Package 'DEWSeq'

April 23, 2024

Type Package

Title Differential Expressed Windows Based on Negative Binomial Distribution

Version 1.17.2

Description DEWSeq is a sliding window approach for the analysis of differentially enriched binding regions eCLIP or iCLIP next generation sequencing data.

Imports BiocGenerics, data.table(>= 1.11.8), GenomeInfoDb, GenomicRanges, methods, S4Vectors, SummarizedExperiment, stats, utils

Depends R(>= 4.0.0), R.utils, DESeq2, BiocParallel

Suggests knitr, tidyverse, rmarkdown, testthat, BiocStyle, IHW

VignetteBuilder knitr

biocViews Sequencing, GeneRegulation, FunctionalGenomics, DifferentialExpression

License LGPL (>= 3)

URL https://github.com/EMBL-Hentze-group/DEWSeq/

Encoding UTF-8

LazyData false

RoxygenNote 7.1.2

BugReports https://github.com/EMBL-Hentze-group/DEWSeq/issues

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

git_branch devel

git_last_commit 19e182d

git_last_commit_date 2024-01-19

Repository Bioconductor 3.19

Date/Publication 2024-04-23

Author Sudeep Sahadevan [aut],
Thomas Schwarzl [aut],
bioinformatics team Hentze [aut, cre]

Maintainer bioinformatics team Hentze <biohentze@embl.de>

Contents

`.readAnnotation` *read annotation data*

Description

read annotation data for windows This is an unexported lowlevel function to read annotations from a file. The file MUST be tab separated and MUST have the following columns:
chromosome: chromosome name
unique_id: unique id of the window
begin: window start co-ordinate
end: window end co-ordinate
strand: strand
gene_id: gene id
gene_name: gene name
gene_type: gene type annotation
gene_region: gene region
Nr_of_region: number of the current region
Total_nr_of_region: total number of regions
window_number: window number

Usage

```
.readAnnotation(
  fname,
  uniqIds = NULL,
  asGRange = TRUE,
  checkWindowNumber = TRUE,
  start0based = TRUE,
  threads = 10
)
```

Arguments

fname	character, file name/path
uniqIds	character vector, filter stable and keep annotation for these unique ids
asGRange	logical, boolean, whether to reaturn a GRange object or a data.frame object
checkWindowNumber	logical, check window number
start0based	logical, TRUE (default) or FALSE. If TRUE, then the start
threads	numeric, number of threads for fread (default: 10) positions are considered to be 0-based

Value

data.frame or GRanges object

DESeqDataSetFromSlidingWindows
create DESeq data object

Description

create DESeq data object from sliding window counts, phenotype data and annotation data

Usage

```
DESeqDataSetFromSlidingWindows(
  countData,
  colData,
  annotObj,
  design,
  tidy = FALSE,
  ignoreRank = FALSE,
  start0based = TRUE
)
```

Arguments

countData	data.frame or matrix, sliding window count data
colData	DataFrame or data.frame, phenotype data, see DESeqDataSet
annotObj	data.frame or character, can either be a data.frame or a file name, see details
design	formula or matrix, design of the experiment, see DESeqDataSet
tidy	logical, If TRUE, first column is of countData is treated as rownames (defalt: FALSE), see DESeqDataSet
ignoreRank	logical, ignore rank, see DESeqDataSet
start0based	logical, TRUE (default) or FALSE. If TRUE, then the start positions in annotObj is considered to be 0-based

Details

If annotObj is a file name, the input file MUST be <TAB> separated, and supports reading in .gz files.
If annotObj is a data.frame, colnames(annotObj) MUST not be empty.
This function checks for the following columns after reading in the file or on data.frame:

- chromosome: chromosome name
- unique_id: unique id of the window, rownames(object) must match this column
- begin: window start co-ordinate, see parameter start0based
- end: window end co-ordinate
- strand: strand
- gene_id: gene id
- gene_name: gene name
- gene_type: gene type annotation
- gene_region: gene region
- Nr_of_region: number of the current region
- Total_nr_of_region: total number of regions
- window_number: window number

This function creates a DESeqDataSet using supplied countData, phenotype data and annotation data. The chromosomal locations and annotations of the sliding windows (parsed from annotObj) can be accessed from the returned object using: rowRanges(object)

