Package 'derfinderPlot'

May 19, 2022

Type Package

Title Plotting functions for derfinder

Version 1.30.0

Date 2021-11-22

Depends R(>= 3.2)

Imports derfinder (>= 1.1.0), GenomeInfoDb (>= 1.3.3), GenomicFeatures, GenomicRanges (>= 1.17.40), ggbio (>= 1.13.13), ggplot2, graphics, grDevices, IRanges (>= 1.99.28), limma, methods, plyr, RColorBrewer, reshape2, S4Vectors (>= 0.9.38), scales, utils

Suggests biovizBase (>= 1.27.2), bumphunter (>= 1.7.6), derfinderData (>= 0.99.0), sessioninfo, knitr (>= 1.6), BiocStyle (>= 2.5.19), org.Hs.eg.db, RefManageR, rmarkdown (>= 0.3.3), testthat, TxDb.Hsapiens.UCSC.hg19.knownGene, covr

VignetteBuilder knitr

Description This package provides plotting functions for results from the derfinder package. This helps separate the graphical dependencies required for making these plots from the core functionality of derfinder.

License Artistic-2.0

LazyData false

URL https://github.com/leekgroup/derfinderPlot

BugReports https://support.bioconductor.org/t/derfinderPlot

biocViews DifferentialExpression, Sequencing, RNASeq, Software, Visualization, ImmunoOncology

RoxygenNote 7.1.1

Encoding UTF-8

Roxygen list(markdown = TRUE)

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

git_branch RELEASE_3_15

git_last_commit 4ebf7a9

git_last_commit_date 2022-04-26

Date/Publication 2022-05-19

Author Leonardo Collado-Torres [aut, cre]
(<https://orcid.org/0000-0003-2140-308X>),
Andrew E. Jaffe [aut] (<https://orcid.org/0000-0001-6886-1454>),
Jeffrey T. Leek [aut, ths] (<https://orcid.org/0000-0002-2873-2671>)

Maintainer Leonardo Collado-Torres <lcolladotor@gmail.com>

R topics documented:

plotCluster 2
plotOverview 4
plotRegionCoverage 6
vennRegions 8

Index **10**

`plotCluster` *Plot the coverage information surrounding a region cluster*

Description

For a given region found in calculatePvalues, plot the coverage for the cluster this region belongs to as well as some padding. The mean by group is shown to facilitate comparisons between groups. If annotation exists, you can plot the trancripts and exons (if any) overlapping in the vicinity of the region of interest.

Usage

```
plotCluster(
  idx,
  regions,
  annotation,
  coverageInfo,
  groupInfo,
  titleUse = "qval",
  txdb = NULL,
  p.ideogram = NULL,
  ...
)
```

Arguments

`idx`	A integer specifying the index number of the region of interest. This region is graphically highlighted by a red bar.
`regions`	The `$regions` output from calculatePvalues.

`annotation` The output from running matchGenes on the output from calculatePvalues.

`coverageInfo` A DataFrame resulting from loadCoverage using `cutoff=NULL`.

`groupInfo` A factor specifying the group membership of each sample. It will be used to color the samples by group.

`titleUse` Whether to show the p-value (`pval`), the q-value (`qval`) or the FWER adjusted p-value (`fwer`) in the title. If `titleUse=none` then no p-value or q-value information is used; useful if no permutations were performed and thus p-value and q-value information is absent.

`txdb` A transcript data base such as `TxDb.Hsapiens.UCSC.hg19.knownGene`. If `NULL` then no annotation information is used.

`p.ideogram` If `NULL`, the ideogram for hg19 is built for the corresponding chromosome. Otherwise an ideogram resuling from plotIdeogram.

`...` Arguments passed to other methods and/or advanced arguments. Advanced arguments:

maxExtend The maximum number of base-pairs to extend the view (on each side) before and after the region cluster of interest. For small region clusters, the one side extension is equal to the width of the region cluster.

colsubset Column subset in case that it was specified in preprocessCoverage.

forceLarge If `TRUE` then the data size limitations are ignored. The window size (region cluster width + 2 times `maxExtend`) has to be less than 100 kb. Note that a single plot at the 300kb range can take around 2 hours to complete.

Details

See the parameter `significantCut` in calculatePvalues for how the significance cutoffs are determined.

Value

A ggplot2 plot that is ready to be printed out. Tecnically it is a ggbio object. The region with the red bar is the one whose information is shown in the title.

Author(s)

Leonardo Collado-Torres

See Also

loadCoverage, calculatePvalues, matchGenes, plotIdeogram

Examples

```
## Load data
library("derfinder")

## Annotate the results with bumphunter::matchGenes()
library("bumphunter")
library("TxDb.Hsapiens.UCSC.hg19.knownGene")
```

```
library("org.Hs.eg.db")
genes <- annotateTranscripts(
    txdb = TxDb.Hsapiens.UCSC.hg19.knownGene,
    annotationPackage = "org.Hs.eg.db"
)
annotation <- matchGenes(x = genomeRegions$regions, subject = genes)

## Make the plot
plotCluster(
    idx = 1, regions = genomeRegions$regions, annotation = annotation,
    coverageInfo = genomeDataRaw$coverage, groupInfo = genomeInfo$pop,
    txdb = TxDb.Hsapiens.UCSC.hg19.knownGene
)
## Resize the plot window and the labels will look good.
## Not run:
## For a custom plot, check the ggbio and ggplot2 packages.
## Also feel free to look at the code for this function:
plotCluster

## End(Not run)
```

`plotOverview` *Plot a karyotype overview of the genome with the identified regions*

Description

Plots an overview of the genomic locations of the identified regions (see calculatePvalues) in a karyotype view. The coloring can be done either by significant regions according to their p-values, significant by adjusted p-values, or by annotated region if using matchGenes.

Usage

```
plotOverview(
  regions,
  annotation = NULL,
  type = "pval",
  significantCut = c(0.05, 0.1),
  ...
)
```

Arguments

`regions`	The `$regions` output from calculatePvalues.
`annotation`	The output from running matchGenes on the output from calculatePvalues. It is only required if `type='annotation'`.

`type` Must be either `pval`, `qval`, `fwer` or `annotation`. It determines whether the plot coloring should be done according to significant p-values (<0.05), significant q-values (<0.10), significant FWER adjusted p-values (<0.05) or annotation regions.

`significantCut` A vector of length two specifiying the cutoffs used to determine significance. The first element is used to determine significance for the p-values and the second element is used for the q-values.

`...` Arguments passed to other methods and/or advanced arguments. Advanced arguments:

base_size Base point size of the plot. This argument is passed to element_text (`size` argument).

areaRel The relative size for the area label when `type='pval'` or `type='qval'`. Can be useful when making high resolution versions of these plots in devices like CairoPNG.

legend.position This argument is passed to theme. From ggplot2: the position of legends. ('left', 'right', 'bottom', 'top', or two-element numeric vector).

Passed to extendedMapSeqlevels.

Value

A ggplot2 plot that is ready to be printed out. Tecnically it is a ggbio object.

Author(s)

Leonardo Collado-Torres

See Also

calculatePvalues, matchGenes

Examples

```
## Construct toy data
chrs <- paste0('chr', c(1:22, 'X', 'Y'))
chrs <- factor(chrs, levels=chrs)
library('GenomicRanges')
regs <- GRanges(rep(chrs, 10), ranges=IRanges(runif(240, 1, 4e7),
    width=1e3), significant=sample(c(TRUE, FALSE), 240, TRUE, p=c(0.05,
    0.95)), significantQval=sample(c(TRUE, FALSE), 240, TRUE, p=c(0.1,
    0.9)), area=rnorm(240))
annotation <- data.frame(region=sample(c("upstream", "promoter",
    "overlaps 5'", "inside", "overlaps 3'", "close to 3'", "downstream"),
    240, TRUE))

## Type pval
plotOverview(regs)

## Not run:
## Type qval
```

```
plotOverview(regs, type='qval')

## Annotation
plotOverview(regs, annotation, type='annotation')

## Resize the plots if needed.

## You might prefer to leave the legend at ggplot2's default option: right
plotOverview(regs, legend.position='right')

## Although the legend looks better on the bottom
plotOverview(regs, legend.position='bottom')

## Example knitr chunk for higher res plot using the CairoPNG device
```{r overview, message=FALSE, fig.width=7, fig.height=9, dev='CairoPNG', dpi=300}
plotOverview(regs, base_size=30, areaRel=10, legend.position=c(0.95, 0.12))
```

For more custom plots, take a look at the ggplot2 and ggbio packages
and feel free to look at the code of this function:
plotOverview

End(Not run)
```

---

plotRegionCoverage *Makes plots for every region while summarizing the annotation*

Description

This function takes the regions found in calculatePvalues and assigns them genomic states contructed with makeGenomicState. The main workhorse functions are countOverlaps and findOverlaps. For an alternative plot check plotCluster which is much slower and we recommend it's use only after quickly checking the results with this function.

Usage

```
plotRegionCoverage(
  regions,
  regionCoverage,
  groupInfo,
  nearestAnnotation,
  annotatedRegions,
  txdb = NULL,
  whichRegions = seq_len(min(100, length(regions))),
  colors = NULL,
  scalefac = 32,
  ask = interactive(),
  ylab = "Coverage",
  verbose = TRUE
)
```

Arguments

regions	The $regions output from calculatePvalues.
regionCoverage	The output from getRegionCoverage used on regions.
groupInfo	A factor specifying the group membership of each sample. It will be used to color the samples by group.
nearestAnnotation	The output from matchGenes used on regions.
annotatedRegions	The output from annotateRegions used on regions.
txdb	A TxDb object. If specified, transcript annotation will be extracted from this object and used to plot the transcripts.
whichRegions	An integer vector with the index of the regions to plot.
colors	If NULL then brewer.pal with the 'Dark2' color scheme is used.
scalefac	The parameter used in preprocessCoverage.
ask	If TRUE then the user is prompted before each plot is made.
ylab	The name of the of the Y axis.
verbose	If TRUE basic status updates will be printed along the way.

