Package 'CONDOP'

February 24, 2016

Type Package

Title Condition-Dependent Operon Predictions

Version 1.0

Date 2016-12-02

Author Vittorio Fortino <vittorio.fortino@ttl.fi>

Maintainer Vittorio Fortino <vittorio.fortino@ttl.fi>

Description An implementation of the computational strategy for the comprehensive analysis of condition-dependent operon maps in prokaryotes proposed by Fortino et al. (2014) <DOI:10.1186/1471-2105-15-145>. It uses RNA-seq transcriptome profiles to improve prokaryotic operon map inference.

License GPL-3

LazyData true

RoxygenNote 5.0.1

Depends R (>= 3.0.1)

Imports stats, utils, mclust, earth, plyr, seqinr, randomForest, rminer, GenomicRanges, GenomeInfoDb, S4Vectors, IRanges

NeedsCompilation no

Repository CRAN

Date/Publication 2016-02-24 15:47:51

R topics documented:

CONDOP-package *Develop ensemble classifiers for condition-dependent operon predictions.*

Description

A novel approach for the identification of condition-dependent operons. This R package allows R users to build customized ensemble classifiers that integrate genomic and expression-based features to distinguish operon pairs (OPs) from non-operon pairs (NOPs) in a given RNA-seq based transcriptome profile. The ensemble classifier is then used to re-define the operon information annotated in DOOR and build a condition-dependent operon map that includes putative operons.

Details

Package:	CONDOP
Type:	Package
Version:	1.0
Date:	2016-01-01
License:	GPL-3

CONDOP provides two functions: pre.proc() and run.CONDOP().

pre.proc() pre-process input data for the main function run.CONDOP. run.CONDOP() integrates both genomic and transcriptomic features to develop an operon pair classifier to be used to re.define the operon map for a given bacterial organism and experimental condition.

Input data consists of annotations and count tables.

Annotations: GFF-like, representing gene/feature annotations; DOOR-like, containing operon annotations downloaded from DOOR database; FASTA-like, containing the genome sequence of the target organism.

GFF-like files can be downloaded from the NCBI genomes ftp directory, ftp://ftp.ncbi.nih.gov/genomes. While, DOOR-like files can be downloaded from http://csbl.bmb.uga.edu/DOOR/displayspecies.php. Finally, FASTA-like files can be downloaded from www.ncbi.nlm.nih.gov. Count tables represent RNA-seq expression profiles. A count table must contain two columns: fwd (coverage depth on the forward strand) and rev (coverage depth on the reverse strand). While, each row indicates the number of reads (depth coverage) at each nucleotide position. Different count table should be related to different experimental conditions. The user must use the output of the pre.proc function as first input for the run.CONDOP.

Author(s)

Fortino Vittorio Maintainer: <vittorio.fortino@ttl.fi>

References

Literature or other references for background information

ct1 *Coverage Vector 1*

Description

An example of data frame containing the transcriptome RNA-seq profile of a prokaryotic organism studied under a specific experimental condition. It contains the coverage vectors for both strands: fwd (coverage depth on the forward strand) and rev (coverage depth on reverse strand).

Usage

```
data(ct1)
```

Format

A data frame.

Value

Return a data frame having two columns: fwd (coverage depth on the forward strand) and rev (coverage depth on reverse strand).

pre.proc *Prepare data inputs for the main function* run.CONDOP().

Description

Load the annotation files and a list of count tables (or coverage vectors). Each count table is related to a specific experimental condition and it must contain two columns: fwd (coverage depth on the forward strand) and rev (coverage depth on the reverse strand). The annotations files are:

- GFF-like file, it can be downloaded from the NCBI genomes ftp directory, ftp://ftp.ncbi.nih.gov/genomes.

- DOOR-like file, it can be downloaded from http://csbl.bmb.uga.edu/DOOR/displayspecies.php.

- FASTA-like file, it can be downloaded from www.ncbi.nlm.nih.gov.

