Package ‘pint’
November 22, 2016

Type Package
Title Pairwise INTEGRation of functional genomics data
Version 1.24.0
Date 2014-09-14
Author Olli-Pekka Huovilainen <ohuovila@gmail.com> and Leo Lahti <leo.lahti@iki.fi>
Maintainer Olli-Pekka Huovilainen <ohuovila@gmail.com>
Description Pairwise data integration for functional genomics, including tools for DNA/RNA/miRNA dependency screens.
License BSD_2_clause + file LICENSE
URL https://github.com/antagomir/pint
BugReports https://github.com/antagomir/pint/issues
LazyLoad yes
Depends mvtnorm, methods, graphics, Matrix, dmt
biocViews aCGH, GeneExpression, Genetics, DifferentialExpression, Microarray
NeedsCompilation no

R topics documented:

ChromosomeModels-class ................................................. 2
fitbyname ................................................................. 4
geneCopyNum .................................................................. 5
GeneDependencyModel-class ............................................. 6
geneExp ........................................................................... 7
ChromosomeModels-class

Class "ChromosomeModels"

Description

Collection of dependency models fitting two data sets in particular chromosome.

Objects from the Class

Function `screen.cgh.mrna` and `screen.cgh.mir` returns an object of this class.

Slots

- `models` a list of `GeneDependencyModels`
- `chromosome` the number of chromosome
- `method` a string with name of the method used in dependency models
- `params` a list of parameters of the used method

Methods

- `[` signature(x = "ChromosomeModels"): Returns the model from the list or returns the dependency models of the arm specified with 'p' or 'q'
- `[<-` signature(x = "ChromosomeModels"): Attaches the a model to the list
- `getChromosome` signature(model = "ChromosomeModels"): Returns the chromosome
- `getArm` signature(model = "ChromosomeModels"): Returns a vector of arms where corresponding dependency model has been calculated.
- `getLoc` signature(model = "ChromosomeModels"): Returns a vector of locations of the genomic dependency models.
- `getScore` signature(model = "ChromosomeModels"): Returns a vector of the scores of the genomic dependency models.
- `getPArm` signature(model = "ChromosomeModels"): Returns the dependency models of the p arm which is of class `ChromosomeModels`
- `getQArm` signature(model = "ChromosomeModels"): Returns the dependency models of the q arm which is of class `ChromosomeModels`
**ChromosomeModels-class**

- `getModelMethod` signature(model = "ChromosomeModels"): Returns the name of the used method.
- `getParams` signature(model = "ChromosomeModels"): Returns a list of used parameters for the method.
- `getWindowSize` signature(model = "ChromosomeModels"): Returns the size of the window used in the dependency models.
- `topGenes` signature(model = "ChromosomeModels", num = "numeric"): Returns a vector of given number of names of the genes which have the highest dependency score. With default value `num = NA` returns all the genes.
- `topModels` signature(model = "ChromosomeModels", num = "numeric"): Returns a list with given number of dependency models which have the highest dependency score. By default returns one model.
- `isEmpty` signature(model = "ChromosomeModels"): Returns TRUE if model has no dependency models.
- `orderGenes` signature(model = "ChromosomeModels"): Returns a data frame with gene names and their model scores sorted.
- `findModel` signature(model = "ChromosomeModels"): Finds a dependency model by gene name and returns it.
- `as.data.frame` signature(x = "ChromosomeModels"): converts dependency models as a dataframe with each row representing a dependency models for one gene. The columns are: `geneName`, `dependencyScore`, `chr`, `arm`, `loc`. If arm information has not been given to screening function, `arm` column is omitted.

**Author(s)**

Olli-Pekka Huovilainen <ohuovila@gmail.com>

**See Also**

For calculation of dependency models for chromosomal arm: `screen.cgh.mrna`. This class holds a number of `GeneDependencyModel` objects. For plotting dependency scores see `dependency score plotting`. Dependency models for whole genome: `GenomeModels`.

**Examples**

```r
data(chromosome17)

## calculate dependency models over chromosome 17
model17 <- screen.cgh.mrna(geneExp, geneCopyNum, windowSize = 10, chr = 17)

model17

## Information of the dependency model which has the highest dependency score
topGenes(model17, 1)

## Finding a dependency model by its name
findModel(model17, "ENSG00000129250")

## Information of the first dependency model
model17[[1]]

#Plotting
```
plot(model17)

# genes in p arm with the highest dependency scores
topGenes(model17[['Var']][['p']], 5)

---

**fit.byname**

*Fit dependency model around one gene between two data sets.*

**Description**

Takes a window from two datasets around chosen gene and fits a selected dependency model between windows.

