Package 'waveTiling'

April 15, 2020

Version 1.28.0

Date 2012-05-14

License GPL (>=2)

Title Wavelet-Based Models for Tiling Array Transcriptome Analysis

Author Kristof De Beuf <kristof.debeuf@UGent.be>, Peter Pipelers <peter.pipelers@ugent.be> and Lieven Clement <lieven.clement@gmail.com>

Maintainer Kristof De Beuf <kristof.debeuf@UGent.be>

Depends oligo, oligoClasses, Biobase, Biostrings, GenomeGraphs

Imports methods, affy, preprocessCore, GenomicRanges, waveslim, IRanges

Suggests BSgenome, BSgenome.Athaliana.TAIR.TAIR9, waveTilingData, pd.atdschip.tiling, TxDb.Athaliana.BioMart.plantsmart22

Description This package is designed to conduct transcriptome analysis for tiling arrays based on fast wavelet-based functional models.

Collate allClasses.R allGenerics.R helperFunctions.R initialize-methods.R methods-MapFilterProbe.R methods-WaveTilingFeatureSet.R methods-WfmFit.R methods-WfmInf.R show-methods.R

URL https://r-forge.r-project.org/projects/wavetiling/

LazyLoad yes

LazyData

biocViews Microarray, DifferentialExpression, TimeCourse, GeneExpression

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

git_branch RELEASE_3_10

git_last_commit 13e6d5c

git_last_commit_date 2019-10-29

Date/Publication 2020-04-14

R topics documented:

`addPheno` *Add phenotypic info to* `WaveTilingFeatureSet`

Description

Function to add phenotypic information such as sample names or group names to a `WaveTilingFeatureSet`-class object

Usage

```
addPheno(object, noGroups, groupNames, replics, ...)
```

Arguments

`object`	object of class `WaveTilingFeatureSet`
`noGroups`	Number of groups in the tiling array experiment
`groupNames`	Vector containing the group or sample names in the tiling array experiment. The vector length should be equal to the indicated number of groups.
`replics`	Numeric vector containing the number of replicates for each group. The vector length should be equal to the indicated number of groups.
`...`	other arguments

Value

object of class WaveTilingFeatureSet annotated with the phenotypic data

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
## Not run:
  data(leafdev)
  leafdev <- as(leafdev,"WaveTilingFeatureSet")
 leafdev <- addPheno(leafdev,noGroups=6,groupNames=c("day8","day9","day10","day11","day12","day13"),replics
  leafdev

## End(Not run)
```

---

bgCorrQn *Background correction and quantile normalization*

Description

Function to perform background correction and quantile normalization of tiling arrays

Usage

```
bgCorrQn(object, useMapFilter=NULL)
```

Arguments

object	object of class WaveTilingFeatureSet
useMapFilter	NULL or object of class mapFilterProbe indicating the probes to use for background correction and quantile normalization

Value

object of class WaveTilingFeatureSet containing the background-corrected and quantile-normalized intensity signals

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
## Not run:
  data(leafdev)
  leafdev <- as(leafdev,"WaveTilingFeatureSet")
  data(leafdevMapAndFilterTAIR9)
  leafdevBQ <- bgCorrQn(leafdev,useMapFilter=leafdevMapAndFilterTAIR9)

## End(Not run)
```

cel2TilingFeatureSet *Read CEL-files to TilingFeatureSet*

Description

Wrapper function to read in CEL-files and output their content as a TilingFeatureSet-class object.

Usage

```
cel2TilingFeatureSet(dataPath, annotationPackage)
```

Arguments

dataPath	character indicating the data path where the CEL-files to read in are stored
annotationPackage	name of the package containing the array probe and annotation information, as produces by the pdInfoBuilder-package

Details

Uses the list.celfiles and read.celfiles functions of the oligo-package.

Value

object of class TilingFeatureSet

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
## No example
```

filterOverlap *function to filter probe sequence overlaps and remap probes*

Description

This function remaps the probe sequence to a reference sequence and filters probe sequence overlaps between PM and MM probes and/or between probes on the forward and reverse strand

Usage

```
filterOverlap(object, remap = TRUE, BSgenomeObject, chrId, strand = c("forward","reverse","both"),
```

Arguments

object	object of class WaveTilingFeatureSet
remap	logical to determine whether the tiling array probe sequences have to be remapped to a more recent reference DNA sequence
BSgenomeObject	object of class BSgenome containing the genome sequence of the species for which the probes need to be filtered and remapped
chrId	vector of numerics identifying the chromosomes for which the probes have to be filtered and/or remapped
strand	character indicating the strands for which the probes have to be filtered and/or remapped (forward, reverse or both)
MM	logical to indicate whether the tiling array contains MM probes or not
...	other arguments

Value

An object of class mapFilterProbe is returned containing the indices of the filtered probes.

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
## Not run:
  data(leafdev)
  as(leafdev,"WaveTilingFeatureSet")
  library(BSgenome.Athaliana.TAIR.TAIR9)
 leafdevMapAndFilterTAIR9 <- filterOverlap(leafdev,remap=TRUE,BSgenomeObject=Athaliana,chrId=1:7,strand="bo

## End(Not run)
```

GenomeInfo-class *Class "genomeInfo"*

Description

class to store the genomic info from a WfmFit-class object

Objects from the Class

Objects can be created by calls of the form new("GenomeInfo",chromosome,strand,minPos,maxPos).

