Package 'FunChIP'

April 15, 2024

Type Package

Title Clustering and Alignment of ChIP-Seq peaks based on their shapes

Version 1.28.0

Date 2021-11-20

Depends R (>= 3.2), GenomicRanges

Imports shiny, fda, doParallel, GenomicAlignments, Rcpp, methods, foreach, parallel, GenomeInfoDb, Rsamtools, grDevices, graphics, stats, RColorBrewer

Author Alice Parodi [aut, cre], Marco Morelli [aut, cre], Laura M. Sangalli [aut], Piercesare Secchi [aut], Simone Vantini [aut]

Maintainer Alice Parodi <alicecarla.parodi@polimi.it>

biocViews StatisticalMethod, Clustering, ChIPSeq

Description Preprocessing and smoothing of ChIP-Seq peaks and efficient implementation of the k-mean alignment algorithm to classify them.

NeedsCompilation yes

LinkingTo Rcpp

License Artistic-2.0

LazyData TRUE

git_url https://git.bioconductor.org/packages/FunChIP

git_branch RELEASE_3_18

git_last_commit ee9b6c9

git_last_commit_date 2023-10-24

Repository Bioconductor 3.18

Date/Publication 2024-04-15

R topics documented:

`FunChIP-package` *Clustering and Alignment of ChIP-Seq peaks based on their shapes*

Description

Efficient implementation of the k-mean alignment algorithm to classify spline-smoothed ChIP-Seq peaks.

Details

Package:	FunChIP
Type:	Package
Version:	0.99.4
Date:	2016-07-07
License:	GPL-3

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

References

Sangalli, L.M., Secchi, P., Vantini, S., Vitelli, V., 2010. *"K-mean alignment for curve clustering"*. Computational Statistics and Data Analysis, 54, 1219-1233.

See Also

`pileup_peak`, `smooth_peak`, `summit_peak`, `distance_peak`, `cluster_peak`, `choose_k`, `plot_peak`

bending_index *The elbow rule to define the proper number of clusters.*

Description

Given a GRanges object with metadata columns related to the classification performed with the cluster_peak method, this function quantifies the elbow rule. See Details for a short presentation of the method and the Vignette of the package for a complete defintion of the index.

Usage

```
bending_index(object, plot.graph.k = FALSE)
```

Arguments

`object`	GRanges object. It must contain the metadata columns associated to the classification to be analyzed. Specifically it must contain the `cluster_NOshift` metadata (and the correspondent set of distances `distance_NOshift`) if the user wants to compute the bending index for the non aligned peaks and/or the `cluster_shift` metadata (and the correspondent set of distances `distance_shift`) if the user wants to compute the bending index for the classification with alignment.
`plot.graph.k`	logical. If `TRUE` the graph of the global distance between the data and corresponding center of the cluster, varying the number of clusters is plotted. Distances are normalized with the total number of peaks n. These are the distances uesd to compute the bending index, as presented in Details. If `object` contains both the results with and without the classification, two lines are drown to show, beside the variation of the distance with an increase of k, also the decrease of the global distance introduced by the alignment procedure. If a single classification is stored in the `object`, only one line is drawn. Default is `FALSE`.

Details

This function consists of the computation for each feasible value of k (from 2 to K ??? 1, with K the maximum number of clusters) of an index that quantifies the magnitude of the elbow. As higher is this index, as the correspondent value of k is meaningful. Specifically it is computed as the distance of the point in k of the global distance function (normalized with the maximum value it assumes) from the line passing by the point in k ??? 1 and in k + 1. For further details, see the Vignette.

Value

The function returns

- a data.frame (or a list with two data.frames, in case of `object` with classification with and without alignment) containing the bending index for different values of the parameter k.
- if `plot.graph.k = TRUE` the graphical representation of the distances (normalized with the total number of peaks n), varying the classification type and the number of clusters.

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

Examples

```
# load the data
data(peaks)

# compute the bending index
index <- bending_index(peaks.data.cluster, plot.graph.k = FALSE)
# from the analysis of this results, a choice of k=3 for
# the classification with shift and k=2 for the classification
# without shift is suggested.
```

---

choose_k *Choice of the final classification of peaks*

Description

Selection of the final classification of the peaks, given the desired number of clusters and the presence or absence of the alignment procedure chosen. This choice is usually driven by the graph returned by cluster_peak.

Usage

```
## S4 method for signature 'GRanges'
choose_k(object, k = NULL, shift.peak = NULL, cleaning = TRUE)
```

Arguments

object	GRanges object. It must contain the metadata columns associated to the chosen classification. Further details are provided in shift.peak.
k	integer. Number of chosen clusters.
shift.peak	logical. If TRUE, the clustering with alignment is chosen, if FALSE, the classification without alignment is selected. If shift.peak = TRUE, object must contain the metadata columns labels_shift, coef_shift, dist_shift; if shift.peak = FALSE it must contain the columns labels_NOshift and dist_NOshift.
cleaning	logical. If TRUE, all metadata columns generated by FunChIP on the GRanges object are removed, and one new column containing the classification result is added: cluster. If FALSE, the metadata columns generated by FunChIP are kept. Default is TRUE. See Value for further details on the metadata column added.