**Usage**

```r
fit.cgh.mir.byname(X, Y, geneName, windowSize, ...)
fit.cgh.mrna.byname(X, Y, geneName, windowSize, ...)
```

**Arguments**

- **X, Y**
  - Data sets. Lists containing the following items:
    - **data** Data in a matrix form. Genes are in columns and samples in rows. e.g. gene copy number.
    - **info** Data frame which contains following information about genes in data matrix.
      - **chr** Factor indicating the chromosome for the gene: (1 to 23, or X or Y)
      - **arm** Factor indicating the chromosomal arm for the gene ('p' or 'q')
      - **loc** Location of the gene in base pairs.
    - **pint.data** can be used to create data sets in this format.
- **geneName** The dependency model is calculated around this gene.
- **windowSize** Size of the data window.
- **...** Arguments to be passed to function `fit.dependency.model`

**Details**

See **fit.dependency.model** for details about dependency models and parameters.

**Value**

DependencyModel

**Author(s)**

Olli-Pekka Huovilainen <ohuovila@gmail.com> and Leo Lahti <leo.lahti@iki.fi>
References


See Also

Results from this function: DependencyModel, fit.dependency.model. Calculating dependency models to chromosomal arm, chromosome or genome screen.cgh.mrna. For calculation of latent variable z: link(z.expectation).

Examples

```r
data(chromosome17)
model <- fit.cgh.mrnabyname(geneExp,geneCopyNum,"ENSG00000132361",10)
## With different model parameters (pCCA)
model2 <- fit.cgh.mrnabyname(geneExp,geneCopyNum,"ENSG00000132361",10,zDimension=5,priors=list(Nm.wxwy.sigma = NULL))
```

geneCopyNum

<table>
<thead>
<tr>
<th>Gene copy number data in chromosome 17</th>
</tr>
</thead>
</table>

description

Preprocessed gene copy number (aCGH) data for 51 patients in chromosome 17.

Usage

data(chromosome17)

Format

A list which contain the following data:

data  gene copy number data in matrix form. Genes are in columns and samples in rows

info  Data frame which contains following information about genes in data matrix.

    chr  Factor indicating the chromosome for the gene (1 to 23, or X or Y

    arm  Factor indicating the chromosomal arm for the gene (‘p’ or ‘q’)

    loc  Location of the gene in base pairs.

---

**GeneDependencyModel-class**

*Class* "GeneDependencyModel"

---

**Description**

A Genomic Dependency model for two data sets

**Objects from the Class**

Used to represent individual dependency models for screening inside ChromosomeModels.

**Slots**

- `loc` middle location of the window in base pairs
- `geneName` name of the gene in the middle of the window
- `chromosome` Chromosome where the dependency model is calculated
- `arm` Chromosome arm where the dependency model is calculated
- `W` a list of X, Y and total components containing the relationship between two data sets; for dependency model for one dataset, only total is given
- `phi` a list of X, Y and total components containing the data set specific covariances; for dependency model for one dataset, only total is given
- `score` score for fitness of model
- `method` name of the used method
- `params` list of parameters used in dependency model
- `data` The data used to calculate the dependency model
- `z` The latent variable Z

**Extends**

Class DependencyModel directly.

**Methods**

- `setLoc<-` signature(model = "GeneDependencyModel"): sets models location
- `setGeneName<-` signature(model = "GeneDependencyModel"): sets models gene name
- `setChromosome<-` signature(model = "GeneDependencyModel"): sets models chromosome
- `setArm<-` signature(model = "GeneDependencyModel"): sets models chromosome arm
- `getLoc` signature(model = "GeneDependencyModel"): Returns the middle location of the window
- `getGeneName` signature(model = "GeneDependencyModel"): Returns the name of the gene in the middle of window
**geneExp**

getChromosome signature(model = "GeneDependencyModel"): Returns the chromosome
getArm signature(model = "GeneDependencyModel"): Returns the chromosome arm
getWindowSize signature(model = "GeneDependencyModel"): Returns the size of window
getZ signature(model = "GeneDependencyModel"): Calculates the expectation of latent variable Z. The original data is needed as arguments as given to screen function

Author(s)

Olli-Pekka Huovilainen <ohuovila@gmail.com>

See Also

For calculation of dependency models for chromosomal arm, chromosome or genome: screen.cgh.mrna. Dependency models for whole chromosome: ChromosomeModels. Dependency models for whole genome: GenomeModels. For plotting dependency scores see dependency score plotting.

Examples

data(chromosome17)

# First genomic dependency model from screening chromosomal arm
models <- screen.cgh.mrna(geneExp, geneCopyNum, 10, chr=17, arm="p")
model <- models[[1]]

# Printing information of the model
model

# Latent variable Z
getZ(model, geneExp,geneCopyNum)

# Contributions of samples and variables to model
plot(model,geneExp,geneCopyNum)

---

geneExp Gene expression data in chromosome 17

Description

Preprocessed gene expression levels of 51 patients in chromosome 17.

Usage

data(chromosome17)

Format

A list which contain the following data:

data gene expression data in matrix form. Genes are in columns and samples in rows
info Data frame which contains following information about genes in data matrix.
  chr Factor of chromosome where the gene is. (1 to 23 or X or Y
  arm Factor of arm of the chromosome arm where the gene is. (‘p’ or ‘q’)
  loc Location of the gene from centromere in base pairs.