Slots

chromosome: Object of class "vector" ~~

strand: Object of class "character" ~~

minPos: Object of class "numeric" ~~

maxPos: Object of class "numeric" ~~

Methods

initialize `signature(.Object = "GenomeInfo")`: ...

show `signature(.Object = "GenomeInfo")`: ...

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("GenomeInfo")
```

---

getNonAnnotatedRegions

Get non-annotated regions

Description

Extract the significant regions found in the wavelet-based transcriptome analysis that don't show any overlap with the existing annotation.

Usage

```
getNonAnnotatedRegions(fit,inf,biomartObj)
```

Arguments

`fit`	object of class `WfmFit`
`inf`	object of class `WfmInf`
`biomartObj`	object of class TxDb representing an annotation database generated from BioMart.

Value

GRangesList object with the non-annotated regions. The first element gives the regions with no annotation overlap on the strand used in the analysis, the second element gives the regions with no annotation overlap on both strands.

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
library(waveTilingData)
library(TxDb.Athaliana.BioMart.plantsmart22)
data(leafdevFit)
data(leafdevInfCompare)
nonAnnoCompare <- getNonAnnotatedRegions(fit=leafdevFit,inf=leafdevInfCompare,biomartObj=TxDb.Athaliana.Bi
nonAnnoCompare
```

getSigGenes *Get significant genes*

Description

Extract the annotated regions (often genes) that overlap with the significant regions found in the wavelet-based transcriptome analysis.

Usage

```
getSigGenes(fit,inf,biomartObj)
```

Arguments

fit	object of class WfmFit
inf	object of class WfmInf
biomartObj	object of class TxDb representing an annotation database generated from BioMart.

Value

GRangesList object. In the elementMetadata of the GRanges elements percOverGene gives the percentage of basepair overlap of the annotated regions by the detected significant region in the analysis; percOverReg gives the percentage of basepair overlap of the detected singificant region in the analysis with the annotated region; totPercOverGene gives per annotated region the total percentage of basepair overlap by all detected significant regions in the analysis that map to that annotated region.

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
library(waveTilingData)
library(TxDb.Athaliana.BioMart.plantsmart22)
data(leafdevFit)
data(leafdevInfCompare)
sigGenesCompare <- getSigGenes(fit=leafdevFit,inf=leafdevInfCompare,biomartObj=TxDb.Athaliana.BioMart.plan
head(sigGenesCompare[[2]])
```

makeContrasts *Construct contrast matrix*

Description

Helper function to construct a contrast matrix to be used in the inference procedure of the wavelet-based transcriptome analysis when conducting a pairwise comparison analysis.

Usage

```
makeContrasts(contrasts, nlevels)
```

Arguments

contrasts	compare: contrasts for pairwise comparison analysis.
nlevels	Number of groups for pairwise comparison analysis.

Value

numeric matrix

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
makeContrasts(contrasts="compare",nlevels=6)
```

---

makeDesign *Construct design matrix*

Description

Helper function to construct a design matrix to be used in the wavelet-based transcriptome analysis.

Usage

```
makeDesign(design=c("time","circadian","group","factorial"), replics, noGroups, factor.levels=NUL
```

Arguments

design	character indicating the design of the tiling array experiment. Currently, the following designs are implemented: time for a time-course design based on polynomial contrasts; circadian for circadian rhythm analysis; group for unordered one-factor designs; factorial for two-factor designs
replics	Numeric vector containing the number of replicates for each group. The vector length should be equal to the indicated number of groups.
noGroups	Number of groups in the tiling array experiment
factor.levels	Factor levels to use if applying two-factor design

Value

numeric matrix

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
makeDesign(design="time",replics=rep(3,6),noGroups=6)
```

MapFilterProbe-class *Class "MapFilterProbe"*

Description

class to store probe information after remapping and/or filtering of probes.

Usage

```
## Accessors

getChromosome(object)
getFilteredIndices(object)
getPosition(object)
getStrand(object)
```

Arguments

object	An instance of MapFilterProbe-class.

Objects from the Class

Objects can be created by calls of the form new("mapFilterProbe",filteredIndices,chromosome,position,stran

Slots

filteredIndices: Object of class "vector" ~~

chromosome: Object of class "vector" ~~

position: Object of class "vector" ~~

strand: Object of class "vector" ~~

Methods

getChromosome signature(object = "MapFilterProbe"): ...

getFilteredIndices signature(object = "MapFilterProbe"): ...

getPosition signature(object = "MapFilterProbe"): ...

getStrand signature(object = "MapFilterProbe"): ...

initialize signature(.Object = "MapFilterProbe"): ...

selectProbesFromFilterOverlap signature(object = "MapFilterProbe"): ...

show signature(object = "MapFilterProbe"): ...

Accessors

In the following code snippets, x is a MapFilterProbe object.

getChromosome(x): Extract the chromosome identifiers.

getFilteredIndices(x): Extract the filtered probe indices.

getPosition(x): Extract the genomic position of the filtered probes.

getStrand(x): Extract the strand orientation info for the filtered probes.

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("MapFilterProbe")

library(waveTilingData)
data(leafdevMapAndFilterTAIR9)
tt1 <- getChromosome(leafdevMapAndFilterTAIR9)
tt2 <- getFilteredIndices(leafdevMapAndFilterTAIR9)
tt3 <- getPosition(leafdevMapAndFilterTAIR9)
tt4 <- getStrand(leafdevMapAndFilterTAIR9)
```

---

plotWfm *plot model fit and genomic regions*

Description

Plot function to visualize the results of the wavelet-based transcriptome analysis. Both the model fit and the significant genomic regions can be plotted and compared with the annotation.

