Package: ACME (via r-universe)

November 25, 2026

Version 2.70.0

Date 2011-06-15

Title Algorithms for Calculating Microarray Enrichment (ACME)

Depends R (>= 2.10), Biobase (>= 2.5.5), methods, BiocGenerics

Imports graphics, stats, utils

Description ACME (Algorithms for Calculating Microarray Enrichment) is a set of tools for analysing tiling array ChIP/chip, DNAse hypersensitivity, or other experiments that result in regions of the genome showing ``enrichment''. It does not rely on a specific array technology (although the array should be a ``tiling'' array), is very general (can be applied in experiments resulting in regions of enrichment), and is very insensitive to array noise or normalization methods. It is also very fast and can be applied on whole-genome tiling array experiments quite easily with enough memory.

License GPL (>= 2)

URL https://github.com/seandavi/ACME

BugReports https://github.com/seandavi/ACME/issues

biocViews Technology, Microarray, Normalization

Repository Bioconductor 3.24

Date/Publication 2026-11-23 22:28:54 UTC

RemoteUrl https://github.com/bioc/ACME

RemoteRef RELEASE_3_24

RemoteSha c00e445a7f15355b08d580031cfa44930a85f0af

Contents

ACMECalcSet-class 2
ACMESet-class 3
aGFF-class 5

aGFFCalc-class 6
do.aGFF.calc 7
example.agff 8
findClosestGene 8
findRegions 9
generics 10
getRefflat 11
read.resultsGFF 12
write.bedGraph 13
write.sgr 14

Index **15**

`ACMECalcSet-class` *Class "ACMECalcSet"*

Description

A subclass of ACMESet that can also store the parameters and results of an ACME calculation

Objects from the Class

Objects can be created by calls of the form `new("ACMECalcSet", assayData, phenoData, featureData, experimentData, annotation, cutpoints, threshold, exprs, vals, ...)`. In addition to the constraints defined by ACMESet, this class can also hold the results (in the assayDataElement `vals`) and the `threshold` and `cutpoints` from an ACME do.aGFF.calc run

Slots

`cutpoints`**:** Object of class `"numeric"` The values of the cutpoints used in an analysis by do.aGFF.calc, one per sample.

`threshold`**:** Object of class `"numeric"` The threshold used in an analysis.

`assayData`**:** Object of class `"AssayData"`. See ExpressionSet for details.

`phenoData`**:** Object of class `"AnnotatedDataFrame"` See ExpressionSet for details.

`featureData`**:** Object of class `"AnnotatedDataFrame"` See ExpressionSet for details.

`experimentData`**:** Object of class `"MIAME"` See ExpressionSet for details.

`annotation`**:** Object of class `"character"` See ExpressionSet for details.

`.__classVersion__`**:** Object of class `"Versions"` See ExpressionSet for details.

Extends

Class `"ACMESet"`, directly. Class `"ExpressionSet"`, by class "ACMESet", distance 2. Class `"eSet"`, by class "ACMESet", distance 3. Class `"VersionedBiobase"`, by class "ACMESet", distance 4. Class `"Versioned"`, by class "ACMESet", distance 5.

Methods

cutpoints `signature(x = "ACMECalcSet")`: A simple getter for the cutpoints.

plot `signature(x = "ACMECalcSet")`: A convenience plotting method that also takes sample and chrom

show `signature(object = "ACMECalcSet")`: A show method

threshold `signature(x = "ACMECalcSet")`: A simple getter for the threshold

vals `signature(x = "ACMECalcSet")`: an accessor for the p-values from a run of do.aGFF.calc. Returns a matrix with samples in columns and probes in rows.

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

See Also

ACMESet

Examples

```
showClass("ACMECalcSet")
data(example.agff)
b <- do.aGFF.calc(example.agff,thresh=0.95,window=1000)
b
head(vals(b))
threshold(b)
cutpoints(b)
```

ACMESet-class *Class "ACMESet"*

Description

An extension of ExpressionSet to deal with ACME data including chromosome locations

Objects from the Class

Objects can be created by calls of the form `new("ACMESet", assayData, phenoData, featureData, experimentData, annotation, exprs, ...)`. The exprs assayDataElement stores the data. The featureData slot stores the chromosome location. In practice, the data.frame underlying the featureData MUST contain three columns named chromosome, start, and end; this is enforced by the class validity method.

Slots

`assayData`: Object of class `"AssayData"`. See ExpressionSet for details.

`phenoData`: Object of class `"AnnotatedDataFrame"` See ExpressionSet for details.

`featureData`: Object of class `"AnnotatedDataFrame"` See ExpressionSet for details.

`experimentData`: Object of class `"MIAME"` See ExpressionSet for details.