**Source**

Collection of dependency models fitting two data sets in whole genome. The dependency models are in a list of *ChromosomeModels* (which represents each chromosome) that have a list of dependency models in that chromosomal arm.

**Objects from the Class**

Function *screen.cgh.mrna* and *screen.cgh.mir* returns an object of this class.

**Slots**

- **chromosomeModels** a list of *ChromosomeModels* of all chromosomes
- **method** a string with name of the method used in dependency model
- **params** a list of parameters of the method

**Methods**

- [[ signature(x = "GenomeModels"): Returns a *ChromosomeModels* from the list. X and Y chromosomes can be accessed with 23 and 24 or 'X' and 'Y'

- [[<- signature(x = "GenomeModels"): Attaches a *ChromosomeModels* to the list. X and Y chromosomes can be accessed with 23 and 24 or 'X' and 'Y'

- **getModelMethod** signature(model = "GenomeModels"): Returns the name of the used method

- **getParams** signature(model = "GenomeModels"): Returns a list of used parameters for the method

- **getChr** signature(model = "GenomeModels"): Returns the chromosome

- **getWindowSize** signature(model = "GenomeModels"): Returns the size of the window used in the dependency models.

- **getModelNumbers** signature(model = "GenomeModels"): Returns the total number of the dependency models.

- **topGenes** signature(model = "GenomeModels", num = "numeric"): Returns a vector of given number of names of the genes which have the highest dependency score. With default value num = NA returns all the genes.

- **topModels** signature(model = "GenomeModels", num = "numeric"): Returns a list with given number of dependency models which have the highest dependency score. By default returns one model.

- **orderGenes** signature(model = "GenomeModels"): Returns a data frame with gene names and their model scores sorted

- **findModel** signature(model = "GenomeModels"): Finds a dependency model by gene name and returns it.

- **as.data.frame** signature(x = "GenomeModels"): converts dependency models as a dataframe with eachs row representing a dependency model for one gene. The columns are: geneName, dependencyScore, chr,
**Description**

Mainly for internal use. Pick neighboring probes for a given probes based on chromosomal locations.

**Usage**

```r
get.neighboring.probes(X, Y, chr, max.dist, control.arms = TRUE, remove.duplicates = TRUE)
```

**Arguments**

- `X`: Data object. See help(screen.cgh.mrna). For instance, geneExp from our example data set.
- `Y`: Data object. See help(screen.cgh.mrna). For instance, geneCopyNum from our example data set.
- `chr`: chromosome to investigate
- `max.dist`: consider probes within this distance
- `control.arms`: take chromosomal arm information into account
- `remove.duplicates`: for each probe, list exactly one (the closest) match. This is useful in one-to-many matching situations where duplicates may not be desired.

**Value**

A list with matched indices for the X and Y data sets.

**Author(s)**

Leo Lahti <leo.lahti@iki.fi>

**References**

See citation("pint")

**Examples**

# Intended for internal use.
get.neighs

Description

Matching function called by the get.neighboring.probes

Usage

get.neighs(X, Y, xchrinds, ychrinds, max.dist, remove.duplicates = TRUE)

Arguments

X        Data object. See help(screen.cgh.mrna). For instance, geneExp from our example data set.
Y        Data object. See help(screen.cgh.mrna). For instance, geneCopyNum from our example data set.
xchrinds chromosomal indices for the probes in X data set
ychrinds chromosomal indices for the probes in Y data set
max.dist consider probes within this chromosomal distance
remove.duplicates remove duplicate probes in many-to-one matching cases, use the closest match

Value

List of matched X and Y indices.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

See citation("pint")

Examples

# Intended for internal use.
join.top.regions  

Merge the overlapping top chromosomal regions.

Description

Select the top models that exceed the threshold and merge the overlapping windows. Useful for interpreting the results.

Usage

```r
join.top.regions(model, feature.info, quantile.th = 0.95, augment = FALSE)
```

Arguments

- **model**: Object of `ChromosomeModels` or `GenomeModels` class.
- **feature.info**: A data frame containing annotations for genes. For instance the `geneExp$info` table from our example data set (see `data(chromosome17)`).
- **quantile.th**: Threshold to define what quantile of the genes to include in the top region list, based on dependency scores for each gene.
- **augment**: If TRUE, list also genes that were not used for modeling but available in the annotations (feature.info) and residing within the same region.

Value

A list; each element is a vector of gene names that correspond to one continuous region.

Author(s)

Leo Lahti <leo.lahti@iki.fi>

References

See citation("pint")

See Also

- `summarize.region.parameters`

Examples

```r
## NOT RUN
# top.regions <- join.top.regions(model, geneExp$info, quantile.th = 0.95)
```
order.feature.info  
*Order the gene information table by chromosomal locations.*

**Description**

Order the gene information table by chromosomal locations. Removes genes with no location information.