Usage

```
plotWfm(fit, inf, biomartObj, minPos, maxPos, trackFeature="exon", two.strand=TRUE, plotData=TRUE,
```

Arguments

fit	object of class WfmFit
inf	object of class WfmInf
biomartObj	object of class TxDb representing an annotation database generated from BioMart.
minPos	minimum genomic position to plot
maxPos	maximum genomic position to plot
trackFeature	track feature. See GenomeGraphs-package. Default is exon.
two.strand	logical indicating whether to plot two strands or not
plotData	logical indicating whether to plot the raw data or not
plotMean	logical indicating whether to plot the fitted overall mean function or not
tracks	vector of integers containing the track numbers to plot. Track numbers correspond with the order of the elements in the list output from the getGenmicRegions-function.

Details

The plot utilities of the GenomeGraphs-package constitute the backbone of the plotWfm function.

Value

nothing returned

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

References

[1] Durinck S, Bullard J, Spellman PT, Dudoit S: GenomeGraphs: integrated genomic data visualization with R. BMC Bioinformatics 2009, 10:Article 2.

Examples

```
library(waveTilingData)
library(TxDb.Athaliana.BioMart.plantsmart22)
data(leafdevFit)
data(leafdevInfCompare)
trs <- transcripts(TxDb.Athaliana.BioMart.plantsmart22)
sel <- trs[elementMetadata(trs)$tx_name %in% "AT1G62500.1",]
start <- start(ranges(sel))-2000
end <- end(ranges(sel))+2000
plotWfm(fit=leafdevFit,inf=leafdevInfCompare,biomartObj=TxDb.Athaliana.BioMart.plantsmart22,minPos=start,
```

---

selectProbesFromFilterOverlap

select probes from MapFilterProbe *object*

Description

Extract the probe indices from a MapFilterProbe object that map to a region between two specified genomic positions

Usage

```
selectProbesFromFilterOverlap(object, chromosome, strand=c("forward","reverse"), minPos=min(getPo
```

Arguments

object	object of class MapFilterProbe
chromosome	chromosome
strand	strand
minPos	minimum genomic position
maxPos	maximum genomic position

Value

A list of 2 elements is returned. The first element "selection" gives the probe indices in the filtered MapFilterProbe object; the second element "selectionInit" gives the probe indices in the original WaveTilingFeatureSet object.

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
library(waveTilingData)
data(leafdevMapAndFilterTAIR9)
tt <- selectProbesFromFilterOverlap(leafdevMapAndFilterTAIR9,chromosome=1,strand="forward",minPos=10000,ma
sel <- tt$selection
length(sel)
head(sel)
selfil <- tt$selectionFiltered
length(selfil)
head(selfil)
```

`selectProbesFromTilingFeatureSet`
select probes from `WaveTilingFeatureSet` *object*

Description

Extract the probe indices from a `WaveTilingFeatureSet` object that map to a region between two specified genomic positions

Usage

```
selectProbesFromTilingFeatureSet(object, chromosome, strand=c("forward","reverse"), minPos, maxPo
```

Arguments

`object`	object of class `WaveTilingFeatureSet`
`chromosome`	chromosome
`strand`	strand
`minPos`	minimum genomic position
`maxPos`	maximum genomic position

Value

vector of integers indicating the probe indices

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
## Not run:
  data(leafdevBQ)
 tt <- selectProbesFromTilingFeatureSet(leafdevBQ,chromosome=1,strand="forward",minPos=22000,maxPos=26000)
  length(tt)
  head(tt)

## End(Not run)
```

WaveTilingFeatureSet-class
Class "WaveTilingFeatureSet"

Description

Class to store expression and phenotypic data from a tiling array experiment, used as input for the wavelet-based transcriptome analysis.

Usage

```
## Accessors

getGroupNames(object)
getNoGroups(object)
getReplics(object)
```

Arguments

object	An instance of WaveTilingFeatureSet-class.

Objects from the Class

Objects can be created by calls of the form new("WaveTilingFeatureSet").

Slots

manufacturer: Object of class "character" ~~

intensityFile: Object of class "character" ~~

assayData: Object of class "AssayData" ~~

phenoData: Object of class "AnnotatedDataFrame" ~~

featureData: Object of class "AnnotatedDataFrame" ~~

experimentData: Object of class "MIAME" ~~

annotation: Object of class "character" ~~

protocolData: Object of class "AnnotatedDataFrame" ~~

.__classVersion__: Object of class "Versions" ~~

Extends

Class "TilingFeatureSet", directly. Class "FeatureSet", by class "TilingFeatureSet", distance 2. Class "NChannelSet", by class "TilingFeatureSet", distance 3. Class "eSet", by class "TilingFeatureSet", distance 4. Class "VersionedBiobase", by class "TilingFeatureSet", distance 5. Class "Versioned", by class "TilingFeatureSet", distance 6.

