Package ‘waveTiling’

March 26, 2015

Version 1.9.2
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.atschip.tiling, TxDb.Athaliana.BioMart.plantsmart22
Description This package is designed to conduct transcriptome analysis for tiling arrays based on fast wavelet-based functional models.
URL https://r-forge.r-project.org/projects/wavetiling/
LazyLoad yes
LazyData
biocViews Microarray, DifferentialExpression, TimeCourse, GeneExpression
NeedsCompilation yes
addPheno adds phenotypic information such as sample names or group names to a WaveTilingFeatureSet-class object.

**Usage**

```r
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

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

---

**bgCorrQn**

*Background correction and quantile normalization*

Description

Function to perform background correction and quantile normalization of tiling arrays

Usage

```r
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
cel2TilingFeatureSet

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:
```r
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**

```r
## 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**

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

**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**

```r
## Not run:
data(leafdev)
leafdev <- as(leafdev, "WaveTilingFeatureSet")
library(BSgenome.Athaliana.TAIR9)
leaf.devMapAndFilterTAIR9 <- filterOverlap(leafdev, remap=TRUE, BSgenomeObject=Athaliana, chrId=1:7, strand="both")
```

```r
## End(Not run)
```
**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)
getSigGenes

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.BioMart.plantsmart22)
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; perOverReg gives the percentage of basepair overlap of the detected significant 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

```r
library(wavetilingdata)
library(TxDb.Athaliana.BioMart.plantsmart22)
data(leafdevFit)
data(leafdevInfCompare)
sigGenesCompare <- getSigGenes(fit=leafdevFit, inf=leafdevInfCompare, biomartObj=TxDb.Athaliana.BioMart.plantsmart22)
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

```r
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

```r
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"), replicas, noGroups, factor.levels=NULL)

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, strand)`.

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"): ...
getAddress 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.
getAddress(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>
plotWfm

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, plotMean=TRUE, tracks)

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
selectProbesFromFilterOverlap

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(getPosition(object)), maxPos=max(getPosition(object))), chromosome)

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.

Examples

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

Author(s)

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

Examples

library(waveTilingData)
data(leafdevMapAndFilterTAIR9)
tt <- selectProbesFromFilterOverlap(leafdevMapAndFilterTAIR9, chromosome=1, strand="forward", minPos=10000, maxPos=20000)
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, maxPos)

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>
WaveTilingFeatureSet-class

Examples

```r
## 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"
WaveTilingFeatureSet-class

Extends


Methods

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

groupNames(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=repnts)
tt1 <- getGroupNames(leafdev)
tt2 <- getNoGroups(leafdev)
tt3 <- getReplics(leafdev)
**wfm.fit**  
*Fit Wfm model to transcriptome data*

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

**Usage**

```r
wfm.fit(object, filter.overlap=NULL, design=c("time","circadian","group","factorial","custom"), n.levels, factor.levels, chromosome, strand, minPos, maxPos, design.matrix, var.eps, prior, eqsmooth, max.it, wave filt, skiplevels, trace, save.obs)
```

**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


Examples

```r
library(wavetilingdata)
data(leafdevbq)
data(leafdevMapAndFilterTAIR9)
leafdevFit <- wfmNfit(leafdevbq, filter.overlap=leafdevMapAndFilterTAIR9, design="time", n.levels=10, chromosome=
```

**wfm.inference**

Perform transcriptome analysis on fitted wavelet-based functional model

**Description**

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

**Usage**

```r
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


Examples

library(waveTilingData)
data(leafdevFit)
delta <- log(c(1.2,2))
leafdevInfCompare <- wfm.inference(leafdevFit, contrasts="compare", delta=c("median",delta))
class to store model fits in the wavelet-based transcriptome analysis.

Usage

```r
## 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, dataWaveletSpace, designMatrix, phenodata, genomeInfo, nLevels, probeposition, wavefilt, kj, prior, f, varf, p, z, nogroups, replics)`.