**Usage**

```r
order.feature.info(feature.info)
```

**Arguments**

- `feature.info`  
  A data frame containing at least the following fields: geneName, chr, and loc.

**Value**

An ordered data frame.

**Author(s)**

Leo Lahti <leo.lahti@iki.fi>

**References**

See citation("pint")

**Examples**

```r
## NOT RUN
feature.info.ordered <- order.feature.info(feature.info)
```

---

**pint.data**  
*Forms a data set and pairs samples in two data sets.*

**Description**

Forms a data set for use in functions in 'pint' package (e.g. `screen.cgh.mrna`). Pairs samples in two data sets.

**Usage**

```r
pint.data(data, info, impute = TRUE, replace.inf = TRUE, remove.duplicates)
pint.match(X, Y, max.dist = 1e7, chrs = NULL, useSegmentedData = FALSE, impute = TRUE, replace.inf = TRUE, remove.duplicates = TRUE)
```
Arguments

data  Probe-level data in a matrix or data frame.
info  Location, chromosome, and chromosome arm. Information of the probes as data frame. Location can be given either as loc or bp, which is middle location of probe, or as start and end. Chromosome arm is given as arm and chromosome as chr.
X, Y  Data sets to be paired.
max.dist  maximum distance between paired genes in base pairs.
chrs  Use to pick a subset of chromosomes in the data. By default, all chromosomes will be used.
useSegmentedData  Logical. If FALSE, rows with identical data are removed (option for pint.match)
remove.duplicates  Logical. If TRUE, rows with identical data are removed (in pint.data), or duplicate signals from many-to-one matches are removed (in pint.match)
impute  Logical. If TRUE, missing values are imputed by replacing them with random samples from a Gaussian distribution following the mean and standard deviation of the non-missing data points from the same sample.
replace.inf  Logical. If TRUE, replace infinite values with highest non-infinite values seen in the data. Otherwise the calculation will halt.

Details

Function pint.match goes through every sample in X and finds the nearest sample in Y which is in the same chromosome arm. If more than one sample in X has same nearest sample in Y, all but one is discarded. Samples with longer distance than max.dist are discarded.

Value

pint.data returns a list with a matrix with sample data and a data frame with chr (chromosome), arm (chromosome arm) and loc (location).

pint.match return a list with two data sets. These can be used in screen.cgh.mrna function.

Author(s)

Olli-Pekka Huovilainen <ohuovila@gmail.com>

See Also

screen.cgh.mrna, screen.cgh.mir, fit.cgh.mir.bynname

Examples

data(chromosome17)

newData <- pint.match(geneExp,geneCopyNum,max.dist=1000)
plot

Dependency score plotting

Description

Plot the contribution of the samples and variables to the dependency model or dependency model fitting scores of chromosome or genome.

Usage

## S3 method for class 'GeneDependencyModel'
plot(x, X, Y, ann.types = NULL, ann.cols = NULL, legend.x = 0, legend.y = 1, legend.xjust = 0, legend.yjust = 1, order = FALSE, cex.z = 0.6, cex.WX = 0.6, cex.WY = 0.6, ...)

## S3 method for class 'ChromosomeModels'
plot(x, hilightGenes = NULL, showDensity = FALSE, showTop = 0, topName = FALSE, type = 'l', xlab = 'gene location', ylab = 'dependency score', main = NULL, pch = 20, cex = 0.75, tpch = 3, tcex = 1, xlim = NA, ylim = NA, ...)

## S3 method for class 'GenomeModels'
plot(x, hilightGenes = NULL, showDensity = FALSE, showTop = 0, topName = FALSE, onePlot = FALSE, type = 'l', ylab = "Dependency Scores", xlab = "Gene location (chromosome)", main = "Dependency Scores in All Chromosomes", pch = 20, cex = 0.75, tpch = 20, tcex = 0.7, mfrow = c(5,5), mar = c(3,2.5,1.3,0.5), ps = 5, mgp = c(1.5,0.5,0.0), ylim=NA, ...)

Arguments

x GeneDependencyModel-class, ChromosomeModels-class, GenomeModels-class; models to be plotted.
X, Y data sets used in dependency modeling.
ann.types a factor for annotation types for samples. Each value corresponds one sample in datasets. Colors are used to indicate different types.
ann.cols colors used to indicate different annotation types. Gray scale is used if 'NULL' given.
legend.x, legend.y the x and y co-ordinates to be used to position the legend for annotation types.
legend.xjust, legend.yjust how the legend is to be justified relative to the legend x and y location. A value of 0 means left or top justified, 0.5 means centered and 1 means right or bottom justified.
order logical; if 'TRUE', values for sample contributions are ordered according to their values.
cex.z, cex.WX, cex.WY Text size for variable names.
hilightGenes vector of strings; Name of genes to be hilighted with dots.
showDensity logical; if ‘TRUE’ small vertical lines are drawn in the bottom of the plot under each gene.

showTop numeric; Number of models with highest dependencies to be highlighted. A horizontal dashed line is drawn to show threshold value. With 0 no line is drawn.