Details

The choice of the optimal number of clusters and the presence of alignment can be guided by the graph plotted in the cluster_peak method. In particular, for the optimal number of clusters k the distance significantly decreases with respect to the lower values of k, and negligibly increases with respect to higher values of k (elbow in the line). The introduction of the alignment leads to better clustering of the data if if the global distance is significantly lowered. registration.

Value

if `cleaning = FALSE`, the GRanges `object` with a new metadata column:

- `cluster` integer. The label correspondent to the classification chosen

If `cleaning = TRUE`, all the metadata columns added through all the analysis are removed and `object` is returned with just the metadata column `cluster`

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

See Also

cluster_peak

Examples

```
# load the data
data(peaks)

# k = 3 clusters with the alignment
# with integer shifts are chosen

peaks.classified <- choose_k(peaks.data.cluster, k = 3,
shift.peak = TRUE, cleaning = FALSE)
```

`cluster_peak` *Clustering the peaks with the k-mean alignment algorithm*

Description

It classifies and aligns the peaks stored in the GRanges object. The method applies the k-mean alignment algorithm with shift of the peaks and distance based on the convex combination of the L^p distances between the spline-smoothed peaks and their derivatives. The order p can be one of 1, 2 and ∞.

Usage

```
## S4 method for signature 'GRanges'
cluster_peak(object, parallel = FALSE, num.cores = NULL,
    n.clust = NULL,  seeds = NULL, shift.peak = NULL, weight = NULL,
    subsample.weight = 100, alpha = 1, p = 1, t.max = 0.5,
    plot.graph.k = TRUE, verbose = TRUE, rescale = FALSE )
```

Arguments

`object`	GRanges object of length N. It must contain the metadata columns `spline`, `spline_der`, `width_spline`, computed by smooth_peak.
`parallel`	logical. If `TRUE`, the clustering for different values of the parameter k in `n.clust` are run in parallel. Default is `FALSE`.
`num.cores`	integer. If `parallel` is `TRUE`, it indicates the number of cores used in the parallelization. If `NULL` (default), the number of cores is automatically identified.
`n.clust`	integer vector (or scalar). Number of clusters in which the data set is divided (possibly one, if `n.clust` is a scalar). For each value of the vector, the cpp function `kmean_function` is called.
`seeds`	vector. Indices of the initial centers of the clusters, needed to initialize the k-mean procedure. The k-mean alignment, like all the k-mean-like algorithms, is dependent on the choice of the initial centers of the clusters, and each initialization of the seeds can generate slightly different results. The values must be included in $1, \ldots, N$. The length of the vector must be equal to the maximum number of clusters analyzed (`max(n.clust)`), otherwise it is truncated to this value, or the missing values are randomly generated. If `NULL` (default), the seeds are detected as the most central values (i.e. peaks with minimum distance from the others) of the set of peaks. If `seeds='random'`, the centers are randomly generated.
`shift.peak`	logical. It indicates whether the alignment via a translation of the abscissae is performed (`shift.peak = TRUE`) or not (`shift.peak = FALSE`). If no value is provided (`shift.peak = NULL`, default), both analyses are performed.
`weight`	real. Weight w of the distance function (see Details for the definitions of the distance function), needed to make the distance between splines and derivatives comparable. If no value is provided (default is `NULL`), it is computed as the median of the ratio between the pairwise distances of the data ($d_0(i,j)$) and of the derivatives ($d_1(i,j)$) $$w = \text{median}\frac{d_0(i,j)}{d_1(i,j)}$$ with $i, j = 1 : \ldots N$.
`subsample.weight`	integer value. Number of data points used to define the `weight`, if not assigned. Using all the peaks to define the weight can be computationally expensive and therefore a subsampling is suggested. If `subsample.weight=NULL` all the data will be used. Default is 100, which is a reasonable trade off between running time and reliability of the estimation.

`alpha` real value between 0 and 1. Value of the convex weight α of the distance to balance the distance between data and derivatives. See details for the definition. Default is 1.

`p` integer value in $\{0, 1, 2\}$. Order of the L^p distance used. In particular `p = 0` stands for the L^∞ distance, `p = 1` for L^1 and `p = 2` for L^2

`t.max` real value. It tunes the maximum shift allowed. In particular the maximum shift at each iteration is computed as

$$\text{max\_shift} = \texttt{t.max} * \text{range}(\texttt{object})$$

and the optimum registration coefficient will be identified between - max_shift and + max_shift. range(`object`) is the maximum amplitude of the peaks. Default is 0.5.

`plot.graph.k` logical. If `TRUE` the graph of the average distance between the data and corresponding center of the cluster, varying the number of clusters is plotted. If `align=NULL`, both the analysis with and without alignment are performed, two lines are drown to show the decrease of the global distance introduced by the alignment procedure. Default is `TRUE`

`verbose` logical. If `TRUE`, some parameters of the algorithm and the progress of the iterations are shown, if `FALSE` no information is provided. Default is `TRUE`, but consider to set the parameter to `FALSE` in case of parallel runs, to avoid the overlap of their outputs.

`rescale` logical. If `TRUE` clustering is performed on *scaled peaks*. For the definition of *scaled peaks* see smooth_peak.

Details

See [Sangalli et al., 2010] and the package vignette for the complete description of the algorithm. The algorithm is completely defined once we fix the family of the warping function for the alignment and the distance function. In this function we focus only on the specific case of

- warping functions: shifts with integer coefficients

$$h(t) = t + c,$$

with c an integer value;

- distance: convex combination of the L^p distance between data and derivatives. The distance between f and g is

$$d(f, g) = (1 - \alpha)\|f - g\|_p + \alpha\ w\|f' - g'\|_p$$

The choice of $\|\cdot\|_p$ corresponds to the value of `p` in input. In particular `p = 0` stands for $\|\cdot\|_{L^\infty}$, `p = 1` for $\|\cdot\|_{L^1}$ and `p = 2` for $\|\cdot\|_{L^2}$

Value

the GRanges `object` with new metadata columns:

- if `align` is `TRUE` or `NULL`, i.e. the clustering with alignment is performed the following metadata columns are added:

  - `cluster_shift`: for each peak, a vector of length equal to the maximum number of chosen clusters, containing at each position k the label of the cluster the peak is assigned to, when the total number of clusters is k and alignment is performed during the clustering. If k is not present in the `n.clust` vector, the corresponding value is `NA`.
  - `coef_shift`: for each peak, a vector of length equal to the maximum number of chosen clusters, containing at each position k the shift coefficient assigned to the peak, when the total number of clusters is k and alignment is performed during clustering. If k is not present in the vector `n.clust` the corresponding value is `NA`.
  - `dist_shift`: for each peak, a vector of length equal to the maximum number of chosen clusters, containing at each position k the distance of the specific peak from the corresponding center of the cluster, when the total number of clusters is k and alignment is performed during clustering. If k is not present in the vector `n.clust` the corresponding value is `NA`.

- if `shift.peak` is `FALSE` or `NULL`, i.e. clustering is performed without alignment, the following metadata columns are added:
  - `cluster_NOshift`: for each peak, a vector of length equal to the maximum number of chosen clusters, containing at each position k the label of the cluster the peak is assigned to, when the total number of clusters is k and no alignment is performed during clustering. If k is not present in the vector `n.clust` the corresponding value is `NA`.
  - `dist_NOshift`: for each peak, vector of length equal to the maximum number of chosen cluster, containing at each position k the distance of the peak from the corresponding center of the cluster , when the total number of clusters is k and no alignment is performed during clustering. If k is not present in the vector `n.clust` the corresponding value is `NA`.

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

References

Sangalli, L. M., Secchi, P., Vantini, S. and Vitelli, V., 2010. K-mean alignment for curve clustering. Computational Statistics and Data Analysis, 54 1219 - 1233.

See Also

choose_k

Examples

```
# load the data
data(peaks)

# cluster and align the data as a
# function of the
# number of cluster (from 1 to 5)
# with and without alignment.
# The automathically generated plot
# can be used to detect the
# optimal number of clusters and the
```

```
# classification method to be used
# (with or without alignment)

clustered_peaks <- cluster_peak ( peaks.data.summit, parallel = FALSE ,
                                  n.clust = 1:5, shift.peak = NULL,
                                  weight = 1, alpha = 1, p = 2,
                                  plot.graph.k = TRUE, verbose = TRUE )
```

`compute_fragments_length` *Computing the length of the fragments in the .bam file.*

Description

Given a .bam file and GRanges object, it computes the positive and negative coverage for each GRanges element, estimates the distance between positive and negative peaks, and finally the fragment length d, i.e. the sum of the length of the reads and the distance between positive and negative peaks. See Details and the package vignette for the description of the method.

Usage

```
compute_fragments_length(object, bamf, min.d = 0, max.d = 200)
```

Arguments

`object`	GRanges object of length N.
`bamf`	Path to the .bam file used to compute the coverage function. The associated .bam.bai index file must also be present.
`min.d`	integer. Minimum value for the distance between positive and negative peaks. Default is 0.
`max.d`	integer. Maximum value for the distance between positive and negative peaks. Default is 200.

Details

Given a set of $n = 1, \dots N$ regions, characterized by their positive and negative coverages, the function computes the distance between the positive peak f_{n+} and the negative peak, shifted by δ f^{δ}_{n-}:

$$D(f_{n+}, f^{\delta}_{n-}) = \frac{\|f_{n+} - f^{\delta}_{n-}\|^2_{L^2}}{\text{width}(\text{union}(f_{n+}, f^{\delta}_{n-}))}$$

The function computes the d_{pn} minimizing the distance between postive and negative peaks

$$d_{pn} = \text{argmin}_{\delta \in [\texttt{min.d},\texttt{max.d}]} \sum_{n=1}^{N} D(f_{n+}, f^{\delta}_{n-})$$

The function returns both the plot of the global distance vs the fragment length $d = d_{pn} + r$, where r is the length of the reads, and the optimum value for d. r is also estimated from the .bam file as the average of the read lengths.

Value

optimum value of the parameter d, to be used in the pileup_peak method.

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

Examples

```
# load the data
# GRanges object

data(GR100)

# import the .bam file

bamf <- system.file("extdata", "test.bam", package="FunChIP",
                    mustWork=TRUE)

# compute the estimated fragment length

d <- compute_fragments_length(GR[1:10], bamf, min.d = 0, max.d = 200)
```

`distance_peak` *Computing distance matrices among spline-smoothed peaks and among their derivatives.*

Description

Given a metric (L^1, L^2 or L^∞), it computes the pairwise distance among the spline approximation of the peaks and among their derivatives. If summit is provided, peaks are centered around the summit to compute the distances.

Usage

```
distance_peak(object, p = 1, rescale = FALSE)
```

Arguments

`object`	GRanges object of length N. It has to contain the metadata columns `spline`, `spline_der`, `width_spline`.
`p`	integer. It must assume values in `{0, 1, 2}`. They correspond respectively to the L^∞, L^1 and L^2 norm. See details for the definition of the norms.
`rescale`	logical. If `TRUE` the distance among *scaled peaks* is computed. For the definition of *scaled peaks* see smooth_peak.

Details

This function computes the pairwise distance of a set of N peaks. Given the spline-smoothed peaks s_i and their derivatives s_i' ($i = 1, \ldots, N$), it returns two matrices `dist_matrix_d0` and `dist_matrix_d1` whose elements (i, j), with $i, j = 1, \ldots, N$, are

$$\texttt{dist\_matrix\_d0}(i,j) = \|s_i - s_j\|_p \quad \texttt{dist\_matrix\_d1}(i,j) = \|s_i' - s_j'\|_p$$

.

In particular, in order to define the distance between two functions f and g:

- define a common domain U, given by the union of the domains of f and g. If a function is not defined on the whole domain, it is extended with 0's on the missing parts. The value of 0 is chosen because the background of the peaks has been removed during the definition of the splines, and hence they can now be continuously extended with 0's.
- choose the order of the norm `p`. Given the function f defined on U we have:

$$\|f\|_0 = \|f\|_{L^\infty} = \max_{x \in U} |f(x)|$$

,

$$\|f\|_1 = \|f\|_{L^1} = \int_U |f(x)| dx$$

,

$$\|f\|_2 = \|f\|_{L^2} = \sqrt{\int_U (f(x))^2 dx}$$

.

Value

list with two components

`dist_matrix_d0`
: N x N matrix of the pairwise distances between the splines.

`dist_matrix_d2`
: N x N matrix of the pairwise distances between the derivatives of splines.

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

Examples

```
# load the data
data(peaks)

# Compute the pairwise
# L2 distance between the peaks
dist_matrices <- distance_peak(peaks.data.summit, p = 2)
```

GR100 *Peaks of a ChIP-seq experiment*

Description

The data-set GR100 contains 100 enriched regions on chromosome 18 of the transcription factor c-Myc on murine cells, obtained by calling peaks with MACS [Zhang et al., 2008] on a ChIP-Seq experiment. The genomic coordinates of the peaks are stored in the GRanges object GR100.

Usage

```
data("GR100")
```

Format

GRanges object with 100 ranges and 0 metadata columns.

References

Zhang et al., 2008. Model-based Analysis of ChIP-Seq (MACS). Genome Biology, vol. 9 (9) pp. R137.

Examples

```
data(GR100)
GR
```

peaks *Coverage of a ChIP-Seq experiment*

Description

It contains a data-set used in all the examples of the FunChIP, together with all the metadata columns generated in the intermediate steps.

