SEGCOND User Manual

Function

segment_genome(data,chr_list,window=1000,minimum_percentage=0.05,break_threshold =5,run_pca=T)

Description

Main segmentation function. The function will partition chromosomes into segments based on a provided "signal" file. A common example of "signal" is RPKM/TPM values of ATAC-seq and ChIP-seq experiments. Other types of values and experiments can be used.

Chromosome segments are defined after the identification of structural breaks, termed breakpoints, with the implementation of the R function breakpoints() from the strucchange package (Zeileis *et al.*, 2002). Given appropriate input data, breakpoints() applies multiple linear regression models along it and uses an F-test (Chow test) to identify boundaries between consecutive segments. For more details, please refer to our publication ***"SEGCOND identifies putative transcriptional condensate-associated genomic regions by integrating multi-omics data".***

Arguments

data

An R data.frame with the following columns:

Chromosome – Start – End - Value #1 - Value#2 - …. Value #N

The file has to be consisted of sorted, non-overlapping bins of fixed width (e.g. 5kb/10kb bins), spanning the beginning to the end of a chromosome. The file can include any number of chromosomes.

Every bin has to be associated with at least a specific value. If run_pca=T, then a PCA analysis is going to be applied on the values of the data provided and only the first principal component values will be kept for downstream analysis. Values from next generation sequencing experiments should be scaled for sequencing depth.

chr_list

A vector specifying the chromosomes that are going to be included in the analysis. The function is going to calculate segments for only the chromosomes specified here even if additional chromosomes are present in the **data** file.

window

A single numeric variable with the number of bins that are evaluated at every iteration of the algorithm. The algorithm applies a sliding window, per chromosome included in the analysis, of size **window**, and calculates breakpoints using the breakpoints() function of the strucchange package at each iteration. Breakpoints demarcate the boundaries of segments. The window slides for **window**/2 until it reaches the end of a chromosome. Default 1000.

minimum_percentage

A single numeric variable that defines the minimum size a segment can have per sliding window iteration. This value is expressed as a percentage and it is related to the window size. Default 0.05.

break_threshold

A single numeric variable. After the sliding window for each chromosome has reached the end of it, breakpoints that are close to each other are merged to the smaller one if their distance is smaller than the threshold defined here. Distance is expressed as number of bins. Default 5.

run_pca

Boolean. If specified as TRUE, a PCA is applied in the data of the input file and the values of the first principal component are used to partition chromosome into segments. This is due to breakpoints() accepting only one-dimensional data. If FALSE, the algorithm expects a single variable per bin and proceeds with the segmentation step using that variable instead.

Output

A list with 5 slots containing the following information:

Slot 1: The input data used for the segmentation pipeline. Data is provided after the PCA transformation. If **run_pca=FALSE** was used, then the original data provided are returned.

Slot 2: A list with the final segment coordinates per chromosome.

Slot 3: Average values and standard deviation of the PC1 values within each segment. If **run_pca=FALSE** was used, then the average and the standard deviation of the input values per segment are provided.

Slot 4: The position of the final breakpoints per chromosome. This data can be useful in case you want to visually represent segment boundaries per chromosome.

Slot 5: Segment coordinates merged together as a single data.frame containing all chromosomes used for the analysis.

Function

get_z_score(marks)

Description

This function adds a pseudocount of 0.01 and then log2 transforms all columns starting from the fourth one of a given file. Afterwards a z-scale transformation is performed per aforementioned column.

Arguments

marks

An R data.frame with the following columns:

Chromosome – Start – End - Value #1 - Value#2 - …. Value #N

The file has to be consisted of sorted, non-overlapping bins of fixed width (e.g. 5kb/10kb bins), spanning the beginning to the end of a chromosome. The file can include any number of chromosomes.

Output

An R data.frame with the same three first columns as **marks** and the rest of the columns log2 normalized and z-scaled.

Function

votes_matrix(cooccupancy_mat,picked_columns=NULL,threshold=NULL,picked_lines=FALSE ,lines=NULL)

Description

The function returns chromosomal coordinates marked with a 0 or 1.