topName logical; If TRUE, gene names are printed to highlighted models with highest dependencies. Otherwise highlighted models are numbered according to their rank in dependency score.

type, xlab, ylab, main
plot type and labels. See plot for details. A text for chromosome (and arm if only models from one arm is plotted) is used in main if NULL is given. In plot.GenomeModels, ylab and xlab affect only if onePlot is TRUE.

onePlot If TRUE, all dependency scores are plotted in one plot window. Otherwise one plot window is used for each chromosome.

pch, cex symbol type and size for hilightGenes. See points for details.

tpch, tcex symbol type and size for genes with highest scores. See points for details.

ylim, xlim axis limits. Default values are calculated from data. Lower limit for y is 0 and upper limit is either 1 or maximum score value. X limits are gene location range. See plot for details.

mfrow, mar, ps, mgp chromosome plots’ layout, marginals, text size and margin line. See par for details.

... optional plotting parameters

Details
Function plots scores of each dependency model of a gene for the whole chromosome or genome according to used method. plot(x, cancerGenes = NULL, showDensity = FALSE, ...) is also usable and chosen according to class of models.

Author(s)
Olli-Pekka Huovilainen <ohuovila@gmail.com>

References

See Also
DependencyModel-class, ChromosomeModels-class, GenomeModels-class, screen.cgh.mrna, screen.cgh.mir

Examples

data(chromosome17)

# pSimCCA model on chromosome 17p
models17pSimCCA <- screen.cgh.mrna(geneExp, geneCopyNum, 10, 17, ‘p’)
plot(models17pSimCCA,
   hilightGenes=c("ENSG00000108342", "ENSG00000108298"), showDensity = TRUE)
## Dependency model around 50th gene

```r
model <- models17ppSimCCA[[50]]
```

## Example annotation types

```r
ann.types <- factor(c(rep("Samples 1 - 10", 10), rep("Samples 11 - 51", 41)))
plot(model, geneExp, geneCopyNum, ann.types, legend.x = 40, legend.y = -4,
     order = TRUE)
```

### Description

Fits dependency models to chromosomal arm, chromosome or the whole genome.

### Usage

```r
screen.cgh.mrna(X, Y, windowSize = NULL, chromosome, arm, method = "pSimCCA", params = list(), max.dist = 1e7, outputType = "models", useSegmentedData = TRUE, match.probes = TRUE, regularized = FALSE)
```

```r
screen.cgh.mir(X, Y, windowSize, chromosome, arm, method = "", params = list(), outputType = "models")
```

### Arguments

- `X, Y` Data sets. It is recommended to place gene/mirna expression data in `X` and copy number data in `Y`. Each is a list with the following items:
  - `data` Data in a matrix form. Genes are in rows and samples in columns. e.g. gene copy number.
  - `info` Data frame which contains following information about genes in data matrix.
    - `chr` Number indicating the chromosome for the gene: (1 to 24). Characters 'X' or 'Y' can be used also.
    - `arm` Character indicating the chromosomal arm for the gene ('p' or 'q')
    - `loc` Location of the gene in base pairs.
  - `pint.data` can be used to create data sets in this format.
- `chromosome` Specify the chromosome for model fitting. If missing, whole genome is screened.
- `arm` Specify chromosomal arm for model fitting. If missing, both arms are modeled.
- `windowSize` Determine the window size. This specifies the number of nearest genes to be included in the chromosomal window of the model, and therefore the scale of the investigated chromosomal region. If not specified, using the default ratio of $1/3$ between features and samples or 15 if the ratio would be greater than 15.
Dependency screening can utilize any of the functions from the package dmt (at CRAN). Particular options include:

- **pSimCCA** probabilistic similarity constrained canonical correlation analysis (Lahti et al. 2009). This is the default method.
- **pCCA** probabilistic canonical correlation analysis (Bach & Jordan 2005).
- **pPCA** probabilistic principal component analysis (Tipping & Bishop 1999).
- **pFA** probabilistic factor analysis (Rubin & Thayer 1982).
- **TPriorpSimCCA** probabilistic similarity constrained canonical correlation analysis with possibility to tune T prior (Lahti et al. 2009).

If anything else, the model is specified by the given parameters.

**method**

Dependency screening can utilize any of the functions from the package dmt (at CRAN). Particular options include:

- **pSimCCA** probabilistic similarity constrained canonical correlation analysis (Lahti et al. 2009). This is the default method.
- **pCCA** probabilistic canonical correlation analysis (Bach & Jordan 2005).
- **pPCA** probabilistic principal component analysis (Tipping & Bishop 1999).
- **pFA** probabilistic factor analysis (Rubin & Thayer 1982).
- **TPriorpSimCCA** probabilistic similarity constrained canonical correlation analysis with possibility to tune T prior (Lahti et al. 2009).

If anything else, the model is specified by the given parameters.

**params**

List of parameters for the dependency model.

