Package 'ExCluster'

November 6, 2026

Title ExCluster robustly detects differentially expressed exons between two conditions of RNA-seq data, requiring at least two independent biological replicates per condition

Version 1.30.0

Author R. Matthew Tanner, William L. Stanford, and Theodore J. Perkins

Date 2018-10-02

Maintainer R. Matthew Tanner <rtann038@uottawa.ca>

Description ExCluster flattens Ensembl and GENCODE GTF files into GFF files, which are used to count reads per non-overlapping exon bin from BAM files. This read counting is done using the function featureCounts from the package Rsubread. Library sizes are normalized across all biological replicates, and ExCluster then compares two different conditions to detect signifcantly differentially spliced genes. This process requires at least two independent biological repliates per condition, and ExCluster accepts only exactly two conditions at a time. ExCluster ultimately produces false discovery rates (FDRs) per gene, which are used to detect significance. Exon log2 fold change (log2FC) means and variances may be plotted for each significantly differentially spliced gene, which helps scientists develop hypothesis and target differential splicing events for RT-qPCR validation in the wet lab.

biocViews ImmunoOncology, DifferentialSplicing, RNASeq, Software

Imports stats, methods, grDevices, graphics, utils

Depends Rsubread, GenomicRanges, rtracklayer, matrixStats, IRanges

License GPL-3

Encoding UTF-8

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

git_branch RELEASE_3_23

git_last_commit 144701a

git_last_commit_date 2026-04-28

Repository Bioconductor 3.23

Date/Publication 2026-11-05

Contents

ExCluster 2
GFF_convert 4
GRangesFromExClustResults 6
plotExonlog2FC 7
processCounts 8

Index **11**

`ExCluster` *ExCluster is the main function in this package, which clusters exon bins within each gene to determine signficiantly differentially expressed exon clusters.*

Description

The ExCluster function takes an input of normalized exon bin read counts per sample, and only accepts data from exactly two conditions at once, requiring two biological replicates per condition. ExCluster also requires GFF annotations to annotate exon bins

Usage

```
ExCluster(exon.Counts, cond.Nums, annot.GFF, GFF.File, out.Dir,
    result.Filename, combine.Exons=TRUE, plot.Results=FALSE, FDR.cutoff=0.05)
```

Arguments

`exon.Counts`	exon.Counts must be assigned a data frame of normalized read counts per exon bin, which are attained from running the processCounts function. For example, if the results of processCounts are assigned to the normCounts variable, exon.Counts is assigned as: exon.Counts=normCounts This input is required.
`cond.Nums`	cond.Nums must be assigned an array of sample numbers corresponding, in exact order, to each BAM file counted and passed into the exon.Counts argument. The length of cond.Nums must also match the number of columns in exon.Counts exactly. For example, analyzing data with 3 biological replicates per each of two conditions, in that order, would be denoted as: cond.Nums=c(1,1,1,2,2,2) cond.Nums must be given at least 2 biological replicates per condition, and exactly 2 conditions – otherwise, the ExCluster package will throw an error describing the problem. This is a required input.
`annot.GFF`	If this argument is specified, it must be given a GFF annotation data frame, which is generated from the GFF_convert function. For example, it can be specified as annot.GFF=GFF, assuming your GFF data frame is assigned to the GFF variable. Either annot.GFF or GFF.File must be specified. If both are specified, annot.GFF will take priority.

`GFF.File` If this argument is specified, it must be given a full file path to a GFF file, including file name and extension. For example, this may look like: /Users/username/path/to/file.gff This argument is not required, but either annot.GFF or GFF.File must be specified.

`out.Dir` The out.Dir argument should be assigned a character string specifying a full folder path where results may be written. This should be a specific folder to the data being analyzed, to avoid writing results from different analyses to the same folder.This argument is not required, but it is STRONGLY recommended that you write out the results of the ExCluster analysis, as this portion of the package can take well over 2 hours to complete. An example path for out.Dir would be: /Users/username/Documents/RNA-seq/project_A/

`result.Filename` result.Filename should be given a character string specifying the specific filename for the ExCluster results table to be written to, within the out.Dir folder. File extension specification is not necessary, as '.txt.' will be added to the result.Filename regardless. This argument is not required, although it may be helpful to name your results by including both condition names, so the specific comparison made is easily identifiable at a later point in time. By default, result.Filename will be assigned the value "ExClust_Results". An example filename that may be helpful could be: "ExClust_res_GeneA_shRNA_vs_scramble"

