

Package ‘geneRxCluster’

July 3, 2025

Date 2013-02-13

Version 1.45.0

License GPL (>= 2)

Description Detect Differential Clustering of Genomic Sites such as gene therapy integrations. The package provides some functions for exploring genomic insertion sites originating from two different sources. Possibly, the two sources are two different gene therapy vectors. Vectors are preferred that target sensitive regions less frequently, motivating the search for localized clusters of insertions and comparison of the clusters formed by integration of different vectors. Scan statistics allow the discovery of spatial differences in clustering and calculation of False Discovery Rates (FDRs) providing statistical methods for comparing retroviral vectors. A scan statistic for comparing two vectors using multiple window widths to detect clustering differentials and compute FDRs is implemented here.

Title gRx Differential Clustering

Author Charles Berry

Maintainer Charles Berry <ccberry@ucsd.edu>

Depends GenomicRanges,IRanges

Suggests RUnit, BiocGenerics

biocViews Sequencing, Clustering, Genetics

git_url <https://git.bioconductor.org/packages/geneRxCluster>

git_branch devel

git_last_commit ae7b8b9

git_last_commit_date 2025-04-15

Repository Bioconductor 3.22

Date/Publication 2025-07-03

Contents

critVal.alpha	2
critVal.power	3
critVal.target	4

geneRxCluster	5
gRxCluster	5
gRxCluster-object	7
gRxPlot	8
gRxPlotClumps	9
gRxSummary	10
noprune	11
plot.cutpoints	11
prune.loglik	12

Index	13
--------------	-----------

critVal.alpha	<i>critical regions</i>
---------------	-------------------------

Description

critical region cutpoints

Usage

```
critVal.alpha(k, p0, alpha, posdiff)
```

Arguments

k	- window width(s)
p0	- length 2 probabilities
alpha	- two tailed
posdiff	- position difference matrix

Details

This version uses alpha and will find TFD

Value

list of cutoffs and attributes

Author(s)

Charles Berry

See Also

[gRxCluster](#) for how and why this function is used

Examples

```
# symmetric odds:
crit <- critVal.alpha(5:25,c(1,1)/2,alpha=0.05,
                     matrix(1,nr=50,nc=21))

crit[[1]]
sapply(crit,c)
# 5:1 odds
asymmetric.crit <- critVal.alpha(5:25,c(1,5)/6,alpha=0.05,
                                matrix(1,nr=50,nc=21))
# show the critical regions
par(mfrow=c(1,2))
gRxPlot(crit,method="critical")
gRxPlot(asymmetric.crit,method="critical")
rm(crit,asymmetric.crit)
```

critVal.power	<i>critical regions</i>
---------------	-------------------------

Description

critical region cutpoints

Usage

```
critVal.power(k, p0, target, pwr = 0.8, odds = 7)
```

Arguments

k	- window width(s)
p0	- length 2 probabilities
target	- false discoveries wanted
pwr	- desired power
odds	- alternative odds ratio

Details

This version uses power and TFD and will limit windows screened

Value

list of cutoffs and attributes

Author(s)

Charles Berry

See Also

[gRxCluster](#) for how and why this function is used

Examples

```
# symmetric odds:
crit <-
  critVal.power(5:25,c(1,1),5,pwr=0.8,odds=7)
crit[[1]]
sapply(crit,c)
# 5:1 odds
asymmetric.crit <-
  critVal.power(5:25,c(1,5),5,pwr=0.8,odds=7)
# show the critical regions
par(mfrow=c(1,2))
gRxPlot(crit,method="critical")
gRxPlot(asymmetric.crit,method="critical")
rm(crit,asymmetric.crit)
```

critVal.target	<i>critical regions</i>
----------------	-------------------------

Description

critical region cutpoints

Usage

```
critVal.target(k, p0, target, posdiff = NULL, ns)
```

Arguments

k	window width(s)
p0	length 2 probabilities
target	- two tailed
posdiff	- position difference matrix
ns	the number of windows passing filter at each k

Details

This version uses TFD and will find alpha implicitly

Value

list of cutoffs and attributes

Author(s)

Charles Berry

See Also

[gRxCluster](#) for how and why this function is used

Examples

```
# symmetric odds:
crit <- critVal.target(5:25,c(1,1),1,ns=rep(10,21))
crit[[1]]
sapply(crit,c)
# 5:1 odds
asymmetric.crit <- critVal.target(5:25,c(1,5),1,ns=rep(10,21))
# show the critical regions
par(mfrow=c(1,2))
gRxPlot(crit,method="critical")
gRxPlot(asymmetric.crit,method="critical")
rm(crit,asymmetric.crit)
```

geneRxCluster

Differential Clustering of Integration Sites

Description

geneRxCluster provides the function [gRxCluster](#) and friends.