Methods

addPheno signature(object = "WaveTilingFeatureSet"): ...

bgCorrQn signature(object = "WaveTilingFeatureSet"): ...

filterOverlap signature(object = "WaveTilingFeatureSet"): ...

getGroupNames signature(object = "WaveTilingFeatureSet"): ...

getNoGroups signature(object = "WaveTilingFeatureSet"): ...

getReplics signature(object = "WaveTilingFeatureSet"): ...

selectProbesFromTilingFeatureSet signature(object = "WaveTilingFeatureSet"): ...

wfm.fit signature(object = "WaveTilingFeatureSet"): ...

Accessors

In the following code snippets, x is a WaveTilingFeatureSet object. The described accessors are specific for WaveTilingFeatureSet-class objects. Other inherited accessors work as expected on this class.

getGroupNames(x): Extract the group or sample names in the tiling array experiment.

getNoGroups(x): Extract the number of groups or samples in the tiling array experiment.

getReplics(x): Extract the number of replicates in the tiling array experiment.

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WaveTilingFeatureSet")
library(waveTilingData)
data(leafdev)
leafdev <- as(leafdev,"WaveTilingFeatureSet")
leafdev <- addPheno(leafdev,noGroups=6,groupNames=c("day8","day9","day10","day11","day12","day13"),replics
tt1 <- getGroupNames(leafdev)
tt2 <- getNoGroups(leafdev)
tt3 <- getReplics(leafdev)
```

---

wfm.fit *Fit Wfm model to trancriptome data*

Description

Main function to fit a wavelet-based functional model to the tiling array expression data.

Usage

```
wfm.fit(object, filter.overlap=NULL, design=c("time","circadian","group","factorial","custom"), n
```

Arguments

object	object of class WaveTilingFeatureSet
filter.overlap	object of class mapFilterProbe
design	character indicating the design of tiling array experiment. Currently, the following designs are implemented: time for a time-course design based on polynomial contrasts; circadian for circadian rhythm analysis; group for unordered one-factor designs; factorial for two-factor designs; custom for other designs. When using design="custom" a specific design.matrix needs to be given.
n.levels	number of levels in wavelet decomposition (integer)
factor.levels	factor levels in case of two-factor analysis. Vector of integers with length the number of factors in the experiment, and with elements the number of levels for the respective factors.
chromosome	numeric to indicate the chromosome associated with transcriptome data to fit
strand	character to indicate the strand orientation associated with transcriptome data to fit. Either "forward" or "reverse".
minPos	integer to indicate minimum genomic position
maxPos	integer to indicate maximum genomic position
design.matrix	custom design matrix to use
var.eps	character indicating how to estimate residual variance. Either "margLik" for marginal maximum likelihood based estimation or "mad" for estimation based on the MAD (more info see references).
prior	character indicating which prior distribution to put on effect functions. Either "normal" for a normally distributed prior, or "improper" for an improper prior (more info see references).
eqsmooth	logical indicating whether to force equal amount of smooth for different effect functions or not
max.it	integer giving the maximum number of iteration for estimation
wave.filt	character indicating which wavelet filter to use. Default is "haar".
skiplevels	integer indicating how many wavelet levels to put equal to 0
trace	logical indicating whether to trace estimation
save.obs	character to indicate which output to store in return object. Either "plot": all info needed to make the plots or "all": store all possible info.

Value

object of class WfmFit

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

References

[1] Clement L, De Beuf K, Thas O, Vuylsteke M, Irizarry RA and Crainiceanu CM. (2012) Fast wavelet based functional models for transcriptome analysis with tiling arrays. Statistical Applications in Genetics and Molecular Biology 11: Iss. 1, Article 4.

[2] De Beuf K, Andriankaja, M, Thas O, Inze D, Crainiceanu CM and Clement L (2012) Model-based analysis of tiling array expression studies with flexible designs. Technical document.

Examples

```
library(waveTilingData)
data(leafdevBQ)
data(leafdevMapAndFilterTAIR9)
leafdevFit <- wfm.fit(leafdevBQ,filter.overlap=leafdevMapAndFilterTAIR9,design="time",n.levels=10,chromosc
```

`wfm.inference` *Perform transcriptome analysis on fitted wavelet-based functional model*

Description

Main function to perform trancriptome analysis on a fitted wavelet-based functional model of class `WfmFit`.

Usage