`annotation`: Object of class `"character"` See ExpressionSet for details.

`.__classVersion__`: Object of class `"Versions"` See ExpressionSet for details.

Extends

Class `"ExpressionSet"`, directly. Class `"eSet"`, by class "ExpressionSet", distance 2. Class `"VersionedBiobase"`, by class "ExpressionSet", distance 3. Class `"Versioned"`, by class "ExpressionSet", distance 4.

Methods

chromosome `signature(object = "ACMESet")`: Accessor for the chromosome. Returns a vector of chromosomes.

end `signature(x = "ACMESet")`: Accessor for the end location for a probe. If that is not known, this could be set to the same value as the start location.

plot `signature(x = "ACMESet")`: A convenience plotting method that takes a sample name and chrom as well.

start `signature(x = "ACMESet")`: Accessor for the start location for a probe.

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

See Also

`ExpressionSet`, `ACMECalcSet`

Examples

```
showClass("ACMESet")
data(example.agff)
example.agff
head(chromosome(example.agff))
head(start(example.agff))
head(end(example.agff))
```

aGFF-class *Class for storing GFF-like data*

Description

The GFF format is quite versatile while remaining simple. This class simply stores the annotation associated with a set of GFF files from the same regions of the genome along with some information about the samples from which the data came and the data (from the "score" column of the GFF file) themselves.

Objects from the Class

Objects can be created by calls of the form `new("aGFF", ...)`. Also, the `read.resultsGFF()` function returns aGFF objects.

Slots

`annotation`**:** Object of class `"data.frame"` with two columns absolutely necessary, "Chromosome" and "Location". Other columns can be included.

`data`**:** Object of class `"matrix"` of the same number of rows as the annotation slot and the same number of columns as the number of rows in the samples slot, containing data for later analysis

`samples`**:** Object of class `"data.frame"` for describing the samples, one row per sample

Methods

plot `signature(x = "aGFF")`: to plot a region along the genome.

print `signature(x = "aGFF")`: simple method to display summary of aGFF object

show `signature(object = "aGFF")`: simple method to display summary of aGFF object

Author(s)

Sean Davis

See Also

`read.resultsGFF` and`aGFFCalc-class`

Examples

```
# Load an example
data(example.agff)
example.agff
```

aGFFCalc-class *Class "aGFFCalc"*

Description

Store results of ACME calculations

Objects from the Class

Objects can be created by calls of the form `new("aGFFCalc", ...)`.

Slots

`call`: Object of class `"call"`, contains the exact call to do.aGFF.calc, for historical purposes

`threshold`: Object of class `"numeric"`, the threshold used in the calculation

`cutpoints`: Object of class `"numeric"`, the data value above which probes were considered positive

`vals`: Object of class `"matrix"`, equivalent in size to the original data matrix, containing the calculated p-values from the ACME algorithm

`annotation`: Object of class `"data.frame"`, currently a copy of the original annotation, possibly reordered in chromosome order

`data`: Object of class `"matrix"`, the original data, possibly reordered

`samples`: Object of class `"data.frame"`, sample metadata

Extends

Class `"aGFF"`, directly.

Methods

plot `signature(x = "aGFFCalc", ask=FALSE)`: plot the results of an ACME calculation

print `signature(x = "aGFFCalc")`: brief overview of the object

show `signature(object = "aGFFCalc")`: brief overview of the object

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

See Also

`do.aGFF.calc`, `aGFF-class`

Examples

```
data(example.agff)
example.agffcalc <- do.aGFF.calc(example.agff,window=1000,thresh=0.9)
example.agffcalc
```

`do.aGFF.calc` *Perform ACME calculation*

Description

This function performs the moving window chi-square calculation. It is written in C, so is quite fast.

Usage

```
do.aGFF.calc(x, window, thresh)
```

Arguments

`x`	An `aGFF` class object
`window`	An integer value, representing the number of basepairs to include in the windowed chi-square calculation
`thresh`	The quantile of the data distribution for each sample that will be used to classify a probe as positive

Details

A window size on the order of 2-3 times the average size of fragments from sonication, digestion, etc. and containing at least 8-10 probes is the recommended size. Larger size windows are probably more sensitive, but obviously reduce the accuracy with which boundaries of signal can be called.

A threshold of between 0.9 and 0.99 seems empirically to be adequate. If one plots the histogram of data values and there is an obvious better choice (such as a bimodal distribution, with one peak representing enrichment), a more data-driven approach may yield better results.

Value

An object of class `aGFFCalc`

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

Examples

```
data(example.agff)
example.agffcalc <- do.aGFF.calc(example.agff,window=1000,thresh=0.9)
example.agffcalc
```

`example.agff` *An example ACME data structure of class ACMESet*

Description

An ACMESet data structure from two Nimblegen arrays, custom tiled to include multiple HOX genes.