Value

DESeq object

Examples

```
data("SLBP_K562_w50s20")
slbpDat <- counts(SLBP_K562_w50s20)
phenoDat <- DataFrame(conditions=as.factor(c(rep('IP',2),'SMI')),
row.names = colnames(slbpDat))
phenoDat$conditions <- relevel(phenoDat$conditions,ref='SMI')
annotDat <- as.data.frame(rowRanges(SLBP_K562_w50s20))
# by default chromsome column is 'seqnames'
# and begin co-ordinate column is 'start'
# rename these columns
colnames(annotDat)[1:2] <- c('chromosome','begin')
slbpDds <- DESeqDataSetFromSlidingWindows(countData = slbpDat,
colData = phenoDat,annotObj = annotDat,design=~conditions)
```

`extractRegions` *extract significant regions*

Description

extract significant windows from output of `resultsDEWSeq` using the supplied padj and log2FoldChange cut-offs and merge these significant windows to regions and create the following columns for each significant region:

- `padj_min`: min. padj value in the region
- `padj_mean`: average padj value in the region
- `padj_max`: max. padj value in the region
- `log2FoldChange_min`: min. log 2 fold change in the region
- `log2FoldChange_mean`: average log 2 fold change in the region
- `log2FoldChange_max`: max. log 2 fold change in the region

Usage

```
extractRegions(
  windowRes,
  padjCol = "padj",
  padjThresh = 0.05,
  log2FoldChangeCol = "log2FoldChange",
  log2FoldChangeThresh = 1,
  start0based = TRUE
)
```

Arguments

`windowRes`	`data.frame`, output from `resultsDEWSeq`
`padjCol`	`character`, name of the adjusted pvalue column (default: padj)
`padjThresh`	`numeric`, threshold for p-adjusted value (default: 0.05)
`log2FoldChangeCol`	`character`, name of the log2foldchange column (default: log2FoldChange)
`log2FoldChangeThresh`	`numeric`, threshold for log2foldchange value (default:1)
`start0based`	`logical`, TRUE (default) or FALSE. If TRUE, then the start positions in `windowRes` is considered to be 0-based

Details

The output data.frame from this function will have the following columns:

- `chromosome`: chromosome name
- `regionStartId`: `unique_id` of the left most window, where an enriched region begins

- region_begin: starting position of the enriched region
- region_end: ending position of the enriched region
- strand: strand info
- windows_in_region: total number of windows that make up the enriched region
- region_length: length of the enrched region
- gene_id: gene id
- gene_name: gene name
- gene_type: gene type annotation
- gene_region: gene region
- Nr_of_region: number of the current region
- Total_nr_of_region: total number of regions
- window_number: window number
- padj_min: min. padj value in the region
- padj_mean: average padj value in the region
- padj_max: max. padj value in the region
- log2FoldChange_min: min. log 2 fold change in the region
- log2FoldChange_max: max. log 2 fold change in the region
- log2FoldChange_mean: average log 2 fold change in the region

Value

data.frame

Examples

```
data("slbpWindows")
# using default cut-off thresholds,
# 'pSlidingWindows.adj' padj value columns
slbpRegions <- extractRegions(slbpWindows,
padjCol = 'pSlidingWindows.adj')
```

---

filterCounts *filter count data*

Description

In addition to count data matrix, htseq-clip also creates a max count matrix.
For each window, this file contains the maximum crosslink site count (height) calculated per nucleotide. This function uses this file to filter the count data file instead of the default prefiltering on rowSums. Windows failing the threshold rowSums(maxWindowCount>=countThresh)>=nSamples will be removed from the object.

Usage

```
filterCounts(object, maxCountFile, countThresh, nsamples)
```

Arguments

object	DESeqDataSet, see DESeqDataSetFromSlidingWindows
maxCountFile	character file name/path to max count matrix
countThresh	numeric max count threshold
nsamples	numeric number of samples where the max count value must be >= countThresh

Value

DESeq object

---

resultsDEWSeq *extract DEWseq results*

Description

This is a modified version of the results function from DESeq2 package.

This function uses chromosomal positions given in the rowRanges(dds) to identify overlapping windows in dds object. For each window, the number of overlapping windows are counted, and the p-value is adjusted for FWER using bonferroni correction.