```
wfm.inference(object, contrast.matrix=NULL, contrasts=c("compare","means","effects","overallMean"
```

Arguments

`object`	object of class `WfmFit`
`contrast.matrix`	custom contrasts matrix
`contrasts`	character indicating the type of transcriptome analysis is to be applied. Currently the following types are implemented: `compare` for doing a pairwise differential expression analysis between any combination of two groups; `effects`, which corresponds to a circadian rhythm analysis if a circadian design is used for the fit, and to a time effects analysis (linear, quadratic,...) if a time-course design is used for the fit; `means` for doing a group-wise transcript discovery analysis.
`delta`	threshold value to be used in the inference procedure. This should be a numeric vector with as first element the threshold for the overall mean transcript discovery and the other elements the threshold for the differential expression, the effects analysis or group-wise mean analysis. If the threshold should be equal for all comparisons, effects or group-wise means only a vector of length 2 is needed. Otherwise, the vector must be of length `r+1` with `r` the number of pairwise comparisons, effects or group-wise means.
`two.sided`	logical indicating if one-sided or two-sided tests are desired
`minRunPos`	minrun by position. An integer to indicate the minimum number of basepairs a significant genomic region should contain.
`minRunProbe`	minrun by probes. An integer to indicate the minimum number of probes the significant genomic region should map to.
`alpha`	significance level
`nsim`	number of simulations used when doing circadian rhythm inference
`rescale`	rescale matrix

Value

object of class `WfmFit`

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

References

[1] Clement L, De Beuf K, Thas O, Vuylsteke M, Irizarry RA and Crainiceanu CM. (2012) Fast wavelet based functional models for transcriptome analysis with tiling arrays. Statistical Applications in Genetics and Molecular Biology 11: Iss. 1, Article 4.

[2] De Beuf K, Andriankaja, M, Thas O, Inze D, Crainiceanu CM and Clement L (2012) Model-based analysis of tiling array expression studies with flexible designs. Technical document.

Examples

```
library(waveTilingData)
data(leafdevFit)
delta <-  log(1.2,2)
leafdevInfCompare <- wfm.inference(leafdevFit,contrasts="compare",delta=c("median",delta))
```

`WfmFit-class` *Class "WfmFit"*

Description

class to store model fits in the wavelet-based transcriptome analysis.

Usage

```
## Accessors

getProbePosition(object)
getNoProbes(object)
getBetaWav(object)
getVarBetaWav(object)
getSmoothPar(object)
getVarEps(object)
getGenomeInfo(object)
getMinPos(object)
getMaxPos(object)
getNoLevels(object)
getDesignMatrix(object)
getPhenoInfo(object)
getDataOrigSpace(object)
getDataWaveletSpace(object)
getWaveletFilter(object)
getKj(object)
getPrior(object)
getF(object)
getVarF(object)
```

Arguments

object	An instance of WfmFit-class.

Objects from the Class

Objects can be created by calls of the form new("WfmFit",betaWav,varbetaWav,smoothPar,varEps,dataOrigSpace

Slots

betaWav: Object of class "matrix" ~~

varbetaWav: Object of class "matrix" ~~

smoothPar: Object of class "matrix" ~~

varEps: Object of class "numeric" ~~

dataOrigSpace: Object of class "matrix" ~~

dataWaveletSpace: Object of class "matrix" ~~

design.matrix: Object of class "matrix" ~~

phenoData: Object of class "data.frame" ~~

genome.info: Object of class "genomeInfo" ~~

n.levels: Object of class "numeric" ~~

probePosition: Object of class "numeric" ~~

wave.filt: Object of class "character" ~~

Kj: Object of class "numeric" ~~

prior: Object of class "character" ~~

F: Object of class "matrix" ~~

varF: Object of class "matrix" ~~

P: Object of class "numeric" ~~

Z: Object of class "matrix" ~~

noGroups: Object of class "numeric" ~~

replics: Object of class "numeric" ~~

Methods

getBetaWav signature(object = "WfmFit"): ...

getChromosome signature(object = "WfmFit"): ...

getDataOrigSpace signature(object = "WfmFit"): ...

getDataWaveletSpace signature(object = "WfmFit"): ...

getDesignMatrix signature(object = "WfmFit"): ...

getF signature(object = "WfmFit"): ...

getGenomeInfo signature(object = "WfmFit"): ...

getKj signature(object = "WfmFit"): ...

getMaxPos signature(object = "WfmFit"): ...

getMinPos signature(object = "WfmFit"): ...

getNoLevels signature(object = "WfmFit"): ...

getNoProbes signature(object = "WfmFit"): ...
getPhenoInfo signature(object = "WfmFit"): ...
getPrior signature(object = "WfmFit"): ...
getProbePosition signature(object = "WfmFit"): ...
getSmoothPar signature(object = "WfmFit"): ...
getStrand signature(object = "WfmFit"): ...
getVarBetaWav signature(object = "WfmFit"): ...
getVarEps signature(object = "WfmFit"): ...
getVarF signature(object = "WfmFit"): ...
getWaveletFilter signature(object = "WfmFit"): ...
initialize signature(.Object = "WfmFit"): ...
show signature(object = "WfmFit"): ...
wfm.inference signature(object = "WfmFit"): ...

Accessors

In the following code snippets, x is a WfmFit object.

getBetaWav(x): Extract the fitted effect functions in the wavelet space.

getChromsome(x): Extract the chromosome identifiers.

getDataOrigSpace(x): Extract the raw expression data in the original data space.

getDataWaveletSpace(x): Extract the raw data in the wavelet space,i.e. the wavelet coefficients.

getDesignMatrix(x): Extract the design matrix used in the wavelet-based analysis.

getF(x): Extract the fitted functional effects in the original data space.

getGenomeInfo(x): Extract the genomic information.

getKj(x): Extract the number of wavelet coefficients estimated per wavelet level.

getMaxPos(x): Extract the maximum genomic probe position.

getMinPos(x): Extract the minimum genomic probe position.

getNoLevels(x): Extract the number of levels in in the wavelet decomposition when fitting the wavelet-based functional model.

getNoProbes(x): Extract the number of probes.

getPhenoInfo(x): Extract the phenotypic information for the tiling array experiment.

getPrior(x): Extract the the type or distribution of the prior imposed on the functional effects in the wavelet space.

getProbePosition(x): Extract probe position.

getSmoothPar(x): Extract the estimated smoothing parameters that control the regularization of the effect functions in the wavelet space.

getStrand(x): Extract the strand orientation info.

getVarBetaWav(x): Extract the variance of the fitted effect functions in the wavelet space.

getVarEps(x): Extract the estimated residual variance in the wavelet space. One variance parameter is estimated per wavelet level.

getVarF(x): Extract the variance of the fitted functional effects in the original data space.

getWaveletFilter(x): Extract the wavelet filter used to transform the data from the original space to the wavelet space.

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmFit")

library(waveTilingData)
data(leafdevFit)
tt1 <- getBetaWav(leafdevFit)
tt2 <- getChromosome(leafdevFit)
tt3 <- getDataOrigSpace(leafdevFit)
tt4 <- getDataWaveletSpace(leafdevFit)
tt5 <- getDesignMatrix(leafdevFit)
tt6 <- getF(leafdevFit)
tt7 <- getGenomeInfo(leafdevFit)
tt8 <- getKj(leafdevFit)
tt9 <- getMaxPos(leafdevFit)
tt10 <- getMinPos(leafdevFit)
tt11 <- getNoLevels(leafdevFit)
tt12 <- getNoProbes(leafdevFit)
tt13 <- getPhenoInfo(leafdevFit)
tt14 <- getPrior(leafdevFit)
tt15 <- getProbePosition(leafdevFit)
tt16 <- getSmoothPar(leafdevFit)
tt17 <- getStrand(leafdevFit)
tt18 <- getVarBetaWav(leafdevFit)
tt19 <- getVarEps(leafdevFit)
tt20 <- getVarF(leafdevFit)
tt21 <- getWaveletFilter(leafdevFit)
```