- **sigmas** Variance parameter for the matrix normal prior distribution of the transformation matrix T. This describes the deviation of T from H.
- **H** Mean parameter for the matrix normal prior distribution of the transformation matrix T.
- **zDimension** Dimensionality of the latent variable.
- **mySeed** Random seed.
- **covLimit** Convergence limit. Default depends on the selected method: 1e-3 for pSimCCA with full marginal covariances and 1e-6 for pSimCCA in other cases.
- **max.dist** Maximum allowed distance between probes. Used in automated matching of the probes between the two data sets based on chromosomal location information.
- **outputType** Specifies the output type of the function. Possible values are "models" and "data.frame".
- **useSegmentedData** Logical. Determines the usage of the method for segmented data.
- **match.probes** To be used with segmented data, or nonmatched probes in general. Using nonmatched features (probes) between the data sets. Development feature, to be documented later.
- **regularized** Regularization by nonnegativity constraints on the projections. Development feature, to be documented later.

**Details**

Function **screen.cgh.mrna** assumes that data is already paired. This can be done with **pint.match**. It takes sliding gene windows with **fixed.window** and fits dependency models to each window with **fit.dependency.model** function. If the window exceeds start or end location (last probe) in the chromosome in the **fixed.window** function, the last window containing the given probe and not exceeding the chromosomal boundaries is used. In practice, this means that dependency score for the last n/2 probes in each end of the chromosome (arm) will be calculated with an identical window, which gives identical scores for these end position probes. This is necessary since the window size has to be fixed to allow direct comparability of the dependency scores across chromosomal windows.

Function **screen.cgh.mir** calculates dependencies around a chromosomal window in each sample in X; only one sample from X will be used. Data sets do not have to be of the same size and X can be considerably smaller. This is used with e.g. miRNA data.

If method name is specified, this overrides the corresponding model parameters, corresponding to the modeling assumptions of the specified model. Otherwise method for dependency models is determined by parameters.

Dependency scores are plotted with **dependency score plotting**.
Value

The type of the return value is defined by the the function argument outputType.

With the argument outputType = "models", the return value depends on the other arguments; returns a `ChromosomeModels` which contains all the models for dependencies in chromosome or a `GenomeModels` which contains all the models for dependencies in genome.

With the argument outputType = "data.frame", the function returns a data frame with eachs row representing a dependency model for one gene. The columns are: `geneName`, `dependencyScore`, `chr`, `arm`, `loc`.

Author(s)

Olli-Pekka Huovilainen <ohuovila@gmail.com> and Leo Lahti <leo.lahti@iki.fi>

References


See Also

To fit a dependency model: `fit.dependency.model`. `ChromosomeModels` holds dependency models for chromosome, `GenomeModels` holds dependency models for genome. For plotting, see: dependency score plotting

Examples

data(chromosome17)

## pSimCCA model on chromosome 17

models17pSimCCA <- screen.cgh.mrna(geneExp, geneCopyNum,
  windowSize = 10, chr = 17)

plot(models17pSimCCA)

## pCCA model on chromosome 17p with 3-dimensional latent variable z

models17pCCA <- screen.cgh.mrna(geneExp, geneCopyNum,
  windowSize = 10,
  chromosome = 17, arm = 'p', method="pCCA",
  params = list(zDimension = 3))

plot(models17pCCA)
summarize.region.parameters

*Summarize overlapping models.*

**Description**

Given a chromosomal region, summarize the model parameters from overlapping models. This heuristics gives a brief summary on average sample and probe effects within the region and aids interpretation. If multiple alteration profiles are detected within the region, the models are grouped and summarization is applied separately for each group containing overlapping models with high similarity.

**Usage**

```r
summarize.region.parameters(region.genes, model, X, Y, grouping.th = 0.9, rm.na = TRUE)
```

**Arguments**

- `region.genes` A vector of gene names determining the investigated region.
- `model` Object of `ChromosomeModels` or `GenomeModels` class.
- `X` Data object. See `help(screen.cgh.mrna)`. For instance, `geneExp` from our example data set.
- `Y` Data object. See `help(screen.cgh.mrna)`. For instance, `geneCopyNum` from our example data set.
- `grouping.th` Similarity threshold for joining neighboring models.
- `rm.na` Remove genes with NA values from the output.

**Details**

Grouping of the models is based on heuristics where highly correlating models (>grouping.th) are merged. Will be improved later.

**Value**

- `z` Mean sample effects, averaged over the overlapping models for each sample.
- `W` Mean probe effects, averaged over the overlapping models for each probe. This is a list with elements `X`, `Y`, corresponding to the two data sets.

**Author(s)**

Leo Lahti &lt;leo.lahti@iki.fi&gt;

**References**

See `citation("pint")`

**See Also**

`merge.top.regions`
window

Examples

```r
# tmp <- summarize.region.parameters(top.region.genes, model, geneExp, geneCopyNum)
# wx <- tmp$W$X
# z <- tmp$z
```

Description

Forms a chosen window of two data matrices to use for `fit.dependency.model` either iteratively picking nearest genes or picking same number of genes from both directions. `sparse.window` forms a window around one sample in the first data set with a number of samples from the second data set.

