Package ‘vulcan’

May 30, 2024

Type Package

Title VirtUal ChIP-Seq data Analysis using Networks

Version 1.26.0

Author Federico M. Giorgi, Andrew N. Holding, Florian Markowetz

Maintainer Federico M. Giorgi <federico.giorgi@gmail.com>

Description Vulcan (VirtUal ChIP-Seq Analysis through Networks) is a package that interrogates gene regulatory networks to infer cofactors significantly enriched in a differential binding signature coming from ChIP-Seq data. In order to do so, our package combines strategies from different BioConductor packages: DESeq for data normalization, ChIPpeakAnno and DiffBind for annotation and definition of ChIP-Seq genomic peaks, csaw to define optimal peak width and viper for applying a regulatory network over a differential binding signature.

License LGPL-3

LazyData TRUE

biocViews SystemsBiology, NetworkEnrichment, GeneExpression, ChIPSeq

NeedsCompilation no

Suggests vulcandata

Depends R (>= 4.0), ChIPpeakAnno,TxDb.Hsapiens.UCSC.hg19.knownGene, zoo, GenomicRanges, S4Vectors, viper, DiffBind, locfit

Imports wordcloud, csaw, gplots, stats, utils, caTools, graphics, DESeq2, Biobase

Encoding UTF-8

RoxygenNote 7.1.1

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

git_branch RELEASE_3_19

git_last_commit 703a294

git_last_commit_date 2024-04-30

Repository Bioconductor 3.19

Date/Publication 2024-05-29
Define the average fragment length

A function to get average fragment length from a ChIP-Seq experiment

Usage

average_fragment_length(bam.files, plot = TRUE, max.dist = 550)

Arguments

bam.files a vector of BAM files locations
plot logical. Should a plot be generated?
max.dist numeric. Maximum fragment length accepted. Default=550
**corr2p**

**Value**

nothing

**Examples**

```r
library(vulcandata)
sheetfile<-'deleteme.csv'
vulcandata::vulcansheet(sheetfile)
a<-read.csv(sheetfile,as.is=TRUE)
bams<-a$bamReads
unlink(sheetfile)
average_fragment_length(bams,plot=TRUE)
```

---

**corr2p**

*Convert correlation coefficient to p-value*

**Description**

This function converts an R value from a correlation calculation into a p-value, using a T distribution with the provided number of samples N minus 2 degrees of freedom.

**Usage**

```r
corr2p(r, N)
```

**Arguments**

- `r`: a correlation coefficient
- `N`: a number of samples

**Value**

- `p`: a p-value

**Examples**

```r
set.seed(1)
a<-rnorm(1000)
b<-a+rnorm(1000, sd=10)
r<-cor(a,b,method='pearson')
corr2p(r,N=length(a))
```
densityauc - Calculate the AUC of a density object

Description
This function will calculate the AUC of a density object generated by the 'density' function.

Usage
densityauc(dens, window)

Arguments
- dens: a density object
- window: a vector with two values, specifying the left and right borders for the AUC to be calculated

Value
a numeric value for the density AUC

Examples
set.seed(1)
a<-rnorm(1000)
d<-density(a)
window<-c(2,3)
da<-densityauc(d,window)

plot(d,main='')
abline(v=window,lty=2)
title(paste0('AUC between lines=',da))

dpareto - Probability density of Pareto distributions

Description
Gives NA on values below the threshold

Usage
dpareto(x, threshold = 1, exponent, log = FALSE)
\textbf{fisherp}

\textit{Fisher integration of p-values}

\textbf{Description}

This function applies the Fisher integration of p-values

\textbf{Usage}

\texttt{fisherp(ps)}

\textbf{Arguments}

\texttt{ps} \hspace{1cm} a vector of p-values

\textbf{Value}

\texttt{p.val} \hspace{1cm} an integrated p-value

\textbf{Examples}

\texttt{ps<-c(0.01,0.05,0.03,0.2)}
\texttt{fisherp(ps)}
Description

This function performs Gene Set Enrichment Analysis

Usage

gsea(
  reflist,  # named vector of reference scores
  set,      # element set
  method = c("permutation", "pareto"),  # one of 'permutation' or 'pareto'
  np = 1000,  # Number of permutations (Default: 1000)
  w = 1,      # exponent used to raise the supplied scores. Default is 1 (original scores unchanged)
  gsea_null = NULL  # a GSEA null distribution (Optional)
)