Usage

```
data("peaks")
```

Format

Several data-set are included

- peaks.data: a GRanges object with 10 ranges and 1 metadata column:
  - `counts`. A vector for each range, with length equal to the width of the range, containing the coverage of the range, i.e. the base-level read counts. It can be computed with the pileup_peak method.
- peaks.data.smooth: a GRanges object with 10 ranges and 6 metadata columns:
  - `counts`. As in `peaks.data`.
  - `spline`. A vector for each range, containing the evaluation of the spline approximation of the peak for each genomic base. It can be computed with the smooth_peak method.
  - `spline_der`. A vector for each range, containing the evaluation of the derivatives of the spline approximation of the peak for each genomic base. It can be computed with the smooth_peak method.
  - `width_spline`. Integer. The number of evaluated points of the spline approximation, i.e. the number of non-zero points, for each range. It can be computed with the smooth_peak method.
  - `start_spline`. Integer. The starting point of the spline approximation. It could be smaller than `start(object)` since the approximation can increase the values at the border to make the curve smooth. It can be computed with the smooth_peak method.
  - `end_spline`. Integer. The end point of the spline approximation. It could be larger than `end(object)` since the approximation can increase the values at the border to make the curve smooth. It can be computed with the smooth_peak method.
- peaks.data.smooth.scaled: a GRanges object with 10 ranges and, beside the 6 metadata columns of `peaks.data.smooth`, 2 more coumns
  - `spline_rescaled`. A vector for each range, containing the evaluation of the scaled spline approximation on the common grid of all the peaks.
  - `spline_der_rescaled`. A vector for each range, containing the evaluation of the derivatives of the scaled spline approximation on the common grid.
- peaks.data.summit: a GRanges object with 10 ranges and 7 metadata columns:
  - `counts`. As `peaks.data`.
  - `spline`. As `peaks.data.smooth`.
  - `spline_der`. As `peaks.data.smooth`.
  - `width_spline`. As `peaks.data.smooth`.
  - `start_spline`. As `peaks.data.smooth`.
  - `end_spline`. As `peaks.data.smooth`.
  - `summit_spline`. The distance from the starting point of the spline of the maximum point (integer) of the spline (or the summit of the peak), for each range. It can be computed with the summit_peak method
- peaks.data.summit.scaled: a GRanges object with 10 ranges and, beside the 7 metadata columns of `peaks.data.summit`, 3 more coumns
  - `spline_rescaled`. As `peaks.data.smooth.scaled`.
  - `spline_der_rescaled`. As `peaks.data.smooth.scaled`.

  - summit_spline_rescaled. The distance from the starting point of the scaled spline of the maximum point (integer) of the scaled spline (or the summit of the peak), for each range. It can be computed with the summit_peak method setting to TRUE the rescale argument.

- peaks.data.cluster: a GRanges object with 10 ranges and 12 metadata columns:
  - counts. As peaks.data.
  - spline. As peaks.data.smooth.
  - spline_der. As peaks.data.smooth.
  - width_spline. As peaks.data.smooth.
  - start_spline. As peaks.data.smooth.
  - end_spline. As peaks.data.smooth.
  - summit_spline. As peaks.data.summit.
  - cluster_NOshift. A vector of length 5 for each range, containing the label of the cluster assigned to the peak in case of clustering without alignment. For example, the second element of the vector is the label of the corresponding peak when the k-mean alignment algorithm is run with 2 clusters. It can be computed with the cluster_peak method with n.clust = 1:5 and shift.peak=FALSE.
  - dist_NOshift. A vector of length 5 for each range, containing the distance from the center of the cluster assigned to the peak in case of clustering without alignment. For example, the second element of the vector is the distance of the corresponding peak from center of the corresponding cluster when the k-mean alignment algorithm is run with 2 clusters. It can be computed with the cluster_peak method with n.clust = 1:5 and shift.peak=FALSE.
  - cluster_shift. A vector of length 5 for each range, containing the label of the cluster assigned to the peak in case of clustering with alignment. For example, the second element of the vector is the label of the corresponding peak when the k-mean alignment algorithm is run with 2 clusters. It can be computed with the cluster_peak method with n.clust = 1:5 and shift.peak=TRUE.
  - coef_shift. A vector of length 5 for each range, containing the optimal shift coefficient of the peak. For example, the second element of the vector is the shift coefficient of the corresponding peak when the k-mean alignment algorithm is run with 2 clusters. It can be computed with the cluster_peak method with n.clust = 1:5 and shift.peak=TRUE.
  - dist_shift. A vector of length 5 for each range, containing the distance from the center of the cluster assigned to the peak in case of clustering with alignment. For example, the second element of the vector is the distance of the corresponding peak from the center of the corresponding cluster when the k-mean alignment algorithm is run with 2 clusters. It can be computed with the cluster_peak method with n.clust = 1:5 and shift.peak=TRUE.

- peaks.data.cluster.scaled: a GRanges object with 10 ranges and, beside the 12 metadata columns of peaks.data.cluster, 3 more coumns
  - spline_rescaled. As peaks.data.smooth.scaled.
  - spline_der_rescaled. As peaks.data.smooth.scaled.
  - summit_spline_rescaled. As peaks.data.summit.scaled.

  It is computed from peaks.data.summit.scaled with the cluster_peak method setting rescale = TRUE.