Arguments

cooccupancy_mat

An R data.frame with the following columns:

Chromosome – Start – End - Value #1 - Value#2 - …. Value #N

The file has to be consisted of sorted, non-overlapping bins of fixed width (e.g. 5kb/10kb bins), spanning the beginning to the end of a chromosome. The file can include any number of chromosomes.

picked_columns

A vector containing column numbers. Values of the provided columns will be checked whether they exceed a given **threshold.** In case they do for a specific chromosomal bin, a "1" value is given to the bin and a value of "0" otherwise. This argument is omitted if **picked_lines=TRUE**

threshold

A number indicating the minimum value that values in the **picked_columns** of the file have to surpass in order to assign a value of "1". This argument is omitted if picked_lines=TRUE

picked_lines

Boolean. If TRUE the user is expected to provide a vector with the row numbers of **cooccupancy_mat** that correspond to "interesting" bins.

lines

A vector containing the row numbers of "interesting" bins found in **cooccupancy_mat.** The argument is omitted if **picked_lines=FALSE**

<u>Output</u>

A data.frame with 4 columns. The first three columns correspond to chromosomal coordinates. The fourth is a column that only takes 0/1 values. 1 values correspond to bins that are "interesting" and 0 to the rest.

<u>Function</u>

fit_zinegbin_model(segments,new_scores,permutations)

<u>Description</u>

This function evaluates whether a segment is enriched in bins that are "interesting" by fitting a background zero-inflated negative binomial model. The function will:

1) Identify the length distribution of the given segments.

2) For every different segment length, segment borders will be randomized and random segments will be produced. For every iteration the number of “interesting” bins falling within the randomized segment is recorded.

3) A zero-inflated negative binomial model is fit for each segment length, modeling the number of interesting bins that are “expected” for a segment of given length.

A p-value and logFC value is then assigned to every segment according to the random distribution that was fit.

Arguments

segments

A data.frame with segments, as produced with segment_genome()

new_scores

A 4 column data.frame with chromosomal bins as the first three columns (the same bins used to call the **segments** with **segment_genome()**) and a fourth column containing 0/1 values. Bins that are associated with a value of 1 are deemed “interesting” while the others not. This object can be produced with the **votes_matrix()** function described above.

permutations

A numeric variable. Zero-inflated negative binomial models are built as background models for each segment of different length, after randomizing the boundaries of the segments and counting the number of interesting bins falling into the randomized segments. This argument controls the number of times this randomizing procedure is going to occur.

Output

A data.frame containing the **segments** provided accompanied by logFC enrichment values and p-values.

Function

get_segment_interactions(misha_path,hic_normalized_track,chr_list,segments,distance_filter=Inf)

Description

The function calculated the average and the median of SHAMAN normalized Hi-C interaction scores found across a segment pair.

Arguments

misha_path

Character variable specifying the path to a misha database. For more details on how to create a misha database refer to either the misha or shaman R packages.

hic_normalized_track

Character variable specifying the name of a SHAMAN Hi-C normalized track.

chr_list

A vector containing chromosome names. Segment SHAMAN interactions are going to be calculated only for segments found in the defined chromosomes.

segments

A data.frame with at least three columns. The first three columns should correspond to chromosomal coordinates of segments.

distance_filter

A numeric variable. If the distance between the genomic coordinates of a pair of segments exceeds the **distance_filter,** no SHAMAN score is going to be calculated. Since Hi-C coverage declines with distance, there aren't enough data points to infer whether two segments truly interact in 3D space for segments that are far apart. Setting this parameter also limits substantially the computational time needed to run this function. Default Inf, i.e. no filter is applied. **Setting a filter is strongly recommended, see our paper for more details.**

Output

A list with two slots.

Slot 1: A list with a symmetric matrix per chromosome with the average of all normalized SHAMAN interaction scores between a pair of segments. rownames and colnames of the matrix correspond to the segments used as input.

Slot 2: Same as Slot 1 but with the median reported.

Function

get_putative_condensates(shaman_hic_scores,shaman_threshold,qualifying_segments,distance_filter)

<u>Description</u>

The function returns a final list of **P**utative **T**ranscriptional **C**ondensates (PTCs).

<u>Arguments</u>

shaman_hic_scores

A list produced by **get_segment_interactions()**

shaman_threshold

A numeric variable. Segments/Segment pairs that exhibit an interaction score higher than **shaman_threshold** are considered to interact in 3D space.

qualifying_segments

A character vector. The vector contains the names of the segments that exhibit a desired property, e.g. they are enriched in enhancer marks.

distance_filter

A numeric variable. Segments whose genomic distance is higher than **distance_filter** will not be considered as interacting in 3D space, no matter the reported SHAMAN score.