Usage

```
pre.proc(gff.file, door.op.file, fasta.file, list.cov.dat,
  remove.cov = list("rRNA"), log2.expr = TRUE, sw = 100,
  save.data.file = NULL, verbose = TRUE)
```

Arguments

gff.file	A full local path indicating the GFF-like file to load <Gene annotations>.
door.op.file	A full local path indicating the DOOR-like file to load (DOOR-operon annotations).
fasta.file	A full local path indicating the FASTA-like file to load or a character string representing the accession number of the genome sequence to download.
list.cov.dat	List of count tables.
remove.cov	List of character values. Each charcater value corresponds to a specific type of annotated features. The coverage depth from those annotated feature will be removed. The default list contains "rRNA". The coverage depth of "rRNA" features will be removed.
log2.expr	Logical value indicating whether CONDOP will be using logged values of expression. The expression values are compiled in RPKM values. Default logical value is TRUE.
sw	Numeric value specifying the sliding window size. Default value is 100.
save.data.file	Character string naming a file. The file will contain the input for the CONDOP main process.
verbose	Indicate whether information about the process should be reported. Defaults to TRUE.

Value

A list of data inputs for the main process `run.CONDOP`.

genes.and.ops	A merged dataframe containing information about genes/features and operons merged.
gseq	A character vector representing the genome sequence of the target organism.
igr.pos	A dataframe containing information about intergenic regions (IRGs) - forward (+) strand.
igr.neg	A dataframe containing information about intergenic regions (IRGs) - reverse (-) strand.
tl.cds	A list of dataframes containing the expression levels of annotated coding sequences (CDS regions). One dataframe for each count table.
tl.igr.pos	A list of dataframes containing the expression levels of intergenic sequences (IGR regions) - forward (+) strand. One dataframe for each count table.
tl.igr.neg	A list of dataframes containing the expression levels of intergenic sequences (IGR regions) - reverse (-) strand. One dataframe for each count table.
sid.points	A list of dataframes containing information about boundaries of transcriptionally active regions.
cut.lhe	A list of numeric vectors indicating the cut-off values to distinguish low expressed RNA-seq data from high expression data on the forward and reverse strands. One dataframe for each count table.

Note

Use the pre.proc function before running run.CONDOP. You do not have to worry about how to make the input data structures for the the run.CONDOP function.

Author(s)

Vittorio Fortino

Examples

```
## Not run:
    file_operon_annot <- system.file("extdata", "1944.opr", package="CONDOP")
    file_genome_seq   <- system.file("extdata", "EC-k12-MG1655.fasta", package="CONDOP")
    data(ct1)
    data.in <- pre.proc(file_genome_annot, file_operon_annot, "NC_000913",
                        list.cov.dat = list(ct1 = ct1))

## End(Not run)
```

---

read.annot.from.gff *Read a GFF file from NCBI and return a GRanges object.*

Description

Read a GFF file from NCBI and return a GRanges object.

Usage

```
read.annot.from.gff(gff.file, locus.tags = TRUE, nrows = -1,
  verbose = TRUE)
```

Arguments

gff.file	a GFF file
locus.tags	only return genes with locus tags. Defaults to TRUE.
nrows	number of rows to read.
verbose	Indicate whether information about the process should be reported. Defaults to TRUE.

Value

GRanges with 4 elementMetadata columns: locus, proetin id, gene id, feature, description and gene name. If all rows are returned (locus.tags=FALSE), then score, phase and tags are included.

Author(s)

Vittorio Fortino read.annot.from.gff()

run.CONDOP *Build condition-dependent operon maps.*

Description

It develops an ensemble operon pair classifier that combines both genomic and transcriptomic features. The ensemble classifier consists of three machine-learning models that are trained on a small set of confirmed operon pairs (OPs) and non-operon pairs (NOPs). The set of OPs and NOPs is identified by crosschecking the DOOR annotation with consecutive, active coding-sequence and intergenic regions, indicated with CDSs and IGR respectively. The trained ensemble classifier is used to predict the operon status of all the gene-pairs, including DOOR-based operon pairs, namely DOPs, and putative operon pairs (POPs). Finally, a linkage process is exploited to combine consecutive operon-pairs classified as OP, and to build the map of condition-dependent operons.