- peaks.data.classified: a GRanges object with 10 ranges and 13 metadata columns:
  - counts. As peaks.data.
  - spline. As peaks.data.smooth.
  - spline_der. As peaks.data.smooth.
  - width_spline. As peaks.data.smooth.
  - start_spline. As peaks.data.smooth.
  - end_spline. As peaks.data.smooth.
  - summit_spline. As peaks.data.summit.
  - cluster_NOshift. As peaks.data.cluster.
  - dist_NOshift. As peaks.data.cluster.
  - cluster_shift. As peaks.data.cluster.
  - coef_shift. As peaks.data.cluster..
  - dist_shift. As peaks.data.cluster.
  - cluster. Integer. The index of the final label assigned, for each range. It can be computed with the choose_k with k = 3 and shift.peak = TRUE.
- peaks.data.classified.scaled: a GRanges object with 10 ranges and, beside the 13 metadata columns of peaks.data.classified, 3 more columns
  - spline_rescaled. As peaks.data.smooth.scaled.
  - spline_der_rescaled. As peaks.data.smooth.scaled.
  - summit_spline_rescaled. As peaks.data.summit.scaled.

  It is computed from peaks.data.cluster.scaled with the choose_k method setting rescale = TRUE.

Examples

```
data(peaks)
```

---

pileup_peak *Computing read counts on a GRanges object.*

Description

Given a GRanges object and the path of a .bam file, it creates the corresponding pileup, containing the read counts on each nucleotide of the peaks of the GRanges object. Reads can be extended to a length d, which is an estimate of the length of the sequencing fragment. See the function compute_fragments_length for details. For each peak this method creates a vector containing these counts, i.e. the coverage function for the extended reads along the whole peak.

Usage

```
## S4 method for signature 'GRanges'
pileup_peak(object, bamf = NULL, d = NULL)
```

Arguments

object	GRanges object containing the genomic coordinates of the peaks.
bamf	Path to the .bam file used to compute the coverage function. The associated .bam.bai index file must also be present.
d	integer. Total length of the fragments. Positive and negative reads are extended in their 3' direction. Default is NULL; this value can be estimated by compute_fragments_length.

Value

the GRanges object with the new metadata column counts containing the coverage functions of the peaks.

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

Examples

```
# load the data
# GRanges object

data(GR)

# import the .bam file

bamf <- system.file("extdata", "test.bam", package="FunChIP",
                    mustWork=TRUE)

# extract the first 10 peaks of the GRange
# and compute the corresponding read counts
# with fragment length 160.

peaks <- pileup_peak(GR[1:10], bamf, d = 160)
```

---

plot_peak *Plotting the peaks.*

Description

It plots the peaks both as counts and splines (if provided). Peaks are centered around their summit. If the clustering has been performed, peaks can be aligned and divided in cluster, shown in different panels.

Usage

```
## S4 method for signature 'GRanges'
plot_peak(object, index = 1:length(object),
line.plot = "spline", col = NULL,
shift = NULL, k = NULL, cluster.peak = FALSE, rescale = FALSE,
lwd= 2, cex.axis = 1,  cex.lab = 1, cex.main = 1)
```

Arguments

object	GRanges object of length N. It must contain the metadata column counts. If it contains the metadata column summit_spline, peaks are plotted centered around their summits (i.e. the 0 of the abscissa is fixed as the summit of the peak). If summit is not provided peaks are not centered.
index	vector. Indices of the peaks to be plotted. Default is 1:length(object) to plot all the peaks.
line.plot	string. Type of plot. If 'spline' (default), the spline approximation of the peaks is plotted. If 'counts', only the raw data are plotted. If 'both', both the raw data and the approximation spline are plotted.
col	vector. Colors used to plot the peaks. If NULL, the rainbow color palette is used. If it has a single value, all the peaks are plotted with the same color. If it is a vector shorter than N, only the first element of col is used. Default is NULL.
shift	logical. This parameter controls the abscissae of the plotted peaks, and plays two different roles, depending on the cluster.peak parameter. \ If cluster.peak is FALSE, shift = TRUE means that peaks are plotted aligned around the summit_spline point. The GRanges object must contain the results of the summit_peak method. If cluster.peak is FALSE and shift = FALSE, peaks are plotted with no centering around the summit. If cluster.peak is FALSE and shift = NULL (default), peaks are centered, if the metadata column summit_spline is present in object, otherwise they are plotted with the original abscissae, with no centering around the summit. \ If cluster.peak = TRUE, the parameter shift sets the clustering result to be plotted. If TRUE the shift results are plotted, otherwise the results associated to NOshift clustering are presented.
k	integer. If cluster.peak = TRUE, results corresponding to k number of clusters are plotted. It must be set to a value included in the n.clust parameter of the correspondent object.
cluster.peak	logical. It FALSE, object contains the set of peaks to be plotted without classification (centered or not around the summit). If TRUE, the plot_peak method plots the result of the classification associated to shift and k parameters; in this case, object must be the output of the cluster_peak and must contain the correspondent classification. Default is FALSE.
rescale	logical. If TRUE the *scaled peaks* are plotted. Default is FALSE. If rescale = TRUE only spline approximations can be shown, then line.plot must be 'spline'.
lwd, cex.axis, cex.lab, cex.main	Optional graphical parameters.

