Package ‘MinimumDistance’

September 24, 2012

Type Package

Title A package for de novo CNV detection in case-parent trios

Version 1.0.0

Date 11/13/2010

Author Robert B Scharpf and Ingo Ruczinski

Maintainer Moiz Bootwalla <`mbootwa1@jhu.edu`>, Robert B Scharpf <`rscharpf@jhsph.edu`>

Description
Analysis of de novo copy number variants in trios from high-dimensional genotyping platforms

License Artistic-2.0

Depends R (>= 2.14), IRanges (>= 1.13.30)

Imports methods, DNAcopy, utils, msm, lattice, BiocGenerics,VanillaICE (>= 1.17.22), ff, SNPchip (>= 2.0.3), Biobase,foreach, oligoClasses (>= 1.17.38)

Suggests human610quadv1bCrlmm, RUnit

Enhances snow, doSNOW

Collate AllClasses.R AllGenerics.R methods-AnnotatedDataFrame.R
methods-AssayData.R methods-Pedigree.R methods-TrioSet.R
methods-TrioSetList.R methods-matrix.R methods-list.R
methods-ff_array.R segment2-methods.R mad-methods.R
lattice-methods.R functions.R utils.R zzz.R

LazyLoad yes

biocViews Microarray, SNP, Bioinformatics, CopyNumberVariants

R topics documented:

`annotatedDataFrameFrom-methods` *Methods for creating AnnotatedDataFrame objects*

Description

Methods for creating AnnotatedDataFrame objects in the package **MinimumDistance**.

Methods

- `signature(object = "array", byrow = "ANY")` Creates an AnnotatedDataFrame from an `array`. `byrow` is ignored.
- `signature(object = "ff_array", byrow = "ANY")` Creates an AnnotatedDataFrame from an `ff_array`. `byrow` is ignored.
- `signature(object = "ff_matrix", byrow = "ANY")` Creates an AnnotatedDataFrame from an `ff_matrix`. `byrow` is ignored.
- `signature(object = "Pedigree", byrow = "logical")` Creates an AnnotatedDataFrame from an object of class `Pedigree`.

Usage

When object is of class `Pedigree`, an `AnnotatedDataFrame` containing phenotypic information on the father, mother, or offspring can be specified by the argument `which`.

```
annotatedDataFrameFrom(object, byrow, sample.sheet,which=c("offspring", "father", "mother")
```

A `data.frame` containing phenotypic information on the samples can be passed to this method through the argument `sample.sheet`. The `sample.sheet` can contain phenotypic information on all the samples. Only the rows relevant to the offspring, for example, will be included when instantating an `AnnotatedDataFrame` when `which` is "offspring". When `sample.sheet` is `missing`, an `AnnotatedDataFrame` with zero rows will be instantiated. When `sample.sheet` is not missing, `row.names` must be specified. The `row.names` are the identifiers for each row in `sample.sheet` and are matched to sample identifiers stored in `object`.

`calculateMindist` *Compute the minimum distance.*

Description

Compute the minimum distance.

Usage

```
calculateMindist(object, ...)
```

Arguments

`object`	A `list` of arrays, a `TrioSetList` object, or an `array` of the log R ratios.
`...`	Ignored.

Details

The 'minimum distance' is the minimum signed absolute difference of the parental log R ratios and the offspring log R ratios. Specifically, let |O-F| denote the absolute difference in the log R ratios comparing offspring to father and |O-M| the absolute difference in the log R ratios comparing offspring to mother. The minimum distance at a marker is the signed minimum of |O-M| and |O-F|. After segmentation of the minimum distance, non-zero segments can indicate a de novo difference in the log R ratio of the offspring and either parent. For example, a positive minimum distance suggests that the log R ratio from the offspring is greater than the log R ratio of either parent.

Value

If `object` is an array, a matrix of the minimum distance is returned. For an object with M markers and T trios, the dimension of the resulting matrix is M x T. If `object` is a list of arrays or a `TrioSetList` object, a list of matrices are returned. The dimension of the ith matrix in the list is M_i x T.

Author(s)

R. Scharpf

Examples

```
data(trioSetListExample)
mdlist <- calculateMindist(lrr(trioSetList))

trioSet <- stack(trioSetList)
md <- calculateMindist(lrr(trioSet))
```

`callDenovoSegments` *Posterior calling for segmented data.*

Description

This function provides a convenient wrapper for the segmentation and posterior calling steps.

Usage

```
callDenovoSegments(path = "", pedigreeData, ext = "", featureData, cdfname, chromosome = 1:22, s
```

Arguments

`path`	`character` string indicating path to BeadStudio files.
`pedigreeData`	Object of class `Pedigree`.
`ext`	`character`: filename extension
`featureData`	An object of class `AnnotatedDataFrame`. Variable labels 'chromosome', 'position', and 'isSnp' are required.
`cdfname`	Name of the package for annotating the chromosome and physical position. Ignored if `featureData` is specified.
`chromosome`	`integer` indicating which autosomal chromosomes to process
`segmentParents`	`Logical`: whether to segment the parental log R ratios using circular binary segmentation
`verbose`	`Logical`: whether to display verbose output indicating progress.
`...`	Additional arguments can be specified for the `segment` function in the package `DNAcopy`