`combine.Exons` combine.Exons should be assigned a logical value of either TRUE or FALSE. This denotes whether exon bins which are always co-expressed in the same transcripts should be combined into 'super-exons'. Doing this can be helpful to increase exon read depth, and it greatly reduces computation time. However, this should only be done in a standard RNA-seq analysis, when no instances of aberrant splicing are predicted. If one suspects aberrant splicing in one of your conditions, this argument should not be set. By default combine.Exons=FALSE

`plot.Results` plot.Results should be assigned a logical value of either TRUE or FALSE. This determines whether or not the ExCluster function should automatically run the plotExonlog2FC function, and plot exon bin log2FCs for each significantly differentially expressed gene. It is generally helpful to run this alongside your analysis, as it saves time. However, your ExCluster results can be saved and read back into R at a later date, from which you can run the plotExonlog2FC function separately. By default plot.Results is set to FALSE.

`FDR.cutoff` The FDR.cutoff argument should be assigned a value between 0.01 and 0.2. Using FDR cutoffs outside these bounds is not recommended. This number determines which false discovery rate cutoff will be used to discover significant genes by ExCluster. However, this parameter is only used if plot.Results is specified. By default FDR.cutoff=0.05

Value

This is the main function of the ExCluster package, and returns a data frame of exon bin log2FCs, log2 read variances, exon clustering per gene, p-values, FDRs, and normalized exon counts. The results of the ExCluster function should typically be assigned to a variable, such as 'ExClustResults'.

Note

The ExCluster packages uses a scatter-distance index function to optimally cut hierarchically clustered exons within each gene. The code for the functions required were adapted in part, or in whole, from the NbClust R package (Charrad et al., 2014). These sections of the code are explicity specified, and the authors of NbClust provide no warranty for the use and fucntionality of said code.

Author(s)

R. Matthew Tanner

References

Charrad M., Ghazzali N., Boiteau V., Niknafs A. (2014). NbClust: An R Package for Determining the Relevant Number of Clusters in a Data Set. Journal of Statistical Software, 61(6), 1-36.

Examples

```
# specify the path to the normCounts file in the ExCluster package
countsPath<- system.file("extdata","normCounts.txt",package="ExCluster")
# now read in the normCounts.txt file
normCounts <- read.table(file=countsPath,header=TRUE,row.names=1,
    stringsAsFactors=FALSE)
# now grab the path to the sub-sampled example GFF file
GFF_file <- system.file("extdata","sub_gen.v23.ExClust.gff3",package="ExCluster")
# assign condition numbers to your samples (we have 4 samples, 2 replicates per condition)
condNums <- c(1,1,2,2)
# now we run ExCluster, assigning its output to the ExClustResults variable
# we are not writing out the ExClustResults table, nor are we plotting exons
# we also use combine.Exons=TRUE, since we are conducting a standard analysis
ExClust_Results <- ExCluster(exon.Counts=normCounts,cond.Nums=condNums,
    GFF.File=GFF_file)
```

GFF_convert *Flatten GTF files into GFF format*

Description

The GFF_convert function takes a GTF file input and flattens in into a GFF file, which contains discrete, non-overlapping exon bin definitions. Transcripts per exon bin are annotated, and normal gene names are added, in addition the Ensembl IDs.

Usage

```
GFF_convert(annot.GTF, GTF.File, GFF.File)
```

Arguments

annot.GTF	This argument must be passed GTF file annotations, either in the form of an rtracklayer object from the rtracklayer import function, or a data table directly imported into R from a GTF file, using tab delimiters. The function will automatically detect the formatting of the file in these cases.
GTF.File	A full file path, including file name & extension, to the GTF file which will be flattened. This parameter is required. An example file path for a GTF file would be: /Users/username/path/to/file.gtf
GFF.File	A full file path to which the flattened GFF file can be written. This argument is not required, although it is strongly recommended that GFF files be saved for future use. An example file path for a GFF file would be: /Users/username/path/to/file.gff

Value

This function returns a GFF data frame, which contains non-overlapping exon bins, flattened from a GTF file. When the function GFF_convert is run, its results should usually be assigned to a variable name, such as 'GFF'.

Note

The GFF_convert function is based on the ideas from the authors of DEXSeq (Anders et al., 2012). Their dexseq_prepare_annotations.py pipeline for flattening GTF files into GFF format was translated into R. Great thanks are owed to Dylan Siriwardena (MSc) for helping write the initial translated R version.