WfmFitCircadian-class *Class "WfmFitCircadian"*

Description

class to store model fits with a circadian rhythm design in the wavelet-based transcriptome analysis.

Objects from the Class

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

Slots

betaWav: Object of class "matrix" ~~

varbetaWav: Object of class "matrix" ~~

smoothPar: Object of class "matrix" ~~

varEps: Object of class "numeric" ~~

dataOrigSpace: Object of class "matrix" ~~

dataWaveletSpace: Object of class "matrix" ~~

design.matrix: Object of class "matrix" ~~

`phenoData`: Object of class `"data.frame"` ~~

`genome.info`: Object of class `"genomeInfo"` ~~

`n.levels`: Object of class `"numeric"` ~~

`probePosition`: Object of class `"numeric"` ~~

`wave.filt`: Object of class `"character"` ~~

`Kj`: Object of class `"numeric"` ~~

`prior`: Object of class `"character"` ~~

`F`: Object of class `"matrix"` ~~

`varF`: Object of class `"matrix"` ~~

`P`: Object of class `"numeric"` ~~

`Z`: Object of class `"matrix"` ~~

`noGroups`: Object of class `"numeric"` ~~

`replics`: Object of class `"numeric"` ~~

Extends

Class `"WfmFit"`, directly.

Methods

initialize `signature(.Object = "WfmFitCircadian")`: ...

show `signature(object = "WfmFitCircadian")`: ...

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmFitCircadian")
```

WfmFitCustom-class *Class "WfmFitCustom"*

Description

class to store model fits with custom design in the wavelet-based transcriptome analysis.

Objects from the Class

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

Slots

`betaWav`: Object of class "`matrix`" ~~

`varbetaWav`: Object of class "`matrix`" ~~

`smoothPar`: Object of class "`matrix`" ~~

`varEps`: Object of class "`numeric`" ~~

`dataOrigSpace`: Object of class "`matrix`" ~~

`dataWaveletSpace`: Object of class "`matrix`" ~~

`design.matrix`: Object of class "`matrix`" ~~

`phenoData`: Object of class "`data.frame`" ~~

`genome.info`: Object of class "`genomeInfo`" ~~

`n.levels`: Object of class "`numeric`" ~~

`probePosition`: Object of class "`numeric`" ~~

`wave.filt`: Object of class "`character`" ~~

`Kj`: Object of class "`numeric`" ~~

`prior`: Object of class "`character`" ~~

`F`: Object of class "`matrix`" ~~

`varF`: Object of class "`matrix`" ~~

`P`: Object of class "`numeric`" ~~

`Z`: Object of class "`matrix`" ~~

`noGroups`: Object of class "`numeric`" ~~

`replics`: Object of class "`numeric`" ~~

Extends

Class "`WfmFit`", directly.

Methods

initialize `signature(.Object = "WfmFitCustom")`: ...

show `signature(object = "WfmFitCustom")`: ...

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmFitCustom")
```

`WfmFitFactor-class` *Class "WfmFitFactor"*

Description

class to store model fits with factorial design in the wavelet-based transcriptome analysis.