Usage

```r
fixed.window(X, Y, middleIndex, windowSize)
iterative.window(X, Y, middleIndex, windowSize)
sparse.window(X, Y, xIndex, windowSize)
```

Arguments

- `X` First data set. In `sparse.window` windows will be formed around each sample in this data set.
- `Y` Second data set.
- `middleIndex` Index of middle position for window.
- `xIndex` Index of middle position in `X` for window.
- `windowSize` Number of genes in window. In `sparse.window` `X` has always one sample in window.

Details

Window contains `windowSize` nearest genes. Warning is given if `windowSize` genes is not found in the same chromosomal arm. Data of both data sets is normalised so that each genes data has zero mean.

Value

List of window data:

- `X` window of the first data set
- `Y` window of the second data set
- `loc` location of gene
- `geneName` name of the gene
- `edge` logical; TRUE if iteration to one direction has stopped because edge of data in chromosomal arm has been found.
- `fail` logical; TRUE if chromosomal arm contains less than `windowSize` genes.
z.effects

Author(s)
Olli-Pekka Huovilainen <ohuovila@gmail.com>

See Also
Dependency model fitting: fit.dependency.model

Examples

```r
data(chromosome17)
window <- iterative.window(geneExp, geneCopyNum, 30, 10)
model <- fit.dependency.model(window$X, window$Y)

# Conversion from DependencyModel to GeneDependencyModel so that gene name and location can be stored
model <- as(model,"GeneDependencyModel")
setGeneName(model) <- window$geneName
setLoc(model) <- window$loc
model

window <- fixed.window(geneExp, geneCopyNum, 10, 10)
model <- fit.dependency.model(window$X, window$Y)
model
```

---

z.effects

The model parameters $z$ and $W$

Description
Contribution of each sample to a dependency model, and contribution of each variable.

Usage

```r
z.effects(model, X, Y = NULL)
W.effects(model, X, Y = NULL)
```

Arguments

- `model`: The fitted dependency model.
- `X, Y`: Data sets used in fitting the dependency modeling functions (`screen.cgh.mrna` or `link{fit.dependency.model}`). Note: Arguments must be given in the same order as in `fit.dependency.model` or `screen.cgh.mrna`. Only `X` is needed for dependency model for one data set.

Details

$z.effects$ gives the contribution of each sample to the dependency score. This is approximated by projecting original data to first principal component of $Wz$. This is possible only when the data window is smaller than half the number of samples.

$W.effects$ gives the contribution of each variable to the observed dependency. This is approximated with the loadings of the first principal component of $Wz$

Original data can be retrieved by locating the row in `X` (or `Y`) which has the same variable (gene) name than `model`. 
Value

z.effects gives a projection vector over the samples and W.effects gives a projection vector over the variables.

Author(s)

Olli-Pekka Huovilainen <ohuovila@gmail.com> and Leo Lahti <leo.lahti@iki.fi>

References


See Also

DependencyModel-class, screen.cgh.mrna

Examples

data(chromosome17)
window <- fixed.window(geneExp, geneCopyNum, 150, 10)

## pSimCCA model around one gene
depmodel <- fit.dependency.model(window$X, window$Y)
# Conversion from DependencyModel to GeneDependencyModel so that gene name and location can be stored
depmodel <- as(depmodel, "GeneDependencyModel")
setGeneName(depmodel) <- window$geneName
setLoc(depmodel) <- window$loc
barplot(z.effects(depmodel, geneExp, geneCopyNum))

## Plot the contribution of each genes to the model. Only the X component is plotted
## here since Wx = Wy (in SimCCA)
barplot(W.effects(depmodel, geneExp, geneCopyNum)$X)

## plot.DpenendencyModel shows also sample and variable effects
plot(depmodel,geneExp,geneCopyNum)
Index

*Topic classes
ChromosomeModels-class, 2
GeneDependencyModel-class, 6
GenomeModels-class, 8

*Topic datasets
geneCopyNum, 5
geneExp, 7

*Topic hplot
plot, 14

*Topic iteration
fit.byname, 4
screen, 16

*Topic math
fit.byname, 4
screen, 16
z.effects, 21

*Topic utilities
get.neighboring.probes, 9
get.neighs, 10
join.top.regions, 11
order.feature.info, 12
summarize.region.parameters, 19

[[ (ChromosomeModels-class), 2
[[,ChromosomeModels-method
(ChromosomeModels-class), 2
[[,GenomeModels-method
(GenomeModels-class), 8
[[<- (ChromosomeModels-class), 2
[[<-,ChromosomeModels-method
(ChromosomeModels-class), 2
[[<-,GenomeModels-method
(GenomeModels-class), 8

as.data.frame,ChromosomeModels-method
(ChromosomeModels-class), 2
as.data.frame,GenomeModels-method
(GenomeModels-class), 8