For further details, please refer documentation for results function in DESeq2 package

Usage

```
resultsDEWSeq(
  object,
  contrast,
  name,
  listValues = c(1, -1),
  cooksCutoff,
  test,
  addMLE = FALSE,
  tidy = FALSE,
  parallel = FALSE,
  BPPARAM = bpparam(),
  minmu = 0.5,
  start0based = TRUE
)
```

Arguments

object	DESeqDataSet, on which the following functions has already been called: nbinomWaldTest
contrast	character vector, list of 2 character vectors or numeric contrast vector contrast this argument specifies what comparison to extract from the object to build a results table, see results
name	character, name the name of the individual effect (coefficient) for building a results table. name argument is ignored if contrast is specified
listValues	list, check results for details of this parameter
cooksCutoff	numeric, theshold on Cook's distance
test	character, this is automatically detected internally if not provided.
addMLE	logical, if betaPrior=TRUE was used
tidy	logical, whether to output the results table with rownames as a first column 'row'. The table will also be coerced to data.frame
parallel	logical, if FALSE, no parallelization. if TRUE, parallel execution using BiocParallel, see next argument BPPARAM
BPPARAM	bpparamClass, an optional parameter object passed internally to bplapply when parallel=TRUE. If not specified, the parameters last registered with register will be used.
minmu	numeric, lower bound on the estimated count (used when calculating contrasts)
start0based	logical, TRUE (default) or FALSE. If TRUE, then the start positions in annotationFile are considered to be 0-based

Details

For a detailed description of the column use mcols(output)$description

Value

DESeqResults object

Examples

```
data("slbpDds")
slbpDds <- estimateSizeFactors(slbpDds)
slbpDds <- estimateDispersions(slbpDds)
slbpDds <- nbinomWaldTest(slbpDds)
slbpWindows <- resultsDEWSeq(slbpDds)

## Not run:
# for a description of the columns in slbpWindows use
mcols(slbpWindows)$description

## End(Not run)
```

slbpDds *ENCODE eCLIP data SLBP in K562*

Description

This is a DESeq dataset object for ENCODE eCLIP data: SLBP in K562 cell lines This is used as an example dataset for a runnable example. This dataset is the output from running the example code for the function DESeqDataSetFromSlidingWindows

Usage

```
data(slbpDds)
```

Format

An object of class "DESeq";

Examples

```
data(slbpDds)
slbpDds
```

slbpRegions *ENCODE eCLIP data SLBP in K562*

Description

This is a DESeq results object for ENCODE eCLIP data: SLBP in K562 cell lines This is used as an example dataset for a runnable example. This dataset is the output from running the example code for the function extractRegions

Usage

```
data(slbpRegions)
```

Format

data.frame;

Examples

```
data(slbpRegions)
head(slbpRegions)
```

slbpVst *ENCODE eCLIP data SLBP in K562*

Description

This is a DESeq normalized sliding window count matrix ENCODE eCLIP data: SLBP in K562 cell lines This is used as an example dataset for a runnable example. This dataset is the output from running the example code for the function `vst`

Usage

```
data(slbpVst)
```

Format

matrix;

Examples

```
data(slbpVst)
head(slbpVst)
```

slbpWindows *ENCODE eCLIP data SLBP in K562*

Description

This is a DESeq results object for ENCODE eCLIP data: SLBP in K562 cell lines This is used as an example dataset for a runnable example. This dataset is the output from running the example code for the function `resultsDEWSeq`

Usage

```
data(slbpWindows)
```

Format

data.frame;

Examples

```
data(slbpWindows)
head(slbpWindows)
```

SLBP_K562_w50s20 *ENCODE eCLIP data for SLBP in K562, low count filtered*

Description

This is ENCODE eCLIP data which was quantified by htseq-clip in sliding-windows of max. length 50nt, the step size was 20. This is not ideal data for DEWSeq since it is lacking replicates, however was small enough for the inclusion of the package.

Usage

```
data(SLBP_K562_w50s20)
```

Format

An object of class "DESeq";

Examples

```
data(SLBP_K562_w50s20)
SLBP_K562_w50s20
```

toBED *windows/regions to BED*

Description

given output of extractRegions, resultsDEWSeq and significance thresholds, extract significant windows, create regions by merging adjacent significant windows. Finally, write the output as a BED file for visualization.

Usage

```
toBED(
  windowRes,
  regionRes,
  fileName,
  padjCol = "padj",
  padjThresh = 0.05,
  log2FoldChangeCol = "log2FoldChange",
  log2FoldChangeThresh = 1,
  trackName = "sliding windows",
  description = "sliding windows"
)
```

Arguments

windowRes	data.frame, output from resultsDEWSeq
regionRes	data.frame, output from extractRegions
fileName	character, filename to save BED output
padjCol	character, name of the adjusted pvalue column (default: padj)
padjThresh	numeric, threshold for p-adjusted value (default: 0.05)
log2FoldChangeCol	character, name of the log2foldchange column (default: log2FoldChange)
log2FoldChangeThresh	numeric, threshold for log2foldchange value (default:1)
trackName	character, name of this track, for visualization
description	character, description of this track, for visualization