Arguments

  reflist  named vector of reference scores
  set      element set
  method   one of 'permutation' or 'pareto'
  np       Number of permutations (Default: 1000)
  w        exponent used to raise the supplied scores. Default is 1 (original scores unchanged)
  gsea_null a GSEA null distribution (Optional)

Value

A GSEA object. Basically a list of s components:

  ES   The enrichment score
  NES  The normalized enrichment score
  ledge The items in the leading edge
  p.value The permutation-based p-value

Examples

gsea<-setNames(-sort(rnorm(1000)),paste0('gene',1:1000))
set<-paste0('gene',sample(1:200,50))
obj<-gsea(reflist=set,method='pareto',np=1000)
obj$p.value
**kmgformat**

**kmgformat - Nice Formatting of Numbers**

**Description**

This function will convert thousand numbers to K, millions to M, billions to G, trillions to T, quadrillions to P

**Usage**

\[
\text{kmgformat}(\text{input}, \text{roundParam} = 1)
\]

**Arguments**

- **input** A vector of values
- **roundParam** How many decimal digits you want

**Value**

A character vector of formatted number names

**Examples**

- **# Thousands**
  ```r
  set.seed(1)
  a<-runif(1000,0,1e4)
  plot(a,yaxt='n')
  kmg<-kmgformat(pretty(a))
  axis(2,at=pretty(a),labels=kmg)
  ```
- **# Millions to Billions**
  ```r
  set.seed(1)
  a<-runif(1000,0,1e9)
  plot(a,yaxt='n',pch=20,col=val2col(a))
  kmg<-kmgformat(pretty(a))
  axis(2,at=pretty(a),labels=kmg)
  ```

---

**null_gsea**

**Calculate Null Distribution for GSEA**

**Description**

This function generates a GSEA null distribution from

**Usage**

\[
\text{null\_gsea}(\text{set}, \text{reflist}, w = 1, np = 1000)
\]
Arguments

- **set**
  - A vector containing gene names.

- **reflist**
  - A named vector containing the weights of the entire signature

- **w**
  - Exponent used to raise the supplied scores. Default is 1 (original scores unchanged)

- **np**
  - Number of permutations (Default: 1000)

Value

A vector of null scores appropriate for the set/reflist combination provided

Examples

```r
reflist<-setNames(-sort(rnorm(26)),LETTERS)
set<-c(‘A’,’B’,’D’,’F’)
nulldist<-null_gsea(set,reflist)
nulldist[1:10]
```

---

**p2corr**

Convert p-value to correlation coefficient

Description

This function converts an p-value into the corresponding correlation coefficient, using a T distribution with the provided number of samples N minus 2 degrees of freedom

Usage

```r
p2corr(p, N)
```

Arguments

- **p**
  - A p-value

- **N**
  - A number of samples

Value

- **r**
  - A correlation coefficient

Examples

```r
N<-100
p<-0.05
p2corr(p,N)
```
p2z

Description
This function gives a gaussian Z-score corresponding to the provided p-value. Careful: sign is not provided.

Usage
p2z(p)

Arguments
p a p-value

Value
z a Z score

Examples
p<-0.05
p2z(p)

pareto.fit
Estimate parameters of Pareto distribution

Description
A wrapper for functions implementing actual methods.

Usage
pareto.fit(data, threshold)

Arguments
data data vector, lower threshold (or 'find', indicating it should be found from the data), method (likelihood or regression, defaulting to former)
threshold numeric value to define the start of the tail

Value
List indicating type of distribution ('pareto'), parameters, information about fit (depending on method), OR a warning and NA if method is not recognized.
Examples