ChromosomeModels, 2, 6–9, 11, 18, 19
ChromosomeModels-class, 2
dependency score plotting, 3, 7, 17, 18
dependency score plotting (plot), 14
DependencyModel, 4–6
findModel (ChromosomeModels-class), 2
findModel,ChromosomeModels-method
(ChromosomeModels-class), 2
findModel,GenomeModels-method
(GenomeModels-class), 8
fit.byname, 4
fit.cgh.mir.byname, 13
fit.cgh.mir.byname (fit.byname), 4
fit.cgh.mrna.byname (fit.byname), 4
fit.dependency.model, 4, 5, 17, 18, 21
fixed.window, 17
fixed.window (window), 20
geneCopyNum, 5
GeneDependencyModel, 2, 3, 9
GeneDependencyModel-class, 6
geneExp, 7
GenomeModels, 3, 7, 11, 18, 19
GenomeModels-class, 8
get.neighboring.probes, 9
get.neighs, 10
getArm (ChromosomeModels-class), 2
getArm,ChromosomeModels-method
(ChromosomeModels-class), 2
getArm,GenDependencyModel-method
(GenomeModels-class), 6
getChromosome (ChromosomeModels-class), 2
getChromosome,ChromosomeModels-method
(ChromosomeModels-class), 2
getChromosome,GenDependencyModel-method
(GenomeModels-class), 6
getGeneName (ChromosomeModels-class), 2
getGeneName,ChromosomeModels-method
(ChromosomeModels-class), 2
getGeneName,GenDependencyModel-method
(GenomeModels-class), 6
getLoc (GenDependencyModel-class), 6
getLoc,ChromosomeModels-method
(ChromosomeModels-class), 2
getLoc,GenDependencyModel-method
(GenomeModels-class), 6
getModelMethod
(ChromosomeModels-class), 2
getModelMethod, ChromosomeModels-method
(ChromosomeModels-class), 2
getModelMethod, GenomeModels-method
(GenomeModels-class), 8
getModelNumbers
(ChromosomeModels-class), 2
getModelNumbers, ChromosomeModels-method
(ChromosomeModels-class), 2
getModelNumbers, GenomeModels-method
(GenomeModels-class), 8
getParams
(ChromosomeModels-class), 2
getParams, ChromosomeModels-method
(ChromosomeModels-class), 2
getParams, GenomeModels-method
(GenomeModels-class), 8
getPArm
(ChromosomeModels-class), 2
getPArm, ChromosomeModels-method
(ChromosomeModels-class), 2
getQArm
(ChromosomeModels-class), 2
getQArm, ChromosomeModels-method
(ChromosomeModels-class), 2
getScore
(GeneDependencyModel-class), 6
getScore, ChromosomeModels-method
(ChromosomeModels-class), 2
getScore, GenomeModels-method
(GenomeModels-class), 6
getWindowSize
(ChromosomeModels-class), 2
getWindowSize, ChromosomeModels-method
(ChromosomeModels-class), 2
getWindowSize, GenomeModels-method
(GenomeModels-class), 8
getz, GeneDependencyModel-method
(GeneDependencyModel-class), 6
isEmpty
(ChromosomeModels-class), 2
isEmpty, ChromosomeModels-method
(ChromosomeModels-class), 2
iterative.window (window), 20
join.top.regions, 11
order.feature.info, 12
orderGenes
(ChromosomeModels-class), 2
orderGenes, ChromosomeModels-method
(ChromosomeModels-class), 2
orderGenes, GenomeModels-method
(GenomeModels-class), 8
par, 15
pint.data, 4, 12, 16
pint.match, 17
pint.match (pint.data), 12
plot, 14, 15
plot, GenomeModels (plot), 14
plot, ChromosomeModels (plot), 14
plot, GenomeModels (plot), 14
points, 15
screen, 16
screen.cgh.mir, 2, 8, 13, 15
screen.cgh.mrna, 2, 3, 5, 7–9, 12, 13, 15, 21, 22
setArm<- (GeneDependencyModel-class), 6
setArm<- (GeneDependencyModel-class), 6
setChromosome<- (GeneDependencyModel-class), 6
setChromosome<- (GeneDependencyModel-class), 6
setGeneName<- (GeneDependencyModel-class), 6
setGeneName<- (GeneDependencyModel-class), 6
setLoc<- (GeneDependencyModel-class), 6
setLoc<- (GeneDependencyModel-class), 6
sparse.window (window), 20
summarize.region.parameters, 19
topGenes
(ChromosomeModels-class), 2
topGenes, ChromosomeModels-method
(ChromosomeModels-class), 2
topGenes, GenomeModels-method
(GenomeModels-class), 8
topModels
(ChromosomeModels-class), 2
topModels, ChromosomeModels-method
(ChromosomeModels-class), 2
topModels, GenomeModels-method
(GenomeModels-class), 8
W.effects (z.effects), 21
window, 20
z.effects, 21