Details

A wrapper for the segmentation and posterior calling steps. Circular binary segmentation is performed on the minimum distance and the offspring log R ratios. The minimum distance (the signed minimum of the absolute difference of the offspring and parental log2 R ratios) should have mean zero in regions of inherited CNV and normal diploid genomes. Hence, the segmentation algorithm should smooth over inherited CNV in which the offspring has the same copy number as the parent as well as normal diploid regions. Therefore, the posterior call for a segment that contains both inherited CNV and normal diploid regions will depend on the relative size of these regions. For example, if most of the segment is diploid the posterior call would be '333' (see State Symbols below for details regarding the state symbols). Conversely, if most of the region contains a deletion transmitted from the mother, the state call will be '322'. Of primary interest are regions for which the offspring copy number differs from the parental copy numbers. Note that for such regions, the CNV in the offspring may be Mendelian or non-Mendelian.

The likelihood for the called state and the likelihood for the normal state are named 'lik.state' and 'lik.norm', respectively. The ratio can provide a useful rank whereby large values indicate strong evidence for the called state.

State Symbols:

The MinimumDistance states can be easily remembered as the latent copy number offset by 1 for the father, mother, and offspring, respectively. For example, a region of the genome for which all individuals in the trio are diploid would have state '333'. By contrast, a region for which the father is diploid and the mother and offspring are hemizygous would have the state symbol '322'.

Value

A object of class `RangedDataCBS` with the predicted trio copy number sates. Only states for which the offspring copy number differs from the parental copy numbers are reliable. In particular, the normal state (state '333') may contain both normal and inherited CNV as the minimum distance for both states is near zero.

Author(s)

R. Scharpf

See Also

`state` for accessing the state symbols (posterior calls) for the genomic ranges.

Examples

```
library(oligoClasses)
foreach::registerDoSEQ()
path <- system.file("extdata", package="MinimumDistance")
fnames <- list.files(path, pattern=".txt")
ped <- Pedigree(fatherIds=fnames[1], motherIds=fnames[2],
offspringIds=fnames[3])
map.segs <- callDenovoSegments(path=path,
       ext="",
       pedigreeData=ped,
       cdfname="human610quadv1b",
       chromosome=1,
       segmentParents=FALSE)
```

`computeBayesFactor` *Compute Bayes factors for minimum distance segments*

Description

Estimate the trio copy number state for each minimum distance segment using the maximum a posterior probability. The posterior probability of the normal state is also returned.

Usage

```
computeBayesFactor(object, ranges, ...)
```

Arguments

`object`	A object of class `TrioSetList` or `TrioSet`.
`ranges`	A `RangedDataCBS` object.
`...`	Additional arguments specified in the methods for `TrioSet` and `TrioSetList`.

Value

A object of class `RangedDataCBS`.

References

The posterior calling is an extension of the joint HMM described in Wang et al, 2008 Nucleic Acids Research.

See Also

`TrioSetList`, `TrioSet` `RangedDataCBS`, `calculateMindist`

Examples

```
library(MinimumDistance)
library(oligoClasses)
foreach::registerDoSEQ()
data(trioSetListExample)
data(md.segs)
data(lrr.segs)
mdlist <- calculateMindist(lrr(trioSetList))
mads.md <- mad2(mdlist, byrow=FALSE)
md.segs2 <- narrow(md.segs, lrr.segs, mad.minimumdistance=mads.md, thr=0.8)
map.segs <- computeBayesFactor(trioSetList, ranges=md.segs2)
```

`concordance` *Functions for assessing concordance*

Description

Functions for assessing concordance and discordance of copy number variant calls

Usage

```
concAtTop(ranges.query, ranges.subject, list.size, verbose = TRUE, ...)
discAtTop(ranges.query, ranges.subject, verbose=TRUE, ...)
```

Arguments

`ranges.query`	RangedDataCNV object
`ranges.subject`	RangedDataCNV object
`list.size`	size of list in query and subject ranges
`verbose`	logical
`...`	Additional arguments passed to `findOverlaps`.

Details

concAtTop calculates three measures of concordance:

1. the proportion of top ranges that overlap between `ranges.query` and `ranges.subject` objects as a function of list size (we assume that each RangedDataCNV object is ordered such that the first range has the highest rank (most evidence of an alteration).

2. the proportion of top ranges in `ranges.query` that appear anywhere in the `ranges.subject` object. Again, a proportion is calculated as a function of list size until the size of the list is equal to `list.size`.

3. the same as (2), but reversing the role of `ranges.query` and `ranges.subject`.

The function `discAtTop` identifies the ranges in `ranges.query` that do not appear in `ranges.subject`.

Value

`distAtTop` returns an object of the same class as `ranges.query`.

`concAtTop` returns a list of 3 elements, corresponding to the 3 approaches for estimating concordance described in the details.

Author(s)

Rob Scharpf

See Also

`findOverlaps`

`correspondingCall` *Find overlapping ranges*

Description

Finds ranges in a `RangedDataCNV` object that overlap with ranges from another `RangedDataCNV` object.