Author(s)

R. Matthew Tanner

References

Anders, S., Reyes, A., and Huber, W. (2012). Detecting differential usage of exons from RNA-seq data. Genome Research, 22(10):2008-17. Lawrence M, Huber W, Pag\'es H, Aboyoun P, Carlson M, et al. (2013) Software for Computing and Annotating Genomic Ranges. PLoS Comput Biol 9(8): e1003118.doi:10.1371/journal.pcbi.1003118

Examples

```
# load the sub-sampled GTF file path from the ExCluster package
GTF_file <- system.file("extdata","sub_gen.v23.gtf", package = "ExCluster")
# now run GTF_file without assigning a GFF_file to write out
# assign this output to the GFF variable
GFF <- GFF_convert(GTF.File=GTF_file)
```

GRangesFromExClustResults *Converts ExCluster results data table to GRanges format.*

Description

GRangesFromExClustResults converts the results from the main ExCluster function into GRanges objects using the GRanges package.

Usage

```
GRangesFromExClustResults(results.ExClust)
```

Arguments

results.ExClust	This argument should be given the table of data that the main ExCluster function produces. This table of data should contain at minimum 12 columns, plus exin bin read count columns if they are present.

Value

This function returns a GRanges object with ExCluster main function results, in the form of a table of data.

Author(s)

R. Matthew Tanner

References

Lawrence M, Huber W, Pag\'es H, Aboyoun P, Carlson M, et al. (2013) Software for Computing and Annotating Genomic Ranges. PLoS Comput Biol 9(8): e1003118.doi:10.1371/journal.pcbi.1003118

Examples

```
# grab the path to the sub-sampled ExCluster toy dataset results from the ExCluster function
ExClust_Results <- system.file("extdata","sub_ExClust_results.txt",package="ExCluster")
# read this data into R
ExClust_Results <- read.table(file=ExClust_Results, sep="\t",header=TRUE,stringsAsFactors=FALSE)
# run the GRangesFromExClustResults function
GRanges.ExClustResults <- GRangesFromExClustResults(results.ExClust=ExClust_Results)
```

plotExonlog2FC *The plotExonlog2FC function plots exon log2FC means and variances for significantly differentially expressed genes in ExCluster results.*

Description

This function takes an input of ExCluster's results dataframe, as well as a directory to write files to, and plots exon bin log2FC means and variances for each significantly differentially expressed gene. Significant genes are determined by applying an FDR cutoff to the ExCluster results. By default, FDR.cutoff=0.05, although this parameter may be set between 0.01 and 0.2.

Usage

```
plotExonlog2FC(results.Data, out.Dir, FDR.cutoff, plot.Type)
```

Arguments

`results.Data`	This argument should be assigned a data frame containing the results from an ExCluster analysis, obtained by running the function ExCluster. This argument is required to be set. Assuming ExCluster results are assigned to the variable clustResults, this parameter woud be set as follows: results.Data=clustResults
`out.Dir`	The outDir argument should be assigned a character string specifying a full folder path where exon bin images per gene may be written. To keep results coherent, this should be a sub-folder within the folder the original ExCluster results were written to. This directory need not exist, as R will recursively create it. However, this argument is required. For example, if ExCluster results were written to the folder /Users/username/Documents/RNA-seq/project_A/ then the folder the images should be something like: /Users/username/Documents/RNA-seq/project_A/exon_log2FC_images/
`FDR.cutoff`	The FDR.cutoff argument should be assigned a value between 0.01 and 0.2. Using FDR cutoffs outside these bounds is not recommended. This number determines which false discovery rate cutoff will be used to discover significant genes by ExCluster. This parameter is not required, and by default FDR.cutoff=0.05
`plot.Type`	The plot.Type function should be assigned values of either "bitmap" or "PNG". Both inputs will plot PNG images. Either Ghostscript or X11 forwarding is required for plotting images on various operating systems. Default = "bitmap"

Value

This function does not return any values, but instead plots exon log2FC means and variances for significant results from an ExCluster function results data.frame.

Author(s)

R. Matthew Tanner

Examples

```
# grab the file path to sub-sampled results from an ExCluster function analysis
ExClust_respath <- system.file("extdata","sub_ExClust_results.txt",
    package="ExCluster")
# load this data into R
ExClust_Results <- read.table(file=ExClust_respath,sep="\t",header=TRUE,
    row.names=1,stringsAsFactors=FALSE)
# now we must specify a directory to write images to
outDir <- paste(tempdir(),"/Images/",sep="")
# now we can run our exon log2FC plotting function
plotExonlog2FC(results.Data=ExClust_Results, out.Dir=outDir)
```

`processCounts` *Counts reads per exon bin for each BAM file, using GFF exon annotations – also normalizes library sizes.*

Description

The processCounts function takes an input of BAM files, and counts reads per exon bin from GFF annotations. GFF annotations may be given either in the form of a data frame assigned to annot.GFF, or the path to a GFF file given to GFF.File. processCounts also requires sample names per BAM file, to unambiguously name the read count columns. Reads may be counted as pairs or individually for paired-end reads, by setting the paired.Reads parameter to TRUE or FALSE.