# Estimate the tail of a population normally distributed
set.seed(1)
x<-rnorm(1000)
q95<-as.numeric(quantile(abs(x),0.95))
fit<-pareto.fit(abs(x),threshold=q95)
# We can infer the p-value of a value very much right to the tail of the
# distribution
value<-5
pvalue<-ppareto(value, threshold=q95, exponent=fit$exponent,
lower.tail=FALSE)/20
plot(density(abs(x)),xlim=c(0,value+0.3),main='
Pareto fit'
)
arrows(value,0.2,value,0)
text(value,0.2,labels=paste0('p=',signif(pvalue,2)))

plot_gsea

Plot GSEA results

Description

This function generates a GSEA plot from a gsea object

Usage

plot_gsea(
  gsea.obj,
  twoColors = c("red", "blue"),
  plotNames = FALSE,
  colBarcode = "black",
  title = "Running Enrichment Score",
  bottomYtitle = "List Values",
  bottomYlabel = "Signature values",
  ext_nes = NULL,
  omit_middle = FALSE
)

Arguments

  gsea.obj GSEA object produced by the gsea function
  twoColors the two colors to use for positive[1] and negative[2] enrichment scores
  plotNames Logical. Should the set names be plotted?
  colBarcode The color of the barcode
  title String to be plotted above the Running Enrichment Score
  bottomYtitle String for the title of the bottom part of the plot
  bottomYlabel String for the label
  ext_nes Provide a NES from an external calculation
  omit_middle If TRUE, will not plot the running score (FALSE by default)
**ppareto**

**Value**

Nothing. a plot is generated in the default output device

**Examples**

```r
reflist<-setNames(-sort(rnorm(26)),LETTERS)
set<-c('A','B','D','F')
obj<-gsea(reflist,set,method='pareto')
plot_gsea(obj)
```

---

**ppareto**

*Cumulative distribution function of the Pareto distributions*  
' Gives NA on values < threshold

**Description**

Cumulative distribution function of the Pareto distributions  
' Gives NA on values < threshold

**Usage**

```r
ppareto(x, threshold = 1, exponent, lower.tail = TRUE)
```

**Arguments**

- `x`  
  Data vector, lower threshold, scaling exponent, usual flags
- `threshold`  
  numeric value to define the start of the tail
- `exponent`  
  the exponent obtained from the pareto.fit function
- `lower.tail`  
  logical. If the lower tail of the distribution should be considered. Default is TRUE

**Value**

Vector of (log) probabilities

**Examples**

```r
# Estimate the tail of a pospulation normally distributed
set.seed(1)
x<-rnorm(1000)
q95<as.numeric(quantile(abs(x),0.95))
fit<-pareto.fit(abs(x),threshold=q95)
# We can infer the pvalue of a value very much right to the tail of the
distribution
value<-5
pvalue<-ppareto(value, threshold=q95, exponent=fit$exponent,
lower.tail=FALSE)/20
plot(density(abs(x)),xlim=c(0,value+0.3),main='Pareto fit')
arrows(value,0.2,value,0)
text(value,0.2,labels=paste0('p=',signif(pvalue,2)))
```
**rea**  

**Description**  

REA: Rank Enrichment Analysis

REA Calculates enrichment of groups of objects over a vector of values associated to a population of objects.

**Usage**

```r
rea(signatures, groups, sweights = NULL, gweights = NULL, minsize = 1)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>signatures</td>
<td>a named vector, with values as signature values (e.g. logFC) and names as object names (e.g. gene symbols)</td>
</tr>
<tr>
<td>groups</td>
<td>a list of vectors of objects (e.g. pathways)</td>
</tr>
<tr>
<td>sweights</td>
<td>weights associated to objects in the signature. If NULL (default) all objects are treated according to the signature rank</td>
</tr>
<tr>
<td>gweights</td>
<td>weights associated to association strength between each object and each group. If NULL (default) all associations are treated equally</td>
</tr>
<tr>
<td>minsize</td>
<td>integer. Minimum size of the groups to be analyzed. Default=1</td>
</tr>
</tbody>
</table>

**Value**

A numeric vector of normalized enrichment scores

**Examples**

```r
signatures<-setNames(-sort(rnorm(1000)),paste0('gene',1:1000))
set1<-paste0('gene',sample(1:200,50))
set2<-paste0('gene',sample(1:1000,50))
groups<-list(set1=set1,set2=set2)
obj<-rea(signatures=signatures,groups=groups)
obj
```

---

**slice**  

**Description**

This function prints a slice of a matrix

**Usage**

```r
slice(matrix)
```
**stouffer**

**Arguments**

- matrix A matrix

**Value**

prints it

**Examples**