Usage

```
correspondingCall(ranges.query, ranges.subject, subject.method)
```

Arguments

`ranges.query`	A `RangedDataCNV` object.
`ranges.subject`	A `RangedDataCNV` object.
`subject.method`	A character string. If provided, a column called 'method' will be added to `ranges.subject` that indicates the statistical algorithm used to call the de novo alterations.

Details

Given a set of de novo calls from one statistical algorithm, this function finds the corresponding calls made by a second statistical algorithm. For any given range in `ranges.query`, one or more ranges in `ranges.subject` may overlap.

Value

An object of the same class as `ranges.subject`.

Author(s)

Rob Scharpf

See Also

`findOverlaps`

`isDenovo`	*Indicator for whether a trio copy number state is a de novo copy number alteration.*

Description

Return an indicator for a de novo copy number alteration.

Usage

```
isDenovo(states)
```

Arguments

`states`	a string with 3 characters. Character elements 1-3 indicate the copy number state for the father, mother, and offspring, respectively. See Details.

Details

We consider a copy number alteration occurring in the offspring that is not present in either parent to be de novo, irrespective of whether the copy number alteration is transmitted by a Mendelian mechanism. Each segment is assigned a trio copy number state by a 3 digit integer code that indicates the copy number state for the father, mother, and offspring. For example, '332' indicates copy number 2 in the mother and father (state '3') and a hemizygous deletion in the offspring (state '2'). See `computeBayesFactor` for details. A subset of the possible states, such as '332', is considered de novo.

Value

Logical

Author(s)

Rob Scharpf

See Also

`computeBayesFactor`. See `state` for the accessor to extract the trio copy number state.

Examples

```
library(oligoClasses)
isDenovo(c(333, 332, 311, 334, 344))
data(map.segs)
table(isDenovo(state(map.segs)))
sts <- unique(state(map.segs))
sts[isDenovo(sts)]
```

`mad2` *Methods for computing the minimum absolute distance.*

Description

Compute the median absolute deviation for elements in a list, array, or matrix. For lists, elements of the list can be matrices or arrays or ff-versions of these.

Usage

```
mad2(object, byrow=FALSE, ...)
```

Arguments

`object`	`object` can be any of the following: class `TrioSetList`, `TrioSet`, or a list of matrices.
`byrow`	When `byrow` is `TRUE`, the MAD is calculated for each marker across all offspring. When `byrow` is `FALSE`, the MAD is computed across all autosomal markers for the fathers, mothers, and offspring. The former provides a robust estimate of the marker-level variance across samples, whereas the latter provides an estimate of variance for the samples.
`...`	Not currently implemented.

Value

A list.

Examples

```
data(trioSetListExample)
foreach::registerDoSEQ()
## computing the MAD of the log R ratios for each individual (across markers)
mads.sample <- mad2(trioSetList, byrow=FALSE)
## compute the MAD of the log R ratio for each marker (across individuals)
## too few samples
##mads.marker <- mad2(trioSetList, byrow=TRUE)
```

`narrow` *Adjust breakpoints from segmentation inward*

Description

Narrow the minimum distance ranges by the segmentation of the offspring copy number estimates.

Usage

```
narrow(object, lrr.segs, thr, mad.minimumdistance, verbose = TRUE)
```

Arguments

`object`	A `RangedDataCNV` object. The segmentation of the minimum distance.
`lrr.segs`	A `RangedDataCNV` object. The segmentation of the log R ratios.
`thr`	Numeric. For segments with a segment mean less than `thr`, the breakpoints are not altered.
`mad.minimumdistance`	A named numeric vector. The names should the `sampleNames` of the `TrioSet` or `TrioSetList` object.
`verbose`	Logical. Whether to display messages that indicate progress.

Details

If the start and stop coordinates for a segment [x, y] with mean log R ratio greater than `thr` in absolute value, the x and y coordinates of the interval may be adjusted. If there are no breakpoints from the segmentation of the offspring log R ratio occurring in [x, y], nothing is done. However, if one or more breakpoints occur in the interval [x,y], one or more new segments can be created. For example, suppose a segment from the log R ratio segmention has breakpoints given by [a, b], where x < b < y. Then the following two intervals are created:

1. [x, b] 2. [b, y]

The motivation is to avoid having a single minimum distance segment spanning differing copy number states in the offspring.

Value

A `RangedDataCBS` object.

Author(s)

Rob Scharpf

Examples

```
data(trioSetListExample)
data(lrr.segs)
data(md.segs)
        md <- calculateMindist(lrr(trioSetList))
md.mads <- mad2(md, byrow=FALSE)
md.segs.narrowed <- narrow(object=md.segs, lrr.segs=lrr.segs, thr=0.1, mad.minimumdistance=md.mads)
```

`Pedigree` *Constructor for Pedigree class*

Description

Constructor for Pedigree class

Usage

```
Pedigree(pedigreeInfo, fatherIds=character(), motherIds=character(), offspringIds=character())
```

Arguments

`pedigreeInfo`	a `data.frame` with column labels F, M, and O containing sample identifiers for the father (F), mother (M), and offspring (O). Each row of the `data.frame` corresponds to a single trio.
`fatherIds`	A vector of sample ids for the father. Ignored if `pedigreeInfo` is specified.
`motherIds`	A vector of sample ids for the mother. Ignored if `pedigreeInfo` is specified.
`offspringIds`	A vector of sample ids for the mother. Ignored if `pedigreeInfo` is specified.