Objects from the Class

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

Slots

`betaWav`: Object of class `"matrix"` ~~

`varbetaWav`: Object of class `"matrix"` ~~

`smoothPar`: Object of class `"matrix"` ~~

`varEps`: Object of class `"numeric"` ~~

`dataOrigSpace`: Object of class `"matrix"` ~~

`dataWaveletSpace`: Object of class `"matrix"` ~~

`design.matrix`: Object of class `"matrix"` ~~

`phenoData`: Object of class `"data.frame"` ~~

`genome.info`: Object of class `"genomeInfo"` ~~

`n.levels`: Object of class `"numeric"` ~~

`probePosition`: Object of class `"numeric"` ~~

`wave.filt`: Object of class `"character"` ~~

`Kj`: Object of class `"numeric"` ~~

`prior`: Object of class `"character"` ~~

`F`: Object of class `"matrix"` ~~

`varF`: Object of class `"matrix"` ~~

`P`: Object of class `"numeric"` ~~

`Z`: Object of class `"matrix"` ~~

`noGroups`: Object of class `"numeric"` ~~

`replics`: Object of class `"numeric"` ~~

Extends

Class `"WfmFit"`, directly.

Methods

initialize `signature(.Object = "WfmFitFactor")`: ...

show `signature(object = "WfmFitFactor")`: ...

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmFitFactor")
```

`WfmFitTime-class` *Class "WfmFitTime"*

Description

class to store model fits with a time-course design in the wavelet-based transcriptome analysis.

Objects from the Class

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

Slots

`betaWav`: Object of class `"matrix"` ~~

`varbetaWav`: Object of class `"matrix"` ~~

`smoothPar`: Object of class `"matrix"` ~~

`varEps`: Object of class `"numeric"` ~~

`dataOrigSpace`: Object of class `"matrix"` ~~

`dataWaveletSpace`: Object of class `"matrix"` ~~

`design.matrix`: Object of class `"matrix"` ~~

`phenoData`: Object of class `"data.frame"` ~~

`genome.info`: Object of class `"genomeInfo"` ~~

`n.levels`: Object of class `"numeric"` ~~

`probePosition`: Object of class `"numeric"` ~~

`wave.filt`: Object of class `"character"` ~~

`Kj`: Object of class `"numeric"` ~~

`prior`: Object of class `"character"` ~~

`F`: Object of class `"matrix"` ~~

`varF`: Object of class `"matrix"` ~~

`P`: Object of class `"numeric"` ~~

`Z`: Object of class `"matrix"` ~~

`noGroups`: Object of class `"numeric"` ~~

`replics`: Object of class `"numeric"` ~~

Extends

Class `"WfmFit"`, directly.

Methods

initialize `signature(.Object = "WfmFitTime")`: ...

show `signature(object = "WfmFitTime")`: ...

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmFitTime")
```

WfmInf-class *Class "WfmInf"*

Description

class to store outputs from the inference in the wavelet-based transcriptome analysis.

Usage

```
## Accessors

getAlpha(object)
getDelta(object)
getTwoSided(object)
getSigProbes(object)
getRegions(object)
getGenomicRegions(object)
getFDR(object)
getEff(object)
getVarEff(object)
```

Arguments

object	An instance of WfmInf-class.

Objects from the Class

Objects can be created by calls of the form new("WfmInf",alpha,delta,two.sided,sigProbes,regions,GlocRegio

Slots

alpha: Object of class "numeric" ~~

delta: Object of class "numeric" ~~

two.sided: Object of class "numeric" ~~

sigProbes: Object of class "list" ~~

regions: Object of class "list" ~~

GlocRegions: Object of class "list" ~~

FDR: Object of class "matrix" ~~

CI: Object of class "array" ~~

eff: Object of class "matrix" ~~

varEff: Object of class "matrix" ~~

genome.info: Object of class "genomeInfo" ~~

Methods

getAlpha signature(object = "WfmInf"): ...

getDelta signature(object = "WfmInf"): ...

getTwoSided signature(object = "WfmInf"): ...

getSigProbes signature(object = "WfmInf"): ...

getRegions signature(object = "WfmInf"): ...

getGenomicRegions signature(object = "WfmInf"): ...

getFDR signature(object = "WfmInf"): ...

getEff signature(object = "WfmInf"): ...

getVarEff signature(object = "WfmInf"): ...

getGenomeInfo signature(object = "WfmFit"): ...

initialize signature(.Object = "WfmInf"): ...

show signature(object = "WfmInf"): ...

plotWfm signature(fit = "WfmFit",inf = "WfmInf"): ...

getSigGenes signature(fit = "WfmFit",inf = "WfmInf"): ...

getNonAnnotatedRegions signature(fit = "WfmFit",inf = "WfmInf"): ...

Accessors

In the following code snippets, x is a WfmInf object.

getAlpha(x): Extract the alpha level of significance used in the wavelet-based analysis.

getDelta(x): Extract the threshold values used in the wavelet-based transcriptome analysis.

getTwoSided(x): Extract the direction of inference conducted in the wavelet-based transcriptome analysis.

getSigProbes(x): Extract the significant probe ids for the wavelet-based transcriptome analysis.

getRegions(x): Extract the significant regions from the wavelet-based transcriptome analysis. Regions are given in terms of the probe ids they map onto.

getGenomicRegions(x): Extract the significant genomic regions from the wavelet-based transcriptome analysis.

getFDR(x): Extract the FDR for each test in the wavelet-based transcriptome analysis.

getEff(x): Extract the estimated effects or contrasts of the wavelet-based transcriptome analysis.

getVarEff(x): Extract the estimated variances of the effects or contrasts in the wavelet-based transcriptome analysis.

getGenomeInfo(x): Extract the genomic information.

