rHVDM

November 11, 2009

R topics documented:

`estimerrors` *computes the standard deviation of the measurement error using pre-calculated tables specific to the plattform or user-defined table*

Description

This method takes as main input and ExpressionSet object, and using pre-calculated tables contained in rHVDM, it returns the same object with an extra slot that contains the standard deviation of the measurement error which is indispensable to then run rHVDM proper. The table can be user-defined. To check which tables are stored in rHVDM, just run the command without arguments (`estimerrors()`), it will return a list of the supported rHVDM plattforms. All parameters of this function are optional, see below for details on the effect of omitting one or more of them. The method assumes the the amount of error is dependent both on the particular microarray and the signal intensity. For more details on how it is computed, please refer to the textual description of rHVDM.

Usage

```
estimerrors(eset,plattid,refchips,errtable)
```

Arguments

`eset`	an object of class ExpressionSet (Biobase), every time this parameter is omitted the function returns a list of supported plattforms (identifier+plattform description).
`plattid`	an optional argument (character format) allowing to specify the plattform identifiers. These identifiers are rHVDM specific. Any identifier will be overriden by a table given as input of the `errtable` argument. In future versions of rHVDM, if this argument and the `errtable` are missing, a search will be done among the supported plattforms to find a matching plattform, based on the individual genes identifiers. For now, omission of both arguments will be conductive to outputting a list of supported plattforms.
`refchips`	a vector of names or column indexes compatible with the ExpressionSet useds (note that this compatibility is not verified by the function, it is up to the user to supply compatible array names). Some microarray might be known to be of not very good quality and although the measurement error will be computed for these chips it is better to leave them out of early stages of the computation. The list to be input should typically contain the least noisy microarrays. This parameter is optional and if omitted, all microarrays are used.
`errtable`	a nx2 table in matrix format. The first column of the matrix should contain reference log of signals in ascending order and the second the variance corresponding to the signals.

Details

Local interpolation and individual array normalisation are used to estimate the standard deviation of the measurement error for each individual transcript on each microarray. The precise method is laid out in the paper cited below, with some minor modifications.

Value

Normally an updated eