Package 'fcScan'

November 12, 2026

Type Package

Title fcScan for detecting clusters of coordinates with user defined options

Version 1.26.0

biocViews GenomeAnnotation, Clustering

Maintainer Pierre Khoueiry <pk17@aub.edu.lb> Abdullah El-Kurdi <ak161@aub.edu.lb>

Description This package is used to detect combination of genomic coordinates falling within a user defined window size along with user defined overlap between identified neighboring clusters. It can be used for genomic data where the clusters are built on a specific chromosome or specific strand. Clustering can be performed with a ``greedy" option allowing thus the presence of additional sites within the allowed window size.

License Artistic-2.0

Encoding UTF-8

Imports stats, plyr, VariantAnnotation, SummarizedExperiment, rtracklayer, GenomicRanges, methods, IRanges, foreach, doParallel, parallel

VignetteBuilder knitr

Suggests RUnit, BiocGenerics, BiocStyle, knitr, rmarkdown

RoxygenNote 6.1.0

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

git_branch RELEASE_3_23

git_last_commit 663fe9f

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-11

Author Abdullah El-Kurdi [aut],
Ghiwa khalil [aut],
Georges Khazen [ctb],
Pierre Khoueiry [aut, cre]

Contents

getCluster 2

Index **5**

`getCluster` *Detect clusters of genomic features (i.e TFBS) in a specified window size and using user defined options*

Description

Given a data frame or a GRanges object or a list of bed/vcf files, this function will look for clusters of genomic features found within a user defined window size and satisfiying user defined categorical conditions.

Usage

```
getCluster(x, w, c, overlap = 0, greedy = FALSE, seqnames = NULL, s = "*",
order = NULL, site_orientation = NULL, site_overlap = 0,
cluster_by = "startsEnds", allow_clusters_overlap = FALSE,
include_partial_sites = FALSE, partial_overlap_percentage= NULL,
threads = 1, verbose = FALSE)
```

Arguments

`x`	Vector of file names or a data frame
`w`	Value for the desired cluster size (numeric)
`c`	A named vector for condition relative to each site name
`overlap`	numeric; If negative, it will correspond to the maximum overlap allowed between consecutive clusters, if positive, it will correspond to the minimum gap allowed between consecutive clusters. The default is set to 0
`greedy`	logical; If FALSE, the formed clusters will contain the exact combination of sites defined in "c". If TRUE, clusters containing the condition and more will be labeled as TRUE as long as window size is below or equal the defined value. The default is set to FALSE
`seqnames`	character; seqname/chromosome to cluster on. The default is set to NULL so the clustering will be performed on all seqnames
`s`	character; The strand to cluster on. Can be "+", "-" or "*". The default is set to "*" which indicate both clusters
`order`	vector; A vector containing the desired order of sites within the identified clusters. The default is set to NULL indicating that the order of sites is not important
`site_orientation`	vector; A vector containing the desired orientation of the sites within the identified clusters. The default is set to NULL

`site_overlap` numeric; Same as "overlap" but for sites. If negative, it will correspond to the maximum overlap allowed between consecutive sites. If positive, it will correspond to the minimum gap allowed between consecutive sites. The default is set to 0

`cluster_by` vector; A vector containing one clustering option. A clustering option is the way in which sites in input data are scanned. For example, if we choose "starts", the clustering begins at the start of the first site and it ends at the start of the last site within the window size. It can be "startsEnds", "endsStarts", "starts", "ends", or "middles". The default is set to "startsEnds"

`allow_clusters_overlap` logical; This option can be only used when "cluster_by" is assigned. It allows to control whether the user wants to have overlapping clusters or not. The default is FALSE

`include_partial_sites` logical; This option can be only used in case where the "cluster_by" choosen is "startsEnds" or "ends". If it is set to TRUE, partially overlapping sites within window size (at the end of the cluster) are allowed. However, if it is set to FALSE, partially overlapping sites are not included in the cluster found. The default is set to FALSE

`partial_overlap_percentage` numeric; This option can be only used in case where the "cluster_by" choosen is "startsEnds" or "ends". it ranges from 0 to 1. If it is set, partially overlapping sites that overlap with the window size (at the end of the cluster) by this percentage are allowed. However, if it is set to 0, it means that all partially overlapping sites are included in the cluster found regardless of the amount of overlap. The default is set to NULL

`threads` integer; This option allows to run getCluster using multiple threads. It is recommended to use it especially when the input dataset used is large. When set to 0, getCluster checks the input dataset and uses the adequate number of threads to get the result faster. It can be set manually according to the available number of threads. The default is set to 1

`verbose` logical; If FALSE, only clusters that passed combination condition are shown. used for debugging purposes. By default it is set to FALSE

Details

The function *getCluster* will cluster coordinates based on a user defined number of sites and window size. The user needs to specify the condition for clustering, which is the sites and the number of sites required in each cluster. The user can exclude sites from clusters. this should be specified also in the condition for clustering. the user can also set the distance required between consective clusters using the overlap argument. If overlap is a negative, this will represent the maximum overlap allowed between clusters. If overlap is positive, this means that the clusters should have a minimum gap of the given value.

The user can also choose to cluster on a specific strand by using the strand option, and specific seqname by specifying the seqnames as an argument.

x is a vector of files or a data frame. When data frame is given as input, it should have 5 columns *seqnames start end strand site*. *start* and *end* column need to be numeric or integer and the rest of the columns are of type character.

The Greedy option controls if more sites are allowed in the cluster. When set to TRUE, getCluster will return clusters containing the required number of sites and more as long as the window size condition is satisfied. When set to FALSE, getCluster will return clusters containing the exact number of the required sites. Default is set to FALSE.

order allows the user to choose whether the clusters should contain sites ordered in a certain way. When set to NULL, sites order is not important. When defined, the order of the sites in the cluster must abide by the user defined order. Default is NULL.

cluster_by is the way in which sites in input data are scanned. For example, if we choose "starts", the clustering begins at the start of the first site and it ends at the start of the last site within the window size. Also, if we choose "ends", the clustering begins at the end of the first site and it ends at the end of the last site within the window size. It can be "startsEnds", "endsStarts", "starts", "ends", or "middles".

Value

The returned value is a GRanges object containing the clusters.

Author(s)

Abdullah El-Kurdi

Ghiwa Khalil

Georges Khazen

Pierre Khoueiry

Examples

```
x = data.frame(seqnames = rep("chr1", times = 16),
start = c(10,17,25,27,32,41,47,60,70,87,94,99,107,113,121,132),
end = c(15,20,30,35,40,48,55,68,75,93,100,105,113,120,130,135),
strand = rep("+", 16),
site = c("s1","s2","s2","s1","s2","s1","s1","s2",
"s1","s2","s2","s1","s2","s1","s1","s2"))

clusters = getCluster(x, w = 25, c = c("s1"=1,"s2"=2),
greedy = TRUE, overlap = -5, s = "+", order = c("s1","s2","s1"))
```

Index

getCluster, 2