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmInf")

library(waveTilingData)
data(leafdevInfCompare)
tt1 <- getAlpha(leafdevInfCompare)
tt2 <- getDelta(leafdevInfCompare)
tt3 <- getTwoSided(leafdevInfCompare)
tt4 <- getSigProbes(leafdevInfCompare)
tt5 <- getRegions(leafdevInfCompare)
tt6 <- getGenomicRegions(leafdevInfCompare)
tt7 <- getFDR(leafdevInfCompare)
tt8 <- getEff(leafdevInfCompare)
tt9 <- getVarEff(leafdevInfCompare)
tt10 <- getGenomeInfo(leafdevInfCompare)
```

WfmInfCompare-class *Class "WfmInfCompare"*

Description

class to store outputs from the inference for a pairwise comparison wavelet-based transcriptome analysis.

Objects from the Class

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

Slots

alpha: Object of class `"numeric"` ~~

delta: Object of class `"numeric"` ~~

two.sided: Object of class `"numeric"` ~~

sigProbes: Object of class `"list"` ~~

regions: Object of class `"list"` ~~

GlocRegions: Object of class `"list"` ~~

FDR: Object of class `"matrix"` ~~

CI: Object of class `"array"` ~~

eff: Object of class `"matrix"` ~~

varEff: Object of class `"matrix"` ~~

Extends

Class `"WfmInf"`, directly.

Methods

initialize `signature(.Object = "WfmInfCompare")`: ...

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmInfCompare")
```

`WfmInfCustom-class` *Class "WfmInfCustom"*

Description

class to store outputs from the inference for a custom design fit in the wavelet-based transcriptome analysis.

Objects from the Class

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

Slots

`alpha`: Object of class `"numeric"` ~~

`delta`: Object of class `"numeric"` ~~

`two.sided`: Object of class `"numeric"` ~~

`sigProbes`: Object of class `"list"` ~~

`regions`: Object of class `"list"` ~~

`GlocRegions`: Object of class `"list"` ~~

`FDR`: Object of class `"matrix"` ~~

`CI`: Object of class `"array"` ~~

`eff`: Object of class `"matrix"` ~~

`varEff`: Object of class `"matrix"` ~~

Extends

Class `"WfmInf"`, directly.

Methods

initialize `signature(.Object = "WfmInfCustom")`: ...

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmInfCustom")
```

`WfmInfEffects-class` *Class "WfmInfEffects"*

Description

class to store outputs from the inference on the effects in the wavelet-based transcriptome analysis.

Objects from the Class

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

Slots

`alpha`: Object of class `"numeric"` ~~

`delta`: Object of class `"numeric"` ~~

`two.sided`: Object of class `"numeric"` ~~

`sigProbes`: Object of class `"list"` ~~

`regions`: Object of class `"list"` ~~

`GlocRegions`: Object of class `"list"` ~~

`FDR`: Object of class `"matrix"` ~~

`CI`: Object of class `"array"` ~~

`eff`: Object of class `"matrix"` ~~

`varEff`: Object of class `"matrix"` ~~

Extends

Class `"WfmInf"`, directly.

Methods

initialize `signature(.Object = "WfmInfEffects")`: ...

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmInfEffects")
```

`WfmInfMeans-class` *Class "WfmInfMeans"*

Description

class to store outputs from the inference on the group-wise means in the wavelet-based transcriptome analysis.

Objects from the Class

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

Slots

`alpha`: Object of class `"numeric"` ~~

`delta`: Object of class `"numeric"` ~~

`two.sided`: Object of class `"numeric"` ~~

`sigProbes`: Object of class `"list"` ~~

`regions`: Object of class `"list"` ~~

`GlocRegions`: Object of class `"list"` ~~

`FDR`: Object of class `"matrix"` ~~

`CI`: Object of class `"array"` ~~

`eff`: Object of class `"matrix"` ~~

`varEff`: Object of class `"matrix"` ~~

Extends

Class `"WfmInf"`, directly.

Methods

initialize `signature(.Object = "WfmInfMeans")`: ...

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmInfMeans")
```

`WfmInfOverallMean-class` *Class "WfmInfOverallMean"*

Description

class to store outputs from the inference on the overall mean expression in the wavelet-based transcriptome analysis.

Objects from the Class

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

Slots

`alpha`: Object of class `"numeric"` ~~

`delta`: Object of class `"numeric"` ~~

`two.sided`: Object of class `"numeric"` ~~

`sigProbes`: Object of class `"list"` ~~

`regions`: Object of class `"list"` ~~

`GlocRegions`: Object of class `"list"` ~~

`FDR`: Object of class `"matrix"` ~~

`CI`: Object of class `"array"` ~~

`eff`: Object of class `"matrix"` ~~

`varEff`: Object of class `"matrix"` ~~

Extends

Class `"WfmInf"`, directly.

Methods

initialize `signature(.Object = "WfmInfOverallMean")`: ...

Author(s)

Kristof De Beuf <kristof.debeuf@ugent.be>

Examples

```
showClass("WfmInfOverallMean")
```

Index