Details

Constructor for object that contains information regarding the case-parent trios

Value

Object of class `Pedigree`

Note

If `pedigreeInfo` is missing, the vector of character strings provided to indicate the ids of the father, mother and offspring should be ordered such that the ith element in each vector are the sample identifiers for one trio.

Author(s)

R. Scharpf

See Also

`Pedigree`

Examples

```
Pedigree()
Pedigree(fatherIds=letters[1], motherIds=letters[2],offspringIds=letters[3])

path <- system.file("extdata", package="MinimumDistance")
load(file.path(path, "pedigreeInfo.rda"))
Pedigree(pedigreeInfo)
```

`Pedigree-class` *Container for storing pedigree information*

Description

Container for storing familial information for father, mother, offspring trios.

Objects from the Class

See the examples for instantiating a `Pedigree` object.

Slots

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

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

Accessors

In the following code descriptions, `object` is an instance of the `Pedigree` class:

`allNames(object)`: character vector of unique sample ids. Note that the length of this vector is not necessarily a multiple of 3 if some parents have multiple offspring.

`annotatedDataFrameFrom(object, byrow=FALSE, sample.sheet,    which=c("offspring", "father", "mothe` Create an AnnotatedDataFrame for the father, mother or, offspring (depending on value of argument `which`). Optionally, a `data.frame` of covariates for the samples can be passed as an argument to `sample.sheet`. If `sample.sheet` is not missing, `row.names` can not be `NULL`.

`dim(object)`: Returns an integer vector of length 2: the first element is the number of trios; the second element is 3.

`fatherNames(object)`: character vector of father ids (not necessarily unique).

`motherNames(object)`: character vector of mother ids (not necessarily unique).

`offspringNames(object)`: character vector of offspring ids (must be unique).

`trios(object)`: `data.frame` of sample ids. Each row in the `data.frame` contains the ids of father, mother, and offspring, respectively.

`object[i, ]`: subset the Pedigree by trio index `i`.

Author(s)

R. Scharpf

See Also

`Pedigree` `annotatedDataFrameFrom`

Examples

```
showClass("Pedigree")
## an empty container
Pedigree()
```

`phenoData` *Extract phenotype data for parents in a trio.*

Description

Extract phenotype data for parents in a trio.

Usage

```
motherPhenoData(object)
fatherPhenoData(object)
```

Arguments

`object`	A object of class `TrioSet` or `TrioSetList`

Details

Extracts phenotypic data for parents

Value

`AnnotatedDataFrame`

Author(s)

R. Scharpf

See Also

`TrioSet`, `TrioSetList`

Examples

```
data(trioSetListExample)
## father phenoData
fatherPhenoData(trioSetList)
## mother phenoData
motherPhenoData(trioSetList)
## offspring phenoData
phenoData(trioSetList)
```

`RangedDataCBS_Examples` *Objects containing segmented data*

Description

Example `RangedDataCBS` objects created from the segmentation of the log R ratios (object `cbs.segs`) and the minimum distance (object `md.segs`).

Usage

```
data(lrr.segs)
data(md.segs)
        data(map.segs)
```

See Also

See `RangedDataCBS` for accessors and methods available for objects of this class. See `computeBayesFactor` for details regarding how the object `map.segs` was instantiated.

Examples

```
data(lrr.segs)
data(md.segs)


###object containing maximum a posterior probabilities
data(map.segs)
```

`segment2` *Wrapper for segment function in package DNAcopy*

Description

`segment2` is a wrapper for the `segment` function in `DNAcopy`. The first argument can be simple data structures: a `list`, a `matrix`, or an `array`, or more complex: `TrioSetList` and `TrioSet`. If the first argument is a list, each element of the list can be a matrix or an array.

Segmentation of `ff_matrix` and `ff_array` objects is also supported.

Details

When `object` is a list of arrays, the argument `id` is required. The easiest way to obtain a `data.frame` of the trio sample names is via the method `trios`, as in the example below.

Value

`RangedDataCBS` object

Arguments

Arguments to `segment2` depend on the class of `object`. In all forms, additional arguments to the `segment` function in the **DNAcopy** package can be passed through the `...` operator.

For `TrioSetList` objects, the arguments are:

`segment2(object, md=NULL, segmentParents=TRUE, verbose=TRUE, ...)`: `md` is a list of matrices containing the minimum distance. See `calculateMindist`. `segmentParents` must be logical. When TRUE, the parental log R ratios are segmented via circular binary segmentation. When FALSE, only the offspring log R ratios are segmented.

For objects of class `TrioSet`:

`segment2(object, md=NULL, segmentParents=TRUE, verbose=TRUE, ...)`: `md` is a matrix of the minimum distance where each column corresponds to a trio in the `TrioSetList` object.

For objects of class `list`:

`segment2(object, pos, chrom, id=NULL, featureNames, segmentParents=TRUE, verbose=TRUE, ...)`: `pos` is a list of the genomic positions (integers) for each row of the elements of the `object` list. Similarly, `chrom` and `featureNames` are lists specifying the chromosome (`integer`) and feature identifiers (`character`) for each row in the `object` list elements, respectively. Note: `pos`, `chrom`, and `featureNames` must be lists of the same size.