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.
getChromosome(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.
gGet GenomeInfo(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 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.
gGet SmoothPar(x): Extract the estimated smoothing parameters that control the regularization of the effect functions.
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>
WfmFitCircadian-class

Examples

```r
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)
```

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"
WfmFitCustom-class

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")

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")
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" ~~
wavefilt: 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")
```

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" ~
```
### Description

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

### Usage

```r
## Accessors

getAlpha(object)
gammaDelta(object)
gammaTwoSided(object)
gammaSigProbes(object)
gammaRegions(object)
gammaGenomicRegions(object)
gammaFDR(object)
gammaEff(object)
gammaVarEff(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, GenomicRegions)`. 

---

<table>
<thead>
<tr>
<th>Description</th>
<th>Class &quot;WfmInf&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>WfmInf-class</td>
<td></td>
</tr>
</tbody>
</table>
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
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")

---

**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")

\begin{verbatim}
WfmInfMeans-class  Class "WfmInfMeans"
\end{verbatim}

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")
```

---

**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

*Topic **classes**
- GenomeInfo-class, 6
- MapFilterProbe-class, 9
- WaveTilingFeatureSet-class, 14
- WfmFit-class, 19
- WfmFitCircadian-class, 22
- WfmFitCustom-class, 23
- WfmFitFactor-class, 25
- WfmFitTime-class, 26
- WfmInf-class, 27
- WfmInfCompare-class, 29
- WfmInfCustom-class, 30
- WfmInfEffects-class, 31
- WfmInfMeans-class, 32
- WfmInfOverallMean-class, 33

*Topic **hplot**
- plotWfm, 11

*Topic **manip**
- addPheno, 2
- bgCorrQn, 3
- cel2TilingFeatureSet, 4
- filterOverlap, 5
- getNonAnnotatedRegions, 6
- getSigGenes, 7
- makeContrasts, 8
- makeDesign, 9
- selectProbesFromFilterOverlap, 12
- selectProbesFromTilingFeatureSet, 13
- wfm.fit, 16
- wfm.inference, 17

addPheno, 2
addPheno, WaveTilingFeatureSet-method (WaveTilingFeatureSet-class), 14

bgCorrQn, 3
bgCorrQn, WaveTilingFeatureSet-method (WaveTilingFeatureSet-class), 14

14

cel2TilingFeatureSet, 4
eSet, 15
FeatureSet, 15
filterOverlap, 5
filterOverlap, WaveTilingFeatureSet-method (WaveTilingFeatureSet-class), 14