Value

A plot for every region showing the coverage of each sample at each base of the region as well as the summarized annotation information.

Author(s)

Andrew Jaffe, Leonardo Collado-Torres

See Also

calculatePvalues, getRegionCoverage, matchGenes, annotateRegions, plotCluster

Examples

```
## Load data
library("derfinder")

## Annotate regions, first two regions only
regions <- genomeRegions$regions[1:2]
annotatedRegions <- annotateRegions(
    regions = regions,
    genomicState = genomicState$fullGenome, minoverlap = 1
)

## Find nearest annotation with bumphunter::matchGenes()
library("bumphunter")
library("TxDb.Hsapiens.UCSC.hg19.knownGene")
genes <- annotateTranscripts(txdb = TxDb.Hsapiens.UCSC.hg19.knownGene)
```

```
nearestAnnotation <- matchGenes(x = regions, subject = genes)

## Obtain fullCov object
fullCov <- list("21" = genomeDataRaw$coverage)

## Assign chr lengths using hg19 information
library("GenomicRanges")
seqlengths(regions) <- seqlengths(getChromInfoFromUCSC("hg19",
    as.Seqinfo = TRUE
))[
    mapSeqlevels(names(seqlengths(regions)), "UCSC")
]

## Get the region coverage
regionCov <- getRegionCoverage(fullCov = fullCov, regions = regions)
#
## Make plots for the regions
plotRegionCoverage(
    regions = regions, regionCoverage = regionCov,
    groupInfo = genomeInfo$pop, nearestAnnotation = nearestAnnotation,
    annotatedRegions = annotatedRegions, whichRegions = 1:2
)

## Re-make plots with transcript information
plotRegionCoverage(
    regions = regions, regionCoverage = regionCov,
    groupInfo = genomeInfo$pop, nearestAnnotation = nearestAnnotation,
    annotatedRegions = annotatedRegions, whichRegions = 1:2,
    txdb = TxDb.Hsapiens.UCSC.hg19.knownGene
)
## Not run:
## If you prefer, you can save the plots to a pdf file
pdf("ders.pdf", h = 6, w = 9)
plotRegionCoverage(
    regions = regions, regionCoverage = regionCov,
    groupInfo = genomeInfo$pop, nearestAnnotation = nearestAnnotation,
    annotatedRegions = annotatedRegions, whichRegions = 1:2,
    txdb = TxDb.Hsapiens.UCSC.hg19.knownGene, ask = FALSE
)
dev.off()

## End(Not run)
```

---

vennRegions *Venn diagram for annotated regions given the genomic state*

Description

Makes a venn diagram for the regions given the genomic state showing how many regions overlap introns, exons, intergenic regions, none or multiple groups.

Usage

```
vennRegions(annotatedRegions, subsetIndex = NULL, ...)
```

Arguments

annotatedRegions	The output from annotateRegions used on regions.
subsetIndex	A vector of to use to subset the regions to use for the venn diagram. It can be a logical vector of length equal to the number of regions or an integer vector. If NULl, then it's ignored.
...	Arguments passed to vennDiagram.

Value

Makes a venn diagram plot for the annotation given the genomic state and the actual venn counts used to make the plot.

Author(s)

Leonardo Collado-Torres

See Also

annotateRegions, vennCounts, vennDiagram

Examples

```
## Load data
library("derfinder")

## Annotate regions
annotatedRegions <- annotateRegions(
    regions = genomeRegions$regions,
    genomicState = genomicState$fullGenome, minoverlap = 1
)

## Make venn diagram
venn <- vennRegions(annotatedRegions)

## Add title and choose text color
venn2 <- vennRegions(annotatedRegions,
    main = "Venn diagram", counts.col =
        "blue"
)

## Subset to only significant regions, so you don't have to annotate them
## again
venn3 <- vennRegions(annotatedRegions,
    subsetIndex =
        genomeRegions$regions$significant == "TRUE", main = "Significant only"
)
```

Index

annotateRegions, *7*, *9*

brewer.pal, *7*

calculatePvalues, *2–7*
countOverlaps, *6*

element_text, *5*
extendedMapSeqlevels, *5*

findOverlaps, *6*

getRegionCoverage, *7*

loadCoverage, *3*

makeGenomicState, *6*
matchGenes, *3–5*, *7*

plotCluster, 2, *6*, *7*
plotIdeogram, *3*
plotOverview, 4
plotRegionCoverage, 6
preprocessCoverage, *3*, *7*

theme, *5*
TxDb, *7*

vennCounts, *9*
vennDiagram, *9*
vennRegions, 8