For objects of class `matrix` :

`segment2(object, pos, chrom, id=NULL, featureNames,segmentParents=TRUE, verbose=TRUE, ...)`: `pos` is a vector of the genomic positions (integers) for each row in the `object` matrix. Similarly, `chrom` and `featureNames` are vectors specifying the chromosome (`integer`) and feature identifiers (`character`) for each row in the `object` matrix, respectively. Note: `pos`, `chrom`, and `featureNames` must be vectors of the same size.

See Also

`segment`

Examples

```
## Not run:  ## examples are checked in the vignette
data(trioSetListExample)
mdlist <- calculateMindist(lrr(trioSetList))
md.segs <- segment2(trioSetList, md=mdlist)
lrr.segs <- segment2(trioSetList, segmentParents=FALSE)

## End(Not run)
```

`stackRangedDataList` *Stack a list of RangedDataCBS objects.*

Description

Stack a list of RangedDataCBS objects.

Usage

```
stackRangedDataList(...)
```

Arguments

`...`	A list. Each element has class `RangedDataCBS`.

Value

Object of class `RangedDataCBS`.

Author(s)

R. Scharpf

See Also

`RangedDataCBS`

Examples

```
data(map.segs)
stackRangedDataList(list(map.segs, map.segs))
```

`TrioSet` *Constructor for TrioSet objects.*

Description

Construct an object of class `TrioSet`.

Usage

```
TrioSet(pedigreeData = Pedigree(), sample.sheet, row.names = NULL, lrr,
  baf, featureData, cdfname, drop = TRUE, mindist=NULL)
```

Arguments

`pedigreeData`	A `Pedigree` object.
`sample.sheet`	A `data.frame` of sample covariates.
`row.names`	Row identifiers for `sample.sheet` that match the names of the trios in the `Pedigree` object.
`lrr`	Matrix of log R ratios
`baf`	Matrix of B allele frequencies
`featureData`	`AnnotatedDataFrame` for features.
`cdfname`	`character` string for annotation package.
`drop`	`Logical`. If FALSE, the row and column `dimnames` are null.
`mindist`	A `matrix` of the minimum distance.

Value

A object of class `TrioSet`.

Author(s)

R. Scharpf

See Also

`TrioSet`, `Pedigree`, `Pedigree`

Examples

```
path <- system.file("extdata", package="MinimumDistance")
load(file.path(path, "logRratio.rda"))
load(file.path(path, "baf.rda"))
load(file.path(path, "pedigreeInfo.rda"))
trioSet <- TrioSet(lrr=logRratio,
   baf=baf,
   pedigree=Pedigree(pedigreeInfo),
   cdfname="human610quadv1bCrlmm")
```

`TrioSet-class` *Class* `"TrioSet"`

Description

A TrioSet is a container for storing high throughput assay data and metadata from genotyping arrays when the study design is case-parent trios. In our application, de novo copy number alterations in affected offspring were of the primary interest. Examination of the joint distribution of the log R ratios and B allele frequencies across members in a trio motivates a container with assay data elements that are 3-dimensional arrays rather than 2-dimensional matrices. The dimension of the arrays is marker x trio x individual. Typically, a TrioSet instance stores the data for a single chromosome and an instance of TrioSetList is a list of TrioSets. While having a single TrioSet for the entire dataset would simplify the classes of objects defined in this package, multiple arrays with thousands of trios and roughly a million markers are impractical on most machines. In addition, storage by chromosome will facilitate parallelization of computation that can be carried out independently on different chromosomes.

Objects from the Class

Objects can be created by calls of the form :

`new("TrioSet", logRRatio, BAF, phenoArray, mindist, mad, ...)`.

Subsetting

As the `TrioSet` class is an extension of `eSet`, the subsetting is similar. One important difference is that the assay date elements are 3 dimensional arrays. While `k` is not a formal argument in the generic for `"["`, `k` can be passed to the `"["` method for `TrioSet` objects for subsetting the 3rd dimension of the assay data.

`object[i, j, drop]`: `i` selects features, `j` selects trios, and `k` (though not part of the generic) selects for the individual in a trio. Valid values for `k` are 1 (selects father), 2 (selects mother), or 3 (selects offspring).

Accessors

The `object` in the accessor descriptions that follow is a `TrioSet`:

`allNames(object)`: The individual ids for each subject. See `offspringNames`, `fatherNames`, and `motherNames` to list ids corresponding to membership in the trio.

`baf(object)`: Extract array of B allele frequencies.

`baf(object) <- value`: assign B allele frequencies. `value` is a 3-dimensional array (feature x trio x sample).

`dim(object)`: Returns the dimension of the assay data elements. Each assay data element is a three dimensional array with dimensions for features, trios, and sample, respectively.

`fatherNames(object) <- value`: Assign character string of father sample names.

`fatherNames(object)`: Return character string of father identifiers.

`fatherPhenoData(object)`: Extract `AnnotatedDataFrame` for father pheno data.

`lrr(object) <- value`: assign log R ratios to assayData. `value` is a 3-dimensional array (feature x trio x sample).