Usage

```
run.CONDOP(data.in, bkgExprCDS = 0.1, bkgExprIGR = 0.25, maxLenIGR = 150,
  win.start.trp = c(100, 10), win.end.trp = c(10, 100), norm.type = "n1",
  cl.run = 30, nfolds = 5, cons = 2, find.ext = FALSE,
  save.TAB.file = NULL, save.BED.file = NULL, return.all = FALSE,
  verbose = TRUE)
```

Arguments

data.in	The output of the pre.proc function.
bkgExprCDS	A threshold to be used for finding active coding-sequence regions. Default values is 0.1.
bkgExprIGR	A threshold to be used for finding the active/transcribed intergenic regions. Default values is 0.25.
maxLenIGR	Maximum length for the intergenic regions. Default values is 150.
win.start.trp	Specify the maximum and the minimum distance from the beginning of a coding region. It is important to validate transcription start points. Defauls values are 100 (max) and 10 (min).
win.end.trp	Specify the minimum and maximum distance from the end of a coding region. It is important to validate transcription end points. Defauls values are 10 (min) and 100 (max).
norm.type	Character vector indicating the method to use for the normalization step. Default value is "n1". n0 - without normalization; n1 - standardization ((x-mean)/sd); n2 - positional standardization ((x-median)/mad); n3 - unitization ((x-mean)/range); n4 - unitization with zero minimum ((x-min)/range); n5 - normalization in range <-1,1> ((x-mean)/max(abs(x-mean))).
cl.run	Number of runs of training/validation. Default values is 30.
nfolds	Indicate the number of folds to be used for the cross-validation step. Default values is 5.

cons	Indicate the minimum number of positive votes necessary to classify a gene pair as operon pair. Default values is 2.
find.ext	To add putative operon pairs classified as OP to the condition-dependent operon map. Defaults to FALSE.
save.TAB.file	Character string naming a file. The final condition operon map is saved in a tab-delimeted text file. Default values is NULL - the cond. operon map is not saved.
save.BED.file	Character string naming a file. The final condition operon map is saved in a BED-like file. Default values is NULL - the cond. operon map is not saved.
return.all	Logical value indicating if extra data must be provided in output.
verbose	Indicate whether information about the process should be reported. Defaults to TRUE.

Value

List of data structures built by CONDOP. If return.all is FALSE:

ndata	A list of dataframes containing OPs and NOPs used for the traing/validation process. One for each count table.
cls	A list of OP classifiers for each count table.
ev.cls	A data.frame containing the evalaution result for the trained classifiers. One for each count table.
pred.TS	A list of dataframes containing the classification results on the training set. One for each count table.
pred.POPs	A list of dataframes containing the prediction results on the POPs. One for each count table.
pred.DOPs	A list of dataframes containg the prediction results on the DOPs. One for each count table.
comap	A list of condition-dependent operon maps (comaps). One for each count table.
info	A list of generic information on the confirmed DOOR based operons. One for each count table.

If return.all is TRUE the run.CONDOP() function also provides..

osp	A list of dataframes containing confirmed operon start points. One for each count table.
oep	A list of dataframes containing confirmed operon end points. One for each count table.
cops	A list of dataframes containing confirmed operons. One for each count table.
OPs	A list of dataframes containing OPs. One for each count table.
NOPs	A list of dataframes containing NOPs. One for each count table.
POPs	A list of dataframes containing POPs. One for each count table.
DOPs	A list of dataframes containing DOPs. One for each count table.

Author(s)

Vittorio Fortino

Examples

```
## Not run:
    file_operon_annot <- system.file("extdata", "1944.opr", package="CONDOP")
    file_genome_seq   <- system.file("extdata", "EC-k12-MG1655.fasta", package="CONDOP")
    data(ct1)
    data.in   <- pre.proc(file_genome_annot, file_operon_annot, "NC_000913",
                          list.cov.dat = list(ct1 = ct1))
    res.comap <- run.CONDOP(data.in = data.in, bkgExprCDS = 0.2, bkgExprIGR = 0.2,
                            maxLenIGR = 150, find.ext = TRUE)

## End(Not run)
```

Index