```r
set.seed(1)
example<-matrix(rnorm(1000),nrow=100,ncol=10)
slice(example)
```

---

**Description**

This function gives a gaussian Z-score corresponding to the provided p-value Careful: sign is not provided

**Usage**

`stouffer(x)`

**Arguments**

- `x` a vector of Z scores

**Value**

- `Z` an integrated Z score

**Examples**

```r
zs<-c(1,3,5,2,3)
stouffer(zs)
```
textplot2 - An x y plot of non-overlapping text

Description

This function is an extension of the 'textplot' function from the 'wordcloud' package, with the extra functionality of specifying the color of the points.

Usage

```r
textplot2(  
  x,  
  y,  
  words,  
  cex = 1,  
  pch = 16,  
  pointcolor = "#FFFFFF00",  
  new = TRUE,  
  show.lines = TRUE,  
  ...  
)
```

Arguments

- `x`: x coordinates
- `y`: y coordinates
- `words`: the text to plot
- `cex`: font size
- `pch`: pch parameter for the plotted points
- `pointcolor`: a string specifying the color of the points (default #FFFFFF00)
- `new`: should a new plot be created
- `show.lines`: if true, then lines are plotted between x,y and the word, for those words not covering their x,y coordinates
- `...`: Additional parameters to be passed to wordlayout and text.

Value

nothing

Examples

```r
obj_names<-apply(expand.grid(LETTERS,LETTERS),1,paste,collapse='')
a<-setNames(runif(26*26),obj_names)
b<-setNames(rnorm(26*26),obj_names)
plot(a,b,pch=20,col='grey')
top<-names(sort(-a))[1:50]
textplot2(a[top],b[top],words=top,new=FALSE,pointcolor='black')
```
val2col

Convert a numeric vector into colors

Description

Convert a numeric vector into colors

Usage

val2col(
  z,
  col1 = "navy",
  col2 = "white",
  col3 = "red3",
  nb breaks = 100,
  center = TRUE,
  rank = FALSE
)

Arguments

z a vector of numbers

col1 a color name for the min value, default 'navy'

col2 a color name for the middle value, default 'white'

col3 a color name for the max value, default 'red3'

nb breaks Number of colors to be generated. Default is 30.

center boolean, should the data be centered? Default is TRUE

rank boolean, should the data be ranked? Default is FALSE

Value

a vector of colors

Examples

a <- rnorm(1000)
cols <- val2col(a)
plot(a, col = cols, pch = 16)
**vulcan**

**VULCAN - VirtUaL Chipseq data Analysis using Networks**

**Description**

This function calculates the enrichment of a gene regulatory network over a ChIP-Seq derived signature.

**Usage**

```
vulcan(vobj, network, contrast, annotation = NULL, minsize = 10)
```

**Arguments**

- **vobj**: a list, the output of the 'vulcan.normalize' function
- **network**: an object of class 'viper::regulon'
- **contrast**: a vector of two fields, containing the condition names to be compared (1 vs 2)
- **annotation**: an optional named vector to convert gene identifiers (e.g. entrez ids to gene symbols) Default (NULL) won’t convert gene names.
- **minsize**: integer indicating the minimum regulon size for the analysis to be run. Default: 10

**Value**

A list of components:

- **peakcounts**: A matrix of raw peak counts, peaks as rows, samples as columns
- **peakrpkms**: A matrix of peak RPKMs, peaks as rows, samples as columns
- **rawcounts**: A matrix of raw gene counts, genes as rows, samples as columns. The counts are associated to the promoter region of the gene
- **rpkms**: A matrix of RPKMs, genes as rows, samples as columns. The RPKMs are associated to the promoter region of the gene
- **normalized**: A matrix of gene abundances normalized by Variance-Stabilizing Transformation (VST), genes as rows, samples as columns. The abundances are associated to the promoter region of the gene
- **samples**: A vector of sample names and conditions
- **msviper**: a multisample virtual proteomics object from the viper package
- **mrs**: A table of master regulators for a specific signature, indicating their Normalized Enrichment Score (NES) and p-value
vulcan.annotate

**Examples**

```r
library(vulcandata)
# Get an example vulcan object (generated with vulcan.import() using the
dummy dataset contained in the \textit{vulcandata} package)
vobj<-vulcandata::vulcanexample()
# Annotate peaks to gene names
vobj<-vulcan.annotate(vobj,lborder=-10000,rborder=10000,method='sum')
# Normalize data for VULCAN analysis
vobj<-vulcan.normalize(vobj)
# Detect which conditions are present
names(vobj$samples)