`mindist(object)`: Accessor for the minimum distance matrix.

`mindist(object) <- value`: Replacement method for slot `mindist`. Value must be a `matrix`.

`motherNames(object) <- value`: Assign a character string for the mother identifiers

`motherPhenoData(object)`: Extract `AnnotatedDataFrame` for mother pheno data.

`motherNames(object)` Return a character vector of mother identifiers

`ncol(object)` Number of trios in the `TrioSet` object

`offspringNames(object) <- value`: Assign a character vector of offspring identifiers

`offspringNames(object)`: Retrieve character vector of offspring identifiers. Note that the result will identical to `sampleNames(object)` as the offspring identifiers uniquely identify a trio

`offspringPhenoData(object)`: Extract a `AnnotatedDataFrame` of the sample-level covariates for the offspring. Alternatively, use `phenoData(object)`.

`order(object)`: order `TrioSet` object by chromosome and physical position

`pedigree(object)`: accessor for pedigree slot. Returns an object of class `Pedigree`.

`trios(object)`: Returns the `data.frame` stored in slot `trios` of the `Pedigree` class object stored in slot `pedigree` of the `TrioSet` object.

Compute the minimum distance

`calculateMindist(object)`: calculate the minimum distance for an object of class TrioSet.

Posterior summaries

`computeBayesFactor(object, ranges, mad.marker,    mad.sample, returnEmission=FALSE, verbose=TRU` Compute posterior probabilities for the trio copy number states. The called trio copy number state is the argmax of the posterior probabilities. See examples below for computing the standard deviation of the markers for argument `mad.marker` and the standard deviation of the samples for argument `mad.sample`.

Visualization

`xyplot(x, data, ...)`

Miscellaneous

`updateObject(object)`: Currently, this method only checks the class of the `featureData` slot. If the class is `AnnotatedDataFrame`, the `featureData` is updated to the `GenomeAnnotatedDataFrame` class.

Author(s)

R. Scharpf

See Also

`TrioSet`, `TrioSetList`, `calculateMindist`

Examples

```
showClass("TrioSet")
## instantiate a TrioSet with an array of log R ratios (logRR) and B
## allele frequencies
library(MinimumDistance)
TrioSet()

        data(trioSetListExample)
        trioSet <- stack(trioSetList)
```

`TrioSetList` *Constructs and object of class* `TrioSetList`

Description

Constructs and object of class `TrioSetList`

Usage

```
TrioSetList(chromosome=integer(), pedigreeData=Pedigree(), sample.sheet, row.names=NULL, lrr, ba
```

Arguments

`chromosome`	Vector of integers indicating which chromosomes are stored in the `TrioSetList` object.
`pedigreeData`	A object of class `Pedigree`.
`sample.sheet`	A `data.frame` of sample-level covariates.
`row.names`	If `sample.sheet` is non-missing, `row.names` must the specified. The `row.names` must match the sample identifiers stored in the argument to `pedigreeData`. If `sample.sheet` is missing, `row.names` is ignored.
`lrr`	Matrix of log R Ratios
`baf`	matrix of B allele frequencies
`featureData`	Object of class `AnnotatedDataFrame` containing feature annotation. If missing, the argument `cdfname` must be specified.
`cdfname`	A character string providing the name of the annotation package.

Value

An object of class `TrioSetList`

See Also

`TrioSetList`, `phenoData`

Examples

```
TrioSetList()
TrioSetList(chromosome=1:22)

## A more realistic example
## Note that a data.frame containing covariates on the samples can be
## passed through the sample.sheet argument
library(human610quadv1bCrlmm)
path <- system.file("extdata", package="MinimumDistance")
load(file.path(path, "pedigreeInfo.rda"))
load(file.path(path, "sample.sheet.rda"))
load(file.path(path, "logRratio.rda"))
load(file.path(path, "baf.rda"))
nms <- paste("NA",substr(sample.sheet$Sample.Name, 6, 10),sep="")
trioSetList <- TrioSetList(lrr=logRratio, ## must provide row.names
   baf=baf,
   pedigree=Pedigree(pedigreeInfo),
   sample.sheet=sample.sheet,
   row.names=nms,
   cdfname="human610quadv1bCrlmm")
motherPhenoData(trioSetList)
fatherPhenoData(trioSetList)
offspringPhenoData(trioSetList)
## log R ratios for the first trioSetList element
str(lrr(trioSetList)[[1]])
## B allele frequencies for the first trioSetList element
str(baf(trioSetList)[[1]])
```

`TrioSetList-class` *Class* `"TrioSetList"`

Description

A container storing pedigree information, as well as low-level statistical summaries used for copy number estimation: the log R ratios and B allele frequencies. The list structure is organized by chromosome, where each element of the list is a `TrioSet` object.

Objects from the Class

Objects from the class can be initialized by:

`new("TrioSetList")`: Instantiate an empty container.

`TrioSetList()`: See `TrioSetList` .

Slots

`assayDataList`**:** list of arrays containing log R ratios and B allele frequencies.

`featureDataList`**:** list of containing feature annotation. Each element of the list is a `AnnotatedDataFrame`.

`pedigree`**:** Object of class `"Pedigree"`. Contains information on the trio-relationships.