Details

Windows defined by k consecutive integration sites are scanned. A two class indicator is tallied to determine whether one class dominates. If one does, a flag is set and the window is retained. Various values of k are used. Conflicts between overlapping windows with the same value of k can occur — two windows are dominated by the two different classes. In that case, the sites of overlap are marked and neither window is retained. Conflicts can also arise between windows differing in their values of k . In that case, the window having the smaller value of k is retained and the other is discarded.

Permutation tests and permutation based false discovery rates are available.

Filtering of windows is allowed so that regions which are sparsely populated need not be studied.

[gRxCluster](#)

gRxCluster

Description

cluster integration sites - optionally perform the permutations needed to estimate the discoveries expected under a null hypothesis

Usage

```
gRxCluster(object, starts, group, kval, nperm = 0L,
  pruneFun = prune.loglik, ..., cutpt.filter.expr, cutpt.tail.expr, tmp.env,
  sample.id, sample.tab)
```

Arguments

<code>object</code>	chromosome names or other grouping of starts
<code>starts</code>	ordered chromosome position or ordered integer vector
<code>group</code>	logical vector separating two groups
<code>kvals</code>	integer vector of window widths
<code>nperm</code>	number of permutations for FDR calculation
<code>pruneFun</code>	a function like <code>prune.loglik</code> .
<code>...</code>	other args
<code>cutpt.filter.expr</code>	(optional) R object or call (or variable naming a call) with (optional) var <code>x</code> (window widths in base pairs) to filter windows. It must evaluate to mode "double". If not specified, <code>as.double(apply(x,2,median,na.rm=TRUE))</code> is used. If an atomic vector of length one is supplied it is expanded to the proper length and coerced to double. If this arg is the name of a variable provided in <code>tmp.env</code> , it must be protected with <code>quote(...)</code> .
<code>cutpt.tail.expr</code>	R object or call (or variable naming a call) with (optional) vars: <code>k</code> , <code>n</code> , and <code>x</code> (as above). Returns list like <code>critVal.target</code> . <code>k</code> is a vector of the number of sites in a collection of windows, and <code>n</code> is a vector of counts or proportions for the two classes of insertion. If not supplied, <code>critVal.target(k,n,target=5,posdiff=x)</code> is used. If this arg is the name of a variable provided in <code>tmp.env</code> , it must be protected with <code>quote(...)</code> .
<code>tmp.env</code>	(optional) environment to use in evaluation of <code>cutpt.*</code> expressions. This is usually needed for <code>critVal.power</code> , which is first calculated and placed in the environment, and the supplied object is used in the expression for <code>cutpt.filter.expr</code> .
<code>sample.id</code>	(optional) integer vector indexing cells in <code>sample.tab</code> to be looked up to determine group under permutation. A factor can be used, too, but will be coerced to integer.
<code>sample.tab</code>	(optional) integer vector containing 0 or 1 in each cell. Its length is the same as <code>max(sample.id)</code> . Both or neither <code>sample.id</code> and <code>sample.tab</code> should be supplied. When supplied <code>sample.tab[sample.id]</code> must equal <code>group</code> . If the arguments are supplied, permutations are of the form <code>sample(sample.tab)[sample.id]</code> . Otherwise they are of the form <code>sample(group)</code> .

Value

a GRanges object with a special metadata slot, see [gRxCluster-object](#)

Author(s)

Charles Berry `example inst/ex-gRxCluster.R`

gRxCluster-object	<i>gRxCluster object</i>
-------------------	--------------------------

Description

Overview of the result of `gRxCluster(...)`

Details

The object returned is a [GRanges](#) object.