# Load an ARACNe network
# This is a regulon object as specified in the VIPER package, named 'network'
load(system.file('extdata','network.rda',package='vulcandata',mustWork=TRUE))
# Run VULCAN
# We can reduce the minimum regulon size, since in this example only one
# chromosome
# was measured, and the networks would otherwise have too few hits
vobj_analysis<-vulcan(vobj,network=network,contrast=c('t90','t0'),minsize=5)
# Visualize output using the msviper plotting function
plot(vobj_analysis$msviper,mrs=7)
```

---

**vulcan.annotate**  
*Function to annotate peaks for VULCAN analysis*

**Description**

This function coalesces and annotates a set of BAM files into peak-centered data. It implements the ChIPPeAAnno methods, with specific choices dealing with defining the genomic area around the promoter and which peaks to include.

**Usage**

```r
vulcan.annotate(
  vobj, 
  lborder = -10000, 
  rborder = 10000, 
  method = c("closest", "strongest", "sum", "topvar", "farthest", "lowvar"), 
  TxDb = NULL 
)
```

**Arguments**

- **vobj**  
  A list of peakcounts, samples and peakrpkms (i.e. the output of the function vulcan.import)

- **lborder**  
  Boundary for peak annotation (in nucleotides) upstream of the Transcription starting site (default: -10000)
vulcan.annotate

rborder Boundary for peak annotation (in nucleotides) downstream of the Transcription starting site (default: 10000)

method Method to deal with multiple peaks found within gene promoter boundaries. One of sum (default), closest, strongest, topvar, farthest or lowvar. This will affect only genes with multiple possible peaks. When a single peak can be mapped to the promoter region of the gene, that peak abundance will be considered as the gene promoter’s occupancy.

sum when multiple peaks are found, sum their contributions

closest when multiple peaks are found, keep only the closest to the TSS as the representative one

strongest when multiple peaks are found, keep the strongest as the representative one

farthest when multiple peaks are found, keep only the closest to the TSS as the representative one

topvar when multiple peaks are found, keep the most varying as the representative one

lowvar when multiple peaks are found, keep the least varying as the representative one

TxDb TxDb annotation object containing the knownGene track. If NULL (the default), TxDb.Hsapiens.UCSC.hg19.knownGene is loaded

Value

A list of components:

peakcounts A matrix of raw peak counts, peaks as rows, samples as columns

peakrpkms A matrix of peak RPKMs, peaks as rows, samples as columns

rawcounts A matrix of raw gene counts, genes as rows, samples as columns. The counts are associated to the promoter region of the gene

rpkms A matrix of RPKMs, genes as rows, samples as columns. The RPKMs are associated to the promoter region of the gene

samples A vector of sample names and conditions

Examples

library(vulcandata)
obj<-vulcan.example()
obj<-vulcan.annotate(obj, lborder=-10000, rborder=10000, method='sum')
vulcan.import

Function to import BAM files

Description

This function coalesces and annotates a set of BAM files into peak-centered data.

Usage

vulcan.import(sheetfile, intervals = NULL)

Arguments

sheetfile  path to a csv annotation file containing sample information and BAM location
intervals   size of the peaks. If NULL (default) it is inferred from the average fragment length observed in the dataset

Value

A list of components:

peakcounts  A matrix of raw peak counts, peaks as rows, samples as columns
peakrpkms   A matrix of peak RPKMs, peaks as rows, samples as columns
samples     A vector of sample names and conditions

Examples

library(vulcandata)
# Generate an annotation file from the dummy ChIP-Seq dataset
vfile<-tempfile()
vulcandata::vulcansheet(vfile)
# Import BAM and BED information into a list object
# vobj<-vulcan.import(vfile)
# This vobj is identical to the object returned by
# vulcandata::vulcanexample()
unlink(vfile)
vulcan.normalize  

Function to normalize promoter peak data

Description

This function normalizes gene-centered ChIP-Seq data using VST

Usage

vulcan.normalize(vobj)

Arguments

vobj  
a list, the output of the 'vulcan.annotate' function

Value

A list of components:

- **peakcounts**  
  A matrix of raw peak counts, peaks as rows, samples as columns

- **peakrpkms**  
  A matrix of peak RPKMs, peaks as rows, samples as columns

- **rawcounts**  
  A matrix of raw gene counts, genes as rows, samples as columns. The counts are associated to the promoter region of the gene

- **rpkms**  
  A matrix of RPKMs, genes as rows, samples as columns. The RPKMs are associated to the promoter region of the gene

- **normalized**  
  A matrix of gene abundances normalized by Variance-Stabilizing Transformation (VST), genes as rows, samples as columns. The abundances are associated to the promoter region of the gene

- **samples**  
  A vector of sample names and conditions

Examples