Usage

```
data(example.agff)
```

Format

The format is: chr "example.agff"

Source

From Scacheri et al., Plot Genet, 2006. Pubmed ID 16604156

Examples

```
data(example.agff)
example.agff
```

`findClosestGene` *Find closest refseq gene*

Description

This function is used to find the nearest refseq transcript(s) to a point in the genome specified. Note that it is limited to the refseq transcripts listed at genome.ucsc.edu, where this function goes for information.

Usage

```
findClosestGene(chrom, pos, genome = "hg17", position = "txStart")
```

Arguments

`chrom`	Usually specified like 'chr1', 'chr2', etc.
`pos`	A position in base pairs in the genome
`genome`	Something like 'hg16', 'hg17', 'mm6', etc.
`position`	The location to measure distance from: one of 'txStart', 'txEnd', 'cdsStart', 'cdsEnd'

Details

The first time the function is run, it checks to see if the refflat table for the given genome is present in the package environment. If not, it downloads it to the /tmp directory and gunzips it (using getRefflat. It is then stored so that in future calls, there is no re-download required.

Value

A data frame with the gene name, refseq id(s), txStart, txEnd, cdsStart, cdsEnd, exon count, and distance. Note that distance is measured as pos-position, so negative values mean that the point in the gene is to the left of the point specified in the function call (with the p-tel on the left).

Note

The function may return more than one transcript, as several transcripts may have the same start site

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

Examples

```
findClosestGene('chr1',100000000,'hg17')
```

---

findRegions *Find all regions in data above p-value threshold*

Description

After the ACME calculation, each probe is associated with a p-value of enrichment. However, one often wants the contiguous regions associated with runs of p-values above a given p-value threshold.

Usage

```
findRegions(x, thresh = 1e-04)
```

Arguments

x	An ACMESetCalc object
thresh	The p-value threshold

Details

Runs of p-values above the p-value threshold will be reported as one "region". These can be used for downstream analyses, export to browsers, submitted for transcription factor binding enrichment analyses, etc.

Value

A data frame with these columns:

Length	The length of the region in probes
TF	Either TRUE or FALSE; TRUE regions represent regions of enrichment while FALSE regions are the regions between the TRUE regions
StartInd	The starting Index of the region
EndInd	The ending Index of the region
Sample	The sample containing the region
Chromosome	The Chromosome of the region
Start	The starting basepairof the region
End	The ending basepair of the region
Median	The median p-value in the region
Mean	The mean p-value in the region

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

See Also

do.aGFF.calc, findClosestGene

Examples

```
data(example.agff)
example.agffcalc <- do.aGFF.calc(example.agff,window=1000,thresh=0.9)
foundregions <- findRegions(example.agffcalc,thresh=0.001)
foundregions[1:6,]
```

---

generics *Generics defined within ACME*

Description

See methods descriptions for details.

Usage

```
vals(x, ...)
chromosome(object, ...)
end(x, ...)
start(x, ...)
plot(x, y, ...)
cutpoints(x, ...)
threshold(x, ...)
```

Arguments

x	An ACMESet or ACMECalcSet object (for cutpoints and threshold)
object	An ACMESet or ACMECalcSet object (for cutpoints and threshold)
y	Treated as missing for plotting these types of objects
...	Passed into method

Details

These are all getters for ACMESet and ACMECalcSet objects.

Value

See methods descriptions for details

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

See Also

ACMESet, ACMECalcSet

Examples

```
data(example.agff)
head(chromosome(example.agff))
head(end(example.agff))
head(start(example.agff))
```

---

getRefflat *Get the refflat table from ucsc for the given genome*

Description

Fetches the refflat table from ucsc, stores in temp dir and then gunzips it and reads it in.

Usage

```
getRefflat(genome = "hg17")
```

Arguments

genome	The genome code from ucsc, like 'hg16', 'mm6', etc.

Value

A data frame mirroring the UCSC table structure.

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

References

http://genome.ucsc.edu

See Also

findClosestGene

Examples

```
rf <- getRefflat('hg17')
```

`read.resultsGFF` *Read Nimblegen GFF files*

Description

A GFF format file is a quite flexible format for storing genomic data. Nimblegen uses these format files as one format for making chip-chip data available. This function reads these files, one per experiment and creates a resulting aGFF-class object.

Usage

```
read.resultsGFF(fnames, path = ".", samples = NULL, notes = NULL, skip = 0, sep = "\t", quote = "\"", ...)
```

Arguments

`fnames`	A vector of filenames
`path`	The path to the filenames
`samples`	A data.frame containing sample information, one row per sample, in the same order as the files in fnames
`notes`	A character vector for notes–not currently stored
`skip`	Number of lines to skip if the file contains a header
`sep`	The field separator–should be a tab character for gff files, but can be set if necessary.
`quote`	The text quote character–again not used for gff file, typically
`...`	...