If the object is `x`, `seqnames(x)` and `ranges(x)` slots demarcate the clusters discovered. There will be one element for each cluster (aka ‘clump’) discovered.

Using the default argument `pruneFun=prune.loglik` or `pruneFun=noprune`, `mcols(x)` will have these columns:

`value1` **and** `value2` are the counts of the two classes of insertion sites for the clusters of object `x`
`clump.id` numbers each cluster.

If the user supplies a custom `pruneFun`, it should return a `GRanges` with those columns and one element for each unique `clump.id`. The column `target.min` has the smallest nominal False Discoveries Expected for each cluster and is added to (or replaces) the `mcols(x)` produced by the argument supplied as `pruneFun`.

`metadata(x)` will include these components:

criticalValues A list object such as supplied by `critVal.target` whose elements each give the cutpoints to be used for a window with `k` sites. `attributes(metadata(object)$criticalValues[[i]])` will contain elements

fd with dimension `c(k+1,4)` of target false discovery expectations and the one-sided p-values

target the target for false discovery which sometimes is specified a priori and sometimes results from calculation

n an upper bound on the number of windows to screen, if this number is needed.

In some cases, an attribute is attached to `metadata(object)$criticalValues`, see [critVal.power](#) for an example.

kvals the number of sites, `k`, to include in a window

perm_cluster_best a list whose canonical element is a vector of values like `x$target.min` obtained from a permutation of the class indicators

summary_matrix a matrix giving the start, end, depth, and counts in each class for every cluster and depth in sequential order

call the call invoking [gRxCluster](#) which may include some arguments added by default.

Author(s)

Charles Berry <ccberry@ucsd.edu>

gRxPlot

gRxPlot

Description

Plot Clumps and/or Critical Regions

Usage

```
gRxPlot(object, pi.0 = NULL, method = c("odds", "criticalRegions"),
        xlim = NULL, main = NULL, xlab = "log odds ratio", breaks = "Sturges",
        kvals = NULL, ...)
```

Arguments

object	either the results of gRxCluster or a list containing cutpoints for critical regions.
pi.0	the background proportion for vector 2
method	character vector of “odds” and/or “criticalRegions”
xlim	limits of the log odds histogram
main	a title for the panel(s)
xlab	label fgor the x-axis of the log odds plot
breaks	see hist
kvals	values to use in selecting a subset of the critical regions to display
...	other args to pass to the plotting routine(s)

Details

The results of a call to [gRxCluster](#) are plotted. Optionally, with `method="criticalRegions"` only the critical regions are plotted or with `method="odds"` the log odds only are plotted.

Value

see [hist](#)

Author(s)

Charles Berry

See Also

[gRxPlotClumps](#) for a more fine grained display

Examples

```
x.seqnames <- rep(letters[1:3],each=500)
x.starts <- c(seq(1,length=500),seq(1,by=2,length=500),seq(1,by=3,length=500))
x.lens <- rep(c(5,10,15,20,25),each=20)
x.group <- rep(rep(c(TRUE,FALSE),length=length(x.lens)),x.lens)
## add a bit of fuzz:
x.group <- 1==rbinom(length(x.group),1,pr=ifelse(x.group,.8,.2))
x.kvals <- as.integer(sort(unique(x.lens)))
x.res <- gRxCluster(x.seqnames,x.starts,x.group,x.kvals)
gRxPlot(x.res)
rm( x.seqnames, x.starts, x.lens, x.group, x.kvals, x.res)
```

gRxPlotClumps

*gRxPlotClumps***Description**

Plot gRxCluster object clumps

Usage

```
gRxPlotClumps(object, data, seqLens, panelExpr = quote(grid()))
```

Arguments

object	result of gRxCluster
data	(optional) GRanges like that from which args to gRxCluster were derived
seqLens	(optional) seqlengths(data) or similar. Can be given if data is missing
panelExpr	- an expression to evaluate after drawing each panel

Details

Plot Relative Frequencies of the two classes according to region. Regions typically alternate between clusters and non-clusters on each chromosome.