`motherPhenoData`**:** Object of class `"AnnotatedDataFrame"`. Contains sample-level covariates for the mother.

`fatherPhenoData`**:** Object of class `"AnnotatedDataFrame"`. Contains sample-level covariates for the father.

`phenoData`**:** Object of class `"AnnotatedDataFrame"`. Contains sample-level covariates for the offspring.

`chromosome`**:** Integer vector indicating which autosomes are contained in the `TrioSetList` object.

Accessors

In the following accessor descriptions, object is a `TrioSetList`:

`object$NAME`: Extract phenotype 'NAME' for offspring.

`"[["`: Extract a `TrioSet` object.

`allnames(object)`: Returns character vector of all the sample names. Note that `sampleNames` of a `TrioSetList` object is not the same. In particular, `sampleNames(object)` returns only the character vector of offspring ids which uniquely identify a trio. Hence, a separate method, `allNames`, is supplied when all the sample ids in the data set are required. Finally, note that the length of the vector returned by `allNames` is not necessarily a multiple of 3 as mothers and fathers with multiple offspring would be included in multiple trios.

`annotation(object)`: character string indicating the array platform

`baf(object)`: Returns a list of B allele frequencies for each chromosome. Each element in the list is a 3-dimensional array (features x trios x samples).

`dims(object)`: Return dimensions of the low-level statistical summaries (log R ratios and B allele frequencies) for each `TrioSet` element in the `TrioSetList`.

`fatherNames(object)`: character vector of father ids.

`fatherPhenoData(object)`: Extract a `AnnotatedDataFrame` of the sample-level covariates for the father.

`length(object)`: The number of chromosomes.

`lrr(object)` Returns list of log R ratios for each chromosome. Each element in the list is a 3-dimensional array (features x trios x samples).

`mad2(object, byrow=TRUE)` Calculates the median absolute deviation (MAD) of the log R ratios in object. When `byrow` is `TRUE`, the MAD is calculated for each marker across all offspring. When `byrow` is `FALSE`, the MAD is computed across all autosomal markers for the fathers, mothers, and offspring. The former provides a robust estimate of the marker-level variance across samples, whereas the latter provides an estimate of variance for the samples.

`mindist(object) <- value`: Assigns the minimum distance to each `TrioSet` element. `value` is a `list`.

`mindist(object)`: Accessor for the minimum distance in each `TrioSet` element. Returns a `list`.

`motherNames(object)`: Character vector of sample identifiers for the mothers.

`motherPhenoData(object)`: Extract a `AnnotatedDataFrame` of the sample-level covariates for the mother.

`ncol(object)`: the number of trios, or equivalently the number of offspring.

`nrow(object)`: The number of features across all `TrioSet` elements.

`offspringNames(object)`: Character vector of offspring identifiers. Note that the offspring ids uniquely identify a trio, and the method `sampleNames` will return the same result.

`order(object)`: order `TrioSet` object by chromosome and physical position

pedigree `signature(object = "TrioSetList")`: Accessor for pedigree information. See also `Pedigree`

`offspringPhenoData(object)`: Extract a `AnnotatedDataFrame` of the sample-level covariates for the offspring. Alternatively, use `phenoData(object)`.

`phenoData(object)`: Extract a `AnnotatedDataFrame` of the sample-level covariates for the offspring.

`sampleNames(object)`: Character vector of unique identifiers for father-mother-offspring trio. As the offspring id uniquely identifies a trio,`sampleNames` returns the same vector of ids as `offspringNames`

`trios(object)`: Returns a `data.frame` of the trios. Each row in the data.frame contains the sample identifiers for the father, mother, and offspring. Parents with multiple offspring will appear in multiple rows.

Subsetting

`x[i, j]`: `i` selects the list elements. `j` selects the trio for each list element. The list elements have class `TrioSet`.

Stacking a `TrioSetList` object

A `TrioSetList` object contains a list of elements of class `TrioSet`. Each list element corresponds to one chromosome. A `TrioSet` object can be constructed from a `TrioSetList` by stacking the `TrioSet` elements:

`stack(object)`: creates a `TrioSet` object from a `TrioSetList` object.

Compute the minimum distance

`calculateMindist(object)`: Compute the minimum distance from the list of log R ratio arrays. Returns a list of matrices of the minimum distance; each element in the list is the minimum distance for one chromosome.

Posterior summaries

`computeBayesFactor(object, ranges, mad.marker,    mad.sample, returnEmission=FALSE, verbose=TRU`
Compute posterior probabilities for the trio copy number states. The called trio copy number state is the argmax of the posterior probabilities. See examples below for computing the standard deviation of the markers for argument `mad.marker` and the standard deviation of the samples for argument `mad.sample`.

Miscellaneous

`updateObject(object)`: Currently, this method only checks the class of the elements in the `featureDataList` slot. If the elements are `AnnotatedDataFrame` instead of `GenomeAnnotatedDataFrame`, the `featureDataList` slot is updated.

Author(s)

R. Scharpf

See Also

`TrioSet`, `Pedigree`

Examples

```
showClass("TrioSetList")
TrioSetList()
data(trioSetExample)
```

`trioSetListExample` *An example of a* `TrioSetList` *object*

Description

A `TrioSetList` object instantiated from HapMap samples arrayed on a high-throughput Illumina genotyping platform.