GenomeInfo (GenomeInfo-class), 6
GenomeInfo-class, 6
getAlpha (WfmInf-class), 27
getAlpha, WfmInf-method (WfmInf-class), 27
getBetaWav (WfmFit-class), 19
getBetaWav, WfmFit-method (WfmFit-class), 19
getChromosome (MapFilterProbe-class), 9
getChromosome, MapFilterProbe-method (MapFilterProbe-class), 9
getChromosome, WfmFit-method (WfmFit-class), 19
getDelta (WfmInf-class), 27
getDelta, WfmInf-method (WfmInf-class), 27
getDesignMatrix (WfmFit-class), 19
getDesignMatrix, WfmFit-method (WfmFit-class), 19
getEff (WfmInf-class), 27
gEff, WfmInf-method (WfmInf-class), 27
gF (WfmFit-class), 19
getF, WfmFit-method (WfmFit-class), 19
getFDR, WfmInf-method (WfmInf-class), 27
getFDR, WfmInf-method (WfmInf-class), 27
getFilteredIndices
    (MapFilterProbe-class), 9
getFilteredIndices, MapFilterProbe-method (MapFilterProbe-class), 9
getGenomeInfo, WfmFit-method (WfmFit-class), 19
getGenomeInfo, WfmInf-method (WfmInf-class), 27
getGenomeInfo, WfmInf-method (WfmInf-class), 19
getGenomicRegions, WfmInf-method (WfmInf-class), 27
getGenomicRegions, WfmInf-method (WfmInf-class), 27
getGroupNames
    (WaveTilingFeatureSet-class), 14
getGroupNames, WaveTilingFeatureSet-method (WaveTilingFeatureSet-class), 14
getKj, WfmFit-method (WfmFit-class), 19
getKj, WfmFit-method (WfmFit-class), 19
getMaxPos, WfmFit-method (WfmFit-class), 19
getMaxPos, WfmFit-method (WfmFit-class), 19
getMinPos, WfmFit-method (WfmFit-class), 19
getMinPos, WfmFit-method (WfmFit-class), 19
getNoGroups
    (WaveTilingFeatureSet-class), 14
getNoGroups, WaveTilingFeatureSet-method (WaveTilingFeatureSet-class), 14
getNoLevels, WfmFit-method (WfmFit-class), 19
getNoLevels, WfmFit-method (WfmFit-class), 19
getNonAnnotatedRegions, 6
getNonAnnotatedRegions, WfmFit, WfmInf-method (WfmInf-class), 27
getNoProbes, WfmFit-method (WfmFit-class), 19
getNoProbes, WfmFit-method (WfmFit-class), 19
getPhenoInfo, WfmFit-method (WfmFit-class), 19
getPhenoInfo, WfmFit-method (WfmFit-class), 19
getPosition, MapFilterProbe-method
    (MapFilterProbe-class), 9
getPosition, MapFilterProbe-method
    (MapFilterProbe-class), 9
getPrior, WfmFit-method (WfmFit-class), 19
getPrior, WfmFit-method (WfmFit-class), 19
getProbePosition, WfmFit-method (WfmFit-class), 19
getProbePosition, WfmFit-method (WfmFit-class), 19
getRegions, WfmInf-method (WfmInf-class), 27
getRegions, WfmInf-method (WfmInf-class), 27
getReplics
    (WaveTilingFeatureSet-class), 14
getReplics, WaveTilingFeatureSet-method
    (WaveTilingFeatureSet-class), 14
getSigGenes, 7
getSigGenes, WfmFit, WfmInf-method (WfmInf-class), 27
getSigProbes, WfmInf-method (WfmInf-class), 27
getSigProbes, WfmInf-method (WfmInf-class), 27
getSmoothPar, WfmFit-method (WfmFit-class), 19
getSmoothPar, WfmFit-method (WfmFit-class), 19
getStrand, MapFilterProbe-class, 9
getStrand, MapFilterProbe-method (MapFilterProbe-class), 9
getStrand, WfmFit-method (WfmFit-class), 19
getStrand, WfmFit-method (WfmFit-class), 19
getTwoSided, WfmInf-method (WfmInf-class), 27
getTwoSided, WfmInf-method (WfmInf-class), 27
getVarBetaWav, WfmFit-method (WfmFit-class), 19
getVarBetaWav, WfmFit-method (WfmFit-class), 19
getVarEff, WfmInf-method (WfmInf-class), 27
getVarEff, WfmInf-method (WfmInf-class), 27
getVarEps, WfmFit-class, 19
getVarEps, WfmFit-method (WfmFit-class), 19
getVarF, WfmFit-class, 19
gVarF, WfmFit-method (WfmFit-class), 19
gVarF, WfmFit-method (WfmFit-class), 19
getWaveletFilter, WfmFit-method (WfmFit-class), 19
getWaveletFilter, WfmFit-method (WfmFit-class), 19
WfmFitCustom-class, 23
WfmFitFactor (WfmFitFactor-class), 25
WfmFitFactor-class, 25
WfmFitTime (WfmFitTime-class), 26
WfmFitTime-class, 26
WfmInf, 30–33
WfmInf (WfmInf-class), 27
WfmInf-class, 27
WfmInfCompare (WfmInfCompare-class), 29
WfmInfCompare-class, 29
WfmInfCustom (WfmInfCustom-class), 30
WfmInfCustom-class, 30
WfmInfEffects (WfmInfEffects-class), 31
WfmInfEffects-class, 31
WfmInfMeans (WfmInfMeans-class), 32
WfmInfMeans-class, 32
WfmInfOverallMean
   (WfmInfOverallMean-class), 33
WfmInfOverallMean-class, 33