <- system.file("extdata", paste0(samples, ".H3K4me3.bed"),
package = "chromswitch")

metadata <- data.frame(Sample = samples,
    H3K4me3 = bedfiles,
    stringsAsFactors = FALSE)

# Obtain a LocalPeaks object by retrieving the peaks in the query region
lpk <- retrievePeaks(H3K4me3,
    metadata = metadata,
    region = GRanges(seqnames = "chr19",
    ranges = IRanges(start = 54924104, end = 54929104)))

# lpk now stores the query region, samples, and associated peaks overlapping
# the query region

# Get the samples from the object
samples(lpk)
```

```
# Get the query region associated with the object
region(lpk)

# Get the set of peaks in each sample which overlap with the query region
peaks(lpk)
```

---

makeBrowserCoord *makeBrowserCoord*

---

## Description

Given coordinates for a genomic region, return a browser-friendly version.

## Usage

```
makeBrowserCoord(chr, start, end)
```

## Arguments

| | |
|---|---|
| chr | The chromosome |
| start | The starting position of the genomic region |
| end | The ending position of the genomic region |

## Value

String with copy-pastable, genome browser-friendly version of coordinates.

## Examples

```
makeBrowserCoord("chr1", 1000, 2000)
```

---

NMI *NMI*

---

## Description

Computes the Normalized Mutual Information betwen two partitions

## Usage

```
NMI(clusters, classes)
```

## Arguments

| | |
|---|---|
| clusters | A vector of cluster assignments |
| classes | A vector giving the true classes of the objects |

## Details

This code comes directly from the package 'clue': https://github.com/cran/clue/blob/098da43010f3803294b4e8R/agreement.R#L161

Hornik K (2017). _clue: Cluster ensembles_. R package version 0.3-53, <URL: https://CRAN.R-project.org/package=clue>.

Hornik K (2005). "A CLUE for CLUster Ensembles." _Journal of Statistical Software_, *14*(12). doi: 10.18637/jss.v014.i12 (URL: http://doi.org/10.18637/jss.v014.i12).

## Value

Numeric

## Examples

```
clusters <- c(0, 0, 2, 1, 1, 0, 1)
classes <- c("A", "A", "A", "B", "B", "A", "B")
NMI(clusters, classes)
```

---

`normalizePeaks` *normalizePeaks*

---

## Description

Given a set of peak calls for different marks and samples, normalize all peaks genome-wide for each sample and mark by rescaling and Winsorizing, i.e. rescale the middle of the data to the range [0, 1] and bound the upper tail to 1 and the lower tail to 0, effectively replacing a fixed amount of extreme values in each tail. Similar to trimming the tails except instead of discarding the tails entirely they're bounded.

## Usage

```
normalizePeaks(peaks, columns, tail = 0.005)
```

## Arguments

| | |
|---|---|
| `peaks` | List of GRanges objects storing peak calls for each sample, where element names correspond to sample IDs |
| `columns` | Character vector specifying the names of columns to normalize |
| `tail` | Optional: numeric, a fraction in [0, 1] specifying how much of the data to bound to 0 (for the lower tail) or 1 (for the upper tail). Default: 0.005. |

## Value

A list of GRanges objects storing peak calls for each sample, with columns specified in `columns` normalized.

## See Also

winsorNorm

### Examples

```
normalizePeaks(H3K4me3, columns = c("signalValue", "pValue", "qValue"))
```

---

`pReciprocalOverlap` *pReciprocalOverlap*

---

### Description

If a and b denote two genomic regions, check whether they overlap reciprocally by p*100

### Usage

```
pReciprocalOverlap(a, b, p)
```

### Arguments

| | |
|---|---|
| a | GRanges object storing first region |
| b | GRanges object storing second region |
| p | Numeric value in [0, 1] giving the fraction of reciprocal overlap to require. |

### Value

Logical value, TRUE if a and b are the same by having a p-reciprocal overlap, FALSE otherwise

### Examples

```
a <- GRanges(seqnames = "chr11",
             ranges = IRanges(start = 112829468, end = 112834468))
b <- GRanges(seqnames = "chr11",
             ranges = IRanges(start = 112829468, end = 113834468))

pReciprocalOverlap(a, b, 0.9)
```

---

`purity` *purity*

---

### Description

Computes the purity of a partition as defined in https://www.ncbi.nlm.nih.gov/pubmed/17483501

### Usage

```
purity(contingency, c, k)
```

### Arguments

| | |
|---|---|
| contingency | Table, contingency table between clusters and conditions as returned by the table function |
| c | Vector of classes |
| k | Vector of clusters |

### Value

Numeric

### Examples

```
clusters <- c(0, 0, 2, 1, 1, 0, 1)
classes <- c("A", "A", "A", "B", "B", "A", "B")
ct <- table(classes, clusters)
purity(contingency = ct)
```

---

## readNarrowPeak *readNarrowPeak*

---

### Description

A helper function for reading in narrow peak calls for a set of samples. Peak calls are assumed to be in ENCODE narrowPeak format (https://genome.ucsc.edu/FAQ/FAQformat.html#format12) as returned by MACS2 (http://liulab.dfci.harvard.edu/MACS/). This is BED6+4 format.

### Usage

```
readNarrowPeak(paths, metadata)
```

### Arguments

| | |
|---|---|
| paths | Character vector storing paths for BED files containing peak calls for each sample, in the same order as in the Sample column of metadata. |
| metadata | A dataframe with at least two columns: "Sample" which stores the sample identifiers, and "Condition" which stores the biological condition labels of the samples. |

### Value

Named list of GRanges objects containing peak calls for each sample.

## Examples