Value

Graphical method to graphically represent data. No output returned,

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

Examples

```
# load the data
data(peaks)

# First example:
# plot of the spline approximation
# of the first 10 peaks
# centered around their summit

plot_peak(peaks.data.summit, index = 1:10,
          shift = TRUE)

# Second example:
# plot of the peaks
# divided in the k=3
# and shift = TRUE clusters
# obtained with the cluster_peak method
plot_peak(peaks.data.cluster,
shift = TRUE, k = 3, cluster.peak = TRUE)
```

---

silhouette_plot *Return the silhouette index for clustered peaks*

Description

It computes the silhouette index for peaks stored in a GRanges object and classified with the cluster_peak method. If the two classifications with a and without alignment are provided, this method computes the index for both these classifications.

Usage

```
silhouette_plot(object, p = 1,
        weight = NULL, alpha = 1,
        rescale = FALSE, t.max = 0.5)
```

Arguments

`object`	GRanges object. It must contain the metadata columns associated to the classification to be analyzed. Specifically it must contain the `cluster_NOshift` metadata if the user wants to compute the silhouette index for the non aligned peaks and/or the `cluster_shift` metadata if the user wants to compute the index for the classification with alignment.
`p`	integer value in {0, 1 , 2}. Order of the L^p distance used. In particular `p = 0` stands for the L^∞ distance, `p = 1` for L^1 and `p = 2` for L^2. Default is 1.
`weight`	real. Weight w of the distance function (see Details for the definition of the distance function), needed to make the distance between splines and derivatives comparable. It has no Default since it must be the same weight used to define the distance for the classification.
`alpha`	real value between 0 and 1. Value of the convex weight α of the distance to balance the distance between data and derivatives. See details for the definition. Default is 1.
`t.max`	real value. It tunes the maximum shift allowed. In particular the maximum shift at each iteration is computed as $$\text{max\_shift} = \texttt{t.max} * \text{range}(\texttt{object})$$ and the optimum registration coefficient will be identified between - max_shift and + max_shift. range(`object`) is the maximum amplitude of the peaks. Default is 0.5.
`rescale`	logical. If `TRUE` clustering is performed on *scaled peaks*. For the definition of *scaled peaks* see smooth_peak.

Details

See [Rousseeuw, 1987] for the detailed definition of the index. Specifically, for the peak i it is computed as

$$s(i) = \frac{a(i) - b(i)}{\max(a(i), b(i))}$$

with $a(i)$ the average dissimilarity of peak i with all other data within the same cluster and $b(i)$ the lowest average dissimilarity of i to any other cluster, of which i is not a member.

Value

The function returns

- the list of the silhouette indeces for the two classifications (if provided in the GRanges object) and for all the choices of the number of clusters
- the graphical representation of the silhouette index, varying the number of clusters and the classification. The average silhouette index is also presented.

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

References

Peter J. Rousseeuw (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. Computational and Applied Mathematics. 20: 53???65.

Examples

```
# load the data
data(peaks)

# computes the silhouette index and
# shows the graph
sil <- silhouette_plot(peaks.data.cluster, p=2, weight = 1, alpha = 1,
                          rescale = FALSE, t.max = 2)
```

---

smooth_peak *Spline smoothing of the peak*

Description

It approximates the read counts associated to every peak with a suitable B-spline function, so that a smoothing representation of the peaks is obtained. The first derivative of the spline is also computed. To obtain a smooth representation, the peak is extended and new initial and final points are identified. See the Vignette of the FunChIP package for a graphical representation of the spline approximation.

Usage

```
## S4 method for signature 'GRanges'
smooth_peak(object, n.breaks = 100, subsample = TRUE,
    subsample.data = 100, order = 4,
    lambda = (10^(seq(-5,5, by = 0.5))),
    GCV.derivatives = TRUE , plot.GCV = FALSE, rescale = FALSE)
```

Arguments

`object`	GRanges object. It must contain the metadata column `counts`.
`n.breaks`	integer. Number of breaks, or knots, for the B-spline basis domain definition. Default is `100`.
`subsample`	logical. If `TRUE`, only a random subset (of size fixed by the parameter `subsample.data`) is used to identify the optimal value of lambda for the penalization via cross-validation. If `subsample=FALSE`, all the peaks of the GRanges data will be used. To contain running times, it is suggested to maintain the default value `subsample = TRUE`.
`subsample.data`	integer. Number of data used for the cross-validation (if `subsample.data` is `TRUE`). Default value is 100. If `subsample = FALSE`, all data points will be used and `subsample.data` is ignored.

`order` integer. Order of the B-spline basis used for the smoothing. The order is one higher than the degree of the spline. Default is 4 (cubic splines).

`lambda` vector (or single value). Contains all the possible values of the smoothing parameter to be considered for the final choice. If a single value is provided, this will be automatically chosen for the smoothing. Default value is $10^{\texttt{seq(-5,5,by=0.5)}}$ to analyze a sufficiently wide set of values. See details below.

`GCV.derivatives` logical. If `TRUE` the Generalized Cross Validation index (GCV) on the derivatives is considered as criteria to identify λ, otherwise the GCV is computed on the data. Default is `TRUE`.