Author(s)

Charles Berry

Examples

```
x.seqnames <- rep(letters[1:3],each=50)
x.starts <- c(seq(1,length=50),seq(1,by=2,length=50),seq(1,by=3,length=50))
x.lens <- rep(c(5,10,15,20,25),each=2)
x.group <- rep(rep(c(TRUE,FALSE),length=length(x.lens)),x.lens)
## add a bit of fuzz:
x.group <- 1==rbinom(length(x.group),1,pr=ifelse(x.group,.8,.2))
x.kvals <- as.integer(sort(unique(x.lens)))
x.res <- gRxCluster(x.seqnames,x.starts,x.group,x.kvals)
gRxPlotClumps(x.res)
rm( x.seqnames, x.starts, x.lens, x.group, x.kvals, x.res)
```

gRxSummary

gRxSummary

Description

Summarize gRxCluster Results

Usage

```
gRxSummary(object, targetFD = NULL)
```

Arguments

object	the result of gRxCluster
targetFD	the critical value target in each tail

Details

Get the FDR and related data for a run of gRxCluster. By selecting a value for targetFD that is smaller than what was used in constructing the object, fewer clumps will be included in the computation for the False Discovery Rate - akin to what would have been obtained from the object if it had been constructed using that value.

Value

a list containing the summarized results

Author(s)

Charles Berry

Examples

```
x.seqnames <- rep(letters[1:3],each=50)
x.starts <- c(seq(1,length=50),seq(1,by=2,length=50),seq(1,by=3,length=50))
x.lens <- rep(c(5,10,15,20,25),each=2)
x.group <- rep(rep(c(TRUE,FALSE),length=length(x.lens)),x.lens)
x.kvals <- as.integer(sort(unique(x.lens)))
x.res <- gRxCluster(x.seqnames,x.starts,x.group,x.kvals,nperm=100L)
gRxSummary(x.res)
rm( x.seqnames, x.starts, x.lens, x.group, x.kvals, x.res)
```

<code>noprune</code>	<i><code>noprune</code></i>
----------------------	-----------------------------

Description

join contiguous windows

Usage

```
noprune(x, ...)
```

Arguments

<code>x</code>	a GRanges object
<code>...</code>	currently unused

Details

return all the candidate sites in a clump without pruning
this is to be used as the `pruneFun` are of `gRxCluster`

Value

same as [gRxCluster](#) less the metadata

Author(s)

Charles Berry

See Also

[gRxCluster-object](#) for more details on what this function returns.

<code>plot.cutpoints</code>	<i><code>plot.cutpoints</code></i>
-----------------------------	------------------------------------

Description

Plot a set of cutpoints - Utility

Usage

```
## S3 method for class 'cutpoints'
plot(crit, pi.0 = NULL, kvals = NULL, ...)
```

Arguments

<code>crit</code>	- a cutpoint object see gRxCluster
<code>pi.0</code>	- optional null value to plot
<code>kvals</code>	- which cutpoints to include in the plot
<code>...</code>	passed to <code>barplot</code>

Details

NOT FOR USERS. Not exported.

Value

list with components of “bar.x” (the value of `hist()`), “kvals” (window widths plotted), and “pi.0” (the input value of `pi.0`)

Author(s)

Charles Berry

`prune.loglik`

prune.loglik

Description

best contiguous region

Usage

```
prune.loglik(x, p.null = 0.5)
```

Arguments

<code>x</code>	a GRanges object
<code>p.null</code>	the probability of category 1 (FALSE)

Details

prune each end of the region using loglik criterion
this is to be used as the `pruneFun` are of `gRxCluster`

Author(s)

Charles Berry

See Also

[gRxCluster-object](#) for details on what this function returns.

Index

* **cluster**

gRxCluster-object, [7](#)

critVal.alpha, [2](#)

critVal.power, [3](#), [6](#), [7](#)

critVal.target, [4](#), [7](#)

geneRxCluster, [5](#)

geneRxCluster-package (geneRxCluster), [5](#)

GRanges, [7](#)

gRxCluster, [2-5](#), [5](#), [7](#), [8](#), [11](#)

gRxCluster-object, [7](#)

gRxPlot, [8](#)

gRxPlotClumps, [8](#), [9](#)

gRxSummary, [10](#)

hist, [8](#)

noprune, [11](#)

plot.cutpoints, [11](#)

prune.loglik, [12](#)