```r
# Not run:
library(vulcandata)
vobj<-vulcandata::vulcanexample()
vobj<-vulcan.annotate(vobj, lborder=-10000, rborder=10000, method='sum')
vobj<-vulcan.normalize(vobj)

# End(Not run)
```
vulcan.pathways  

Function to calculate pathway enrichment over a ChIP-Seq profile

Description
This function applies Gene Set Enrichment Analysis or Rank Enrichment Analysis over a ChIP-Seq signature contained in a vulcan package object.

Usage
vulcan.pathways(
  vobj,
  pathways,
  contrast = NULL,
  method = c("GSEA", "REA"),
  np = 1000
)

Arguments
vobj  a list, the output of the 'vulcan.annotate' function
pathways  a list of vectors, one vector of gene identifiers per pathway
contrast  a vector with the name of the two conditions to compare. If method=='REA', contrast can be set to 'all', and Rank Enrichment Analysis will be performed for every sample independently, compared to the mean of the dataset.
method  either 'REA' for Rank Enrichment Analysis or 'GSEA' for Gene Set Enrichment Analysis
np  numeric, only for GSEA, the number of permutations to build the null distribution. Default is 1000

Value
if method=='GSEA', a named vector, with pathway names as names, and the normalized enrichment score of either the GSEA as value. If method=='REA', a matrix, with pathway names as rows and specific contrasts as columns (the method 'REA' allows for multiple contrasts to be calculated at the same time)

Examples
library(vulcandata)
vfile<- tempfile()
vulcandata::vulcansheet(vfile)
#vobj<-vulcan.import(vfile)
vobj<-vulcandata::vulcanexample()
unlink(vfile)
vobj<-vulcan.annotate(vobj,lborder=-10000,rborder=10000,method='sum')
vobj<-vulcan.normalize(vobj)
# Create a dummy pathway list (in entrez ids)
pathways<-list(
    pathwayA=rownames(vobj$normalized)[1:20],
    pathwayB=rownames(vobj$normalized)[21:50]
)
# Which contrast groups can be queried
names(vobj$samples)
results_gsea<-vulcan.pathways(vobj,pathways,contrast=c('t90', 't0'),
    method='GSEA')
results_rea<-vulcan.pathways(vobj,pathways,contrast=c('all'),method='REA')

---

### wstouffer

**Weighted Stouffer integration of Z scores**

**Description**

This function gives a gaussian Z-score corresponding to the provided p-value. Careful: sign is not provided.

**Usage**

```
wstouffer(x, w)
```

**Arguments**

- `x` a vector of Z scores
- `w` weight for each Z score

**Value**

- `Z` an integrated Z score

**Examples**

```
zs<-c(1,-3,5,2,3)
ws<-c(1,10,1,2,1)
wstouffer(zs,ws)
```
Description
This function gives a gaussian p-value corresponding to the provided Z-score

Usage
z2p(z)

Arguments
z  a Z score

Value
a p-value

Examples
z<-1.96
z2p(z)
Index

average_fragment_length, 2

corr2p, 3

densityauc, 4
dpareto, 4

fisherp, 5

gsea, 6

kmgformat, 7

null_gsea, 7

p2corr, 8
p2z, 9
pareto.fit, 9
plot_gsea, 10
ppareto, 11

rea, 12

slice, 12
stouffer, 13

textplot2, 14

val2col, 15
vulcan, 16
vulcan.annotate, 17
vulcan.import, 19
vulcan.normalize, 20
vulcan.pathways, 21

wstouffer, 22

z2p, 23