`plot.GCV` logical. If `TRUE`, the plot of the GCV of the data and derivatives is shown as a function of λ. Default value is `FALSE`.

`rescale` logical. If `TRUE` *scaled peaks* are also provided. From the spline approximation of the peak a new curve is defined. It is obtained scaling both the abscissa grid and the values of the coverages of the splines. All the *scaled peaks* have a common grid of width equal to the minimum width of the orignal splines and area equal to 1. Default is `FALSE`.

Details

It creates a piece-wise polynomial of fixed `order` s approximating the data (B-spline expansion, Ramsay and Silverman, 2005). Given the point wise defined function $f : (x, f(x))$, the `smooth_peak` method returns the evaluation of s on the x grid ($s(x)$) minimizing, for a fixed λ,

$$ERR(\lambda) = \|f - s\|^2_{L^2} + \lambda \|s''\|^2_{L_2}$$

, with s'' being the second derivative of the function s and $\|s\|_{L^2}$ the L^2 norm of the function, i.e. the integral on the domain of s of s^2.

The choice of λ is crucial for the definition of the spline, and it can be selected by minimizing the Generalized Cross-Validation index

$$GCV(\lambda) = \frac{nSSE}{(n - df(\lambda))^2}$$

, with SSE the error computed as

$$SSE = \|f - s\|^2_{L^2}$$

, if `GCV.derivatives = FALSE`, or

$$SSE = \|\nabla f - s'\|^2_{L^2}$$

, if `GCV.derivatives = TRUE`, and $df(\lambda)$ is the number of the degrees of freedom of the basis expansion automatically computed from s. For further details on the cross-validation procedure and on the computation of the number of degrees of freedom see Ramsay and Silverman, 2005.

If `plot.GCV` is `TRUE`, the plot of the GCV index as a function of λ is presented, which can be used to identify the optimal value of the parameter. If the plot is decreasing in λ, one could consider to increase the allowed values of λ to find the minimum of the curve.

Value

the GRanges `object` with new metadata columns:

- `width_spline` integer. Value containing the width of the smoothed peak, i.e. the number of non-zero values of the spline approximation. This value is not necessarily equal to the original width of the peak, as the approximation can stretch outside the original width of the peak: to ensure smoothness some 0 values can be introduced at the edges of the region.
- `spline` vector. Evaluation of the spline on the grid of size `width_spline`.
- `spline_der` vector. Evaluation of the derivatives of the spline on the grid of size `width_spline`.
- `start_spline` integer. Genomic coordinate of the initial point of the spline approximation.
- `end_spline` integer. Genomic coordinate of the final point of the spline approximation.

If `rescale` is `TRUE` two more metadata columns are added:

- `spline_rescaled` vector. Evaluation of the *scaled peaks* functions on a grid of width equal to the minimum of `width_spline`.
- `spline_der_rescaled` vector. Evaluation of the derivatives of the *scaled peaks* on a grid of width equal to the minimum of `width_spline`.

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

References

Ramsay, J.O., Silverman, B.W., 2005. Functional Data Analysis, 2nd ed. Springer, New York, NY.

Examples

```
# load the data
data(peaks)

# it computes the spline approximation
# of the pealks given the
# GRange with the metadata counts.
# It is obtained by the pileup_peak method

# Default paramters are used: GCV is
# computed on the derivatives.

peaks.spline <- smooth_peak(peaks.data, lambda = 10^(-4:6),
                            subsample.data = 50, GCV.derivatives = TRUE )

peaks.spline.scaled <- smooth_peak(peaks.data, lambda = 10^(-4:6),
                           subsample.data = 50, GCV.derivatives = TRUE, rescale = TRUE )
```

summit_peak *Finding the summits of the peaks of the GRanges object*

Description

It identifies the summit of the peak and stores it in a new metadata column.

Usage

```
## S4 method for signature 'GRanges'
summit_peak(object, summit = NULL, rescale = FALSE)
```

Arguments

object	GRanges object of length N. If summit is not provided, object must contain the metadata column spline.
summit	vector of length N. It contains the x coordinate of the summit of the peaks, i.e. the distance of the summit from the starting position of the spline approximation of peak (distance from start_spline). If summit is NULL the summit of each peak is identified as the maximum point of the spline.
rescale	logical. If TRUE the distance among *scaled peaks* is computed. For the definition of *scaled peaks* see smooth_peak.

Value

the GRanges object with the new metadata column summit_spline. In case of rescale = TRUE an extra metadata column summit_spline_rescaled is added, containing the summit of the *scaled peak*.

Author(s)

Alice Parodi, Marco J. Morelli, Laura M. Sangalli, Piercesare Secchi, Simone Vantini

Examples

```
# load the data
data(peaks)

# Computing the summits of the peaks from
# the spline-smoothed approximation.

peaks.spline.summit <- summit_peak(peaks.data.smooth)
```

Index