Value

write to file

Examples

```
data(slbpRegions)
data(slbpWindows)
outFile <- tempfile('SLBP_visualization.bed')
# the results are written to a temp file in this example
toBED(slbpWindows,slbpRegions,outFile,padjCol='pSlidingWindows.adj')
```

---

topWindowStats *stats for the top windows in each region*

Description

given window resutls and normalized counts, combine significant overlapping windows into regions and for each region, pick two candidate winodws:

1. with highest log2FoldChange and
2. with highest normalized mean in treatment samples (see parameter treatmentCols)

Return a data.frame with region information and stats, and for the selected windows, the following information:

- unique_id of the window
- start and end co-ordinates
- log2FoldChange
- normalized mean expression in treatment and control samples and
- individual normalized expression in replicates

Usage

```
topWindowStats(
  windowRes,
  padjCol = "padj",
  padjThresh = 0.05,
  log2FoldChangeCol = "log2FoldChange",
  log2FoldChangeThresh = 1,
  start0based = TRUE,
  normalizedCounts,
  treatmentCols,
  treatmentName = "treatment",
  controlName = "control",
  op = "max"
)
```

Arguments

windowRes	data.frame, output from resultsDEWSeq
padjCol	character, name of the adjusted pvalue column (default: padj)
padjThresh	numeric, threshold for p-adjusted value (default: 0.05)
log2FoldChangeCol	character, name of the log2foldchange column (default: log2FoldChange)
log2FoldChangeThresh	numeric, threshold for log2foldchange value (default:1)
start0based	logical, TRUE (default) or FALSE. If TRUE, then the start positions in windowRes is considered to be 0-based
normalizedCounts	data.frame or matrix, normalized read counts per window. rownames(normalizedCounts) and unique_id column from windoeRes must match see counts, vst or rlog
treatmentCols	character vector, column names in normalizedCounts for treatment/case samples. The remaining columns in the data.frame will be considered control samples
treatmentName	character, treatment name, see Details (default: treatment)
controlName	character, control name, see Details (default: control)
op	character, can be one of max (default) or min. max returns windows with maximum log2FoldChange and mean normalized expression in the treatmentCols columns, min returns windows with minimum log2FoldChange and mean normalized expression

Details

The output data.frame of this function has the following columns:

- chromosome: chromosome name
- gene_id: gene id
- gene_name: gene name

- gene_region: gene region
- gene_type: gene type annotation
- regionStartId: unique_id of the left most window, where a enriched region begins
- region_begin: start position of the enriched region
- region_end: end position of the enriched region
- region_length: length of the enrched region
- strand: strand info
- Nr_of_region: number of the current region
- Total_nr_of_region: total number of regions
- log2FoldChange_min: min. log 2 fold change in the region
- log2FoldChange_mean: average log 2 fold change in the region
- log2FoldChange_max: max. log 2 fold change in the region
- unique_id.log2FCWindow: unique_id of the window with largest log2FoldChange
- begin.log2FCWindow: start position of the window with largest log2FoldChange
- end.log2FCWindow: end of the window with largest log2FoldChange
- log2FoldChange.log2FCWindow: log2FoldChange of the window with largest log2FoldChange
- treatmentName.mean.log2FCWindow: mean of the normalized expression of the treatment samples for log2FCWindow, names in treatmentCols are used to calculate mean and treatmentName is from the parameter treatmentName
- controlName.mean.log2FCWindow: mean of the normalized expression of the control samples for log2FCWindow, colnames(normalizedCounts) not found in treatmentCols are used to calculate mean and controlName is from the parameter controlName
- the next columns will be normalized expression values of the log2FCWindow from individual treatment and control samples.
- unique_id.meanWindow: unique_id of the window with largest mean in all treatment samples from treatmentCols
- begin.meanWindow: start position of the mean window
- end.meanWindow: end position of the mean window
- log2FoldChange.meanWindow:log2FoldChange of the mean window
- treatmentName.mean.meanWindow: mean of the normalized expression of the treatment samples for meanWindow, names in treatmentCols are used to calculate mean and treatmentName is from the parameter treatmentName
- controlName.mean.meanWindow: mean of the normalized expression of the control samples for log2FCWindow, colnames(normalizedCounts) not found in treatmentCols are used to calculate mean and controlName is from the parameter controlName
- the next columns will be normalized expression values of the meanWindow from individual treatment and control samples

Value

data.frame

Examples

```
data(slbpWindows)
data(slbpVst)
slbpList <- topWindowStats(slbpWindows,padjCol = 'pSlidingWindows.adj',
normalizedCounts = slbpVst, treatmentCols = c('IP1','IP2'),
treatmentName = 'SLBP',controlName = 'SMI')
```

Index