Details

The output is an ACMESet object.

Value

A single ACMESet object.

Author(s)

Sean Davis <sdavis2@mail.nih.gov>

References

http://www.sanger.ac.uk/Software/formats/GFF/

See Also

ACMESet

Examples

```
datdir <- system.file('extdata',package='ACME')
fnames <- dir(datdir)
example.agff <- read.resultsGFF(fnames,path=datdir)
```

---

write.bedGraph *Write bedGraph format tracks for UCSC genome browser*

Description

Generate bedGraph format files for the UCSC genome browser. This function will write the bedGraph files associated with a aGFFcalc object. There will be either one or two files (default two) representing the raw data and the calculated data (which is output as -log10(val) for visualization purposes for EACH sample).

Usage

```
write.bedGraph(x, raw = TRUE, vals = TRUE, directory = ".")
```

Arguments

x	An ACMESet or ACMECalcSet object
raw	Boolean. Create a file for the raw data?
vals	Boolean. Create a file for the calculated p-values?
directory	Give a directory for storing the files

Author(s)

Sean Davis

Examples

```
data(example.agff)
write.bedGraph(example.agff)
```

`write.sgr` *Write Affy IGB .sgr format files*

Description

The affy Integrated Genome Browser (IGB) is a powerful, fast browser for genomic data. The file format is simple (three columns: chromosome, location, and score) to generate. This function will write the sgr files associated with a aGFFcalc object. There will be either one or two files (default two) representing the raw data and the calculated data (which is output as -log10(val) for visualization purposes).

Usage

```
write.sgr(x, raw = TRUE, vals = TRUE, directory = ".")
```

Arguments

`x`	An ACMESet or ACMECalcSet object
`raw`	Boolean. Create a file for the raw data?
`vals`	Boolean. Create a file for the calculated p-values?
`directory`	Give a directory for storing the files

Author(s)

Sean Davis

Examples

```
data(example.agff)
write.sgr(example.agff)
```

Index

∗ **IO**
findClosestGene, 8
getRefflat, 11
read.resultsGFF, 12
write.bedGraph, 13
write.sgr, 14
∗ **classes**
ACMECalcSet-class, 2
ACMESet-class, 3
aGFF-class, 5
aGFFCalc-class, 6
∗ **datasets**
example.agff, 8
∗ **htest**
do.aGFF.calc, 7
∗ **manip**
findRegions, 9
generics, 10
read.resultsGFF, 12

ACMECalcSet, *4*, *11*
ACMECalcSet-class, 2
ACMESet, *2*, *3*, *11*, *13*
ACMESet-class, 3
aGFF-class, 5
aGFFCalc-class, 6

chromosome *(*generics*)*, 10
chromosome,ACMESet-method
*(*ACMESet-class*)*, 3
cutpoints *(*generics*)*, 10
cutpoints,ACMECalcSet-method
*(*ACMECalcSet-class*)*, 2

do.aGFF.calc, *6*, 7, *10*

end *(*generics*)*, 10
end,ACMESet-method *(*ACMESet-class*)*, 3
eSet, *2*, *4*
example.agff, 8

ExpressionSet, *2*, *4*

findClosestGene, 8, *10*, *12*
findRegions, 9

generics, 10
getRefflat, *9*, 11

plot *(*generics*)*, 10
plot,ACMECalcSet-method
*(*ACMECalcSet-class*)*, 2
plot,ACMESet-method *(*ACMESet-class*)*, 3
plot,aGFF-method *(*aGFF-class*)*, 5
plot,aGFFCalc-method *(*aGFFCalc-class*)*, 6
print,aGFF-method *(*aGFF-class*)*, 5
print,aGFFCalc-method *(*aGFFCalc-class*)*,
6

read.resultsGFF, *5*, 12

show,ACMECalcSet-method
*(*ACMECalcSet-class*)*, 2
show,aGFF-method *(*aGFF-class*)*, 5
show,aGFFCalc-method *(*aGFFCalc-class*)*, 6
start *(*generics*)*, 10
start,ACMESet-method *(*ACMESet-class*)*, 3

threshold *(*generics*)*, 10
threshold,ACMECalcSet-method
*(*ACMECalcSet-class*)*, 2

vals *(*generics*)*, 10
vals,ACMECalcSet-method
*(*ACMECalcSet-class*)*, 2
Versioned, *2*, *4*
VersionedBiobase, *2*, *4*

write.bedGraph, 13
write.sgr, 14