```
samples <- c("E068", "E071", "E074", "E101", "E102", "E110")
bedfiles <- system.file("extdata", paste0(samples, ".H3K4me3.bed"),
package = "chromswitch")
Conditions <- c(rep("Brain", 3), rep("Other", 3))

metadata <- data.frame(Sample = samples,
    H3K4me3 = bedfiles,
    Condition = Conditions,
    stringsAsFactors = FALSE)

readNarrowPeak(bedfiles, metadata)
```

---

`reducePeaks` *reducePeaks*

---

## Description

Given a LocalPeaks object, merge peaks which are in the same sample and are separated by no more than gap base pairs. When two non-overlapping peaks are merged, a new peak is created which starts at the starting position of the first peak and ends at the ending position of the second peak, spanning the range of both peaks and the gap between them.

## Usage

```
reducePeaks(localpeaks, gap)
```

## Arguments

| | |
|---|---|
| `localpeaks` | LocalPeaks object |
| `gap` | Numeric value, specifying the threshold distance for merging. Peaks in the same sample which are within this many bp of each other will be merged. |

## Value

The LocalPeaks object that was provided as input, with nearby peaks merged

## Examples

```
samples <- c("E068", "E071", "E074", "E101", "E102", "E110")
bedfiles <- system.file("extdata", paste0(samples, ".H3K4me3.bed"),
package = "chromswitch")

metadata <- data.frame(Sample = samples,
    H3K4me3 = bedfiles,
    stringsAsFactors = FALSE)

lpk <- retrievePeaks(H3K4me3,
    metadata = metadata,
    region = GRanges(seqnames = "chr19",
    ranges = IRanges(start = 54924104, end = 54929104)))
```

```
reducePeaks(lpk, gap = 300)
```

---

`retrievePeaks` *retrievePeaks*

---

## Description

Given a peak calls for a set of samples, for each sample, get the peaks which overlap a specified genomic region of interest. Typically, this corresponds to the region for which we will construct a feature matrix representing peaks in the region in order to call a chromatin state switch.

## Usage

```
retrievePeaks(peaks, metadata, region)
```

## Arguments

| | |
|---|---|
| `peaks` | List of GRanges objects storing peak calls for each sample |
| `metadata` | Dataframe with a column "Sample" which stores the sample identifiers, and at least one column, titled by the histone mark or ChIP-seq target, storing paths to the BED files containing peak calls |
| `region` | GRanges object specifying one genomic region, the query region |

## Value

LocalPeaks object as described in `LocalPeaks`

## Examples

```
samples <- c("E068", "E071", "E074", "E101", "E102", "E110")
bedfiles <- system.file("extdata", paste0(samples, ".H3K4me3.bed"),
package = "chromswitch")

metadata <- data.frame(Sample = samples,
    H3K4me3 = bedfiles,
    stringsAsFactors = FALSE)

retrievePeaks(H3K4me3,
    metadata = metadata,
    region = GRanges(seqnames = "chr19",
    ranges = IRanges(start = 54924104, end = 54929104)))
```

summarizePeaks *summarizePeaks*

## Description

Given peaks for a set of samples in a query region, construct a sample-by- feature matrix where each row is a vector of summary statistics computed from peaks in the region.

## Usage

```
summarizePeaks(localpeaks, mark, cols, fraction = TRUE, n = FALSE)
```

## Arguments

| | |
|---|---|
| localpeaks | LocalPeaks object |
| mark | String specifying the name of the mark for which the LocalPeaks object is given |
| cols | Character vector of column names on which to compute summary statistics |
| fraction | Loogical: compute the fraction of the region overlapped by peaks? |
| n | Logical: compute the number of peaks in the region? |

## Value

A matrix where rows are samples and columns are features

## Examples

```
samples <- c("E068", "E071", "E074", "E101", "E102", "E110")
bedfiles <- system.file("extdata", paste0(samples, ".H3K4me3.bed"),
package = "chromswitch")

metadata <- data.frame(Sample = samples,
    H3K4me3 = bedfiles,
    stringsAsFactors = FALSE)

lpk <- retrievePeaks(H3K4me3,
    metadata = metadata,
    region = GRanges(seqnames = "chr19",
    ranges = IRanges(start = 54924104, end = 54929104)))

summarizePeaks(lpk, mark = "H3K4me3", cols = c("qValue", "signalValue"))
```

vMeasure *vMeasure*

## Description

Computes the V measure of a set of clusters given ground-truth classes, as defined in https://aclweb.org/anthology/D/D07/1043.pdf

## Usage

```
vMeasure(contingency, c, k)
```

## Arguments

| | |
|---|---|
| contingency | Table, contingency table between clusters and conditions as returned by the table function |
| c | Vector of classes |
| k | Vector of clusters |

## Value

Numeric

## Examples

```
clusters <- c(0, 0, 2, 1, 1, 0, 1)
classes <- c("A", "A", "A", "B", "B", "A", "B")
ct <- table(classes, clusters)
vMeasure(contingency = ct)
```

winsorNorm *winsorNorm*

## Description

Normalize a numeric vector by rescaling and Winsorizing, i.e. rescale the middle of the data to the range [0, 1] and bound the upper tail to 1 and the lower tail to 0, effectively replacing a fixed amount of extreme values in each tail. Similar to trimming the tails except instead of discarding the tails entirely they're bounded.

## Usage

```
winsorNorm(x, trim)
```

## Arguments

| | |
|---|---|
| x | A numeric vector, the data to be normalized |
| trim | Numeric, a fraction in [0, 1] specifying how much of the data to bound to 0 (for the lower tail) or 1 (for the upper tail) |

## Value

Numeric vector

## Examples

```
x <- seq(1, 100, by = 1)
x

# Bound the lower and upper 5% of values in the vector
winsorNorm(x, trim = 0.05)
```

# Index