Usage

```
processCounts(bam.Files,sample.Names,annot.GFF,GFF.File,paired.Reads, stranded.Reads,
                out.File,temp.Dir=NULL,num.Cores=NULL)
```

Arguments

`bam.Files`	An array of full file paths, specifying BAM files from RNA-seq experiments for which exon bin reads will be counted. Each item in the array must specify a full file path to a unique BAM file – duplicate BAM files are not allowed. This is a required input.
`sample.Names`	An array of sample names corresponding, in exact order, to each BAM file provided to the bam.Files argument. The length of sample.Names must also match the length of bam.Files exactly. This is a required input.
`annot.GFF`	If this argument is specified, it must be given a GFF annotation data frame, which is generated from the GFF_convert function. For example, it can be specified as annot.GFF=GFF, assuming your GFF data frame is assigned to the GFF variable. Either annot.GFF or GFF.File must be specified. If both are specified, annot.GFF will take priority.
`GFF.File`	If this argument is specified, it must be given a full file path to a GFF file, including file name and extension. For example, this may look like: /Users/username/path/to/file.gff This argument is not required, but either annot.GFF or GFF.File must be specified.

`paired.Reads` paired.Reads should be assigned a value of either TRUE or FALSE, and functions as a logical operator to determine whether or not reads should be counted as pairs. For example, if you are counting reads from a BAM file with paired-end reads, you should set paired.Reads=TRUE By default paired.Reads is FALSE, and thus need not be set for single-end reads.

`stranded.Reads` This argument should be given a logical TRUE/FALSE value, determining whether RNA-seq reads are stranded or unstranded. This argument is set to FALSE by default. If your RNA-seq reads are stranded, use stranded.Reads=TRUE. If you are unsure, either assume that reads are unstranded, or consult the sequencing facility from which you obtained your RNA-seq fastq files.

`out.File` This argument should be given a variable with a string which contains the full file path to which you wish to write your normalized exon bin counts. For example, you could set outFile<-"/path/to/normCounts.txt", and run processCounts with out.File=outFile

`temp.Dir` This argument should be given a variable with a string which contains the full file path to a temporary directory to which you wish to write featureCounts temporary files. In some cases this can greatly speed up read counting. By default, featureCounts temporary files will be written to R's tempdir() directory.

`num.Cores` This argument should be given a positive integer value which corresponds to the number of cores you with featureCounts to use, when exon reads are counted from your GFF file. For example, using 4 cores would look like: num.Cores=4 This argument defaults to using 1 core within the function.

Value

This function returns normalized exon counts per non-overlapping exon bin in the form of a data frame, in which the user has manually entered sample names which will be present as column headers. This should typically be assigned to a variable when run, such as 'normCounts'.

Note

processCounts runs the featureCounts function from the Rsubread package (Liao et al., 2013). As their function is being called from an external package, the Rsubread authors and license owners make no warranties as to its performance.

Author(s)

R. Matthew Tanner

References

Liao, Y., Smyth, G.K., and Shi, W. (2013). The Subread aligner: fast, accurate and scalable read mapping by seed-and-vote. Nucleic Acids Research, 41(10):e108.

Examples

```
# specify the path to the ExCluster package
ExClust_Path <- system.file(package="ExCluster")
# now find the bam files within that folder
```

```
bamFiles <- list.files(ExClust_Path,recursive=TRUE,pattern="*.bam",
    full.names=TRUE)
# now grab the path to the sub-sampled example GFF file
GFF_file <- system.file("extdata","sub_gen.v23.ExClust.gff3",
    package="ExCluster")
# assign sample names (only 2 replicates per condition in this example)
sampleNames <- c("iPSC_cond1_rep1","iPSC_cond1_rep2","iPSC_cond2_rep1",
    "iPSC_cond2_rep2")
# now run processCounts, with paired.Reads=TRUE for paired-end data
normCounts <- processCounts(bam.Files=bamFiles, sample.Names=sampleNames,
    GFF.File=GFF_file, paired.Reads=TRUE)
```

Index

ExCluster, 2

GFF_convert, 4
GRangesFromExClustResults, 6

plotExonlog2FC, 7
processCounts, 8