Usage

```
data(trioSetListExample)
```

Format

`TrioSetList` object

Details

Each element in the `TrioSetList` is a `TrioSet`.

Source

Two HapMap trios.

Examples

```
data(trioSetListExample)
class(trioSetList[[1]])
```

`TrioSetListLD` *Constructor for TrioSetList class for large data*

Description

Constructor for TrioSetList class for large data

Usage

```
TrioSetListLD(path, fnames, ext="", samplesheet, row.names, pedigreeData, featureData, annotatio
```

Arguments

`path`	Path to plain-text files containing log R ratios and B allele frequencies. Files should contain data for a single sample.
`fnames`	Character string providing the filenames.
`ext`	Character string indicating whether the `fnames` has a file extension (e.g., ".txt")
`samplesheet`	(Optional) `data.frame` containing phenotypic / experimental covariates on the samples. Note that if `samplesheet` is provided, `row.names` must be specified.

`row.names`	Character vector indicating the sample id for each row in `samplesheet`. `row.names` should be unique and, ideally, correspond to `fnames`,
`pedigreeData`	An object of class `Pedigree`.
`featureData`	A `GenomeAnnotatedDataFrame`
`annotationPkg`	Character string indicating the annotation package used to extract information on the features (chromosome, physical position, and whether the feature is polymorphic ('isSnp')).
`outdir`	Character string indicating the path for storing `ff` objects. Ignored if the **ff** package is not loaded.

Details

If the **ff** package is loaded, the `assayData` elements will be of class `ff_array`. Otherwise, the `assayData` elements will be ordinary arrays. For large datasets (or for computers with limited RAM), loading the **ff** may be required.

Value

A `TrioSetList` object

Author(s)

R. Scharpf

See Also

`TrioSetList`

Examples

```
if(require("ff")){
library(ff)
ldPath(tempdir())
path <- system.file("extdata", package="MinimumDistance")
fnames <- list.files(path, pattern="[FMO].txt")
trioSetListff <- TrioSetListLD(path=path,
       fnames=fnames,
       pedigreeData=Pedigree(data.frame(F="F.txt",
       M="M.txt", O="O.txt")),
       annotationPkg="human610quadv1bCrlmm",
       outdir=ldPath())
}
```

`xypanelMD` *Panel function for plotting log R ratios and B allele frequencies for a trio*

Description

This is a panel function for xyplot function in lattice that is useful for plotting log R ratios and B allele frequencies for a trio

Usage

```
xypanelMD(x, y, id, gt, is.snp, range, cex, col.hom = "grey20", fill.hom =
"lightblue", col.het = "grey20", fill.het = "salmon", col.np = "grey20",
fill.np = "grey60", show.state = TRUE, lrr.segs, md.segs, ...,
subscripts)
```

Arguments

`x`	Physical position in MB.
`y`	Either the log R ratios, minimum distance, or B allele frequency.
`id`	The id for the trio.
`gt`	Genotype call (if available)
`is.snp`	Indicator for whether the marker is polymorphic.
`range`	A genomic interval of class `RangedDataHMM`.
`cex`	Size of plotting symbols.
`col.hom`	Color for homozygous genotypes.
`fill.hom`	Fill color for homozygous genotypes (ignored for some plotting symbols).
`col.het`	Color for heterozygous genotypes.
`fill.het`	Fill color for heterozygous genotypes (ignored for some plotting symbols).
`col.np`	Color for nonpolymorphic markers
`fill.np`	Fill color for nonpolymorphic markers (ignored for some plotting symbols).
`show.state`	Whether to display the inferred copy number state stored in the object passed to the `range` argument.
`lrr.segs`	Object of class `RangedDataCBS` containing the results of the CBS segmentation of the log R ratios. Optional
`md.segs`	Object of class `RangedDataCBS` containing the results of the CBS segmentation of the minimum distance. Optional
`...`	Additional arguments to panel.xyplot.
`subscripts`	See lattice xyplot.

Details

This function is not typically called directly, but is passed to the `panel` argument in the `xyplot` method.

Value

Nothing is returned.

Author(s)

R. Scharpf

See Also

`xyplot`, `xypanel`, `panel.xyplot`

Examples

```
library(oligoClasses)
data(trioSetListExample)
trioSet <- stack(trioSetList)
mindist(trioSet) <- calculateMindist(lrr(trioSet))
data(map.segs)
data(lrr.segs)
data(md.segs)
## select a range with a possible de novo copy number alteration in the offspring
rd <- map.segs[which(isDenovo(state(map.segs)))[1], ]
figs <- MinimumDistance:::xyplotTrioLrrBaf(rd=rd,
   object=trioSet,
   frame=200e3,
   ylab="log R ratio and BAFs",
   xlab="physical position",
   panel=MinimumDistance:::xypanelTrioBaf,
   scales=list(y=list(alternating=1)),
   lrr.segments=lrr.segs,
   md.segments=md.segs,
   layout=c(1, 4), ylim=c(-3,1.5),
   par.strip.text=list(lines=0.8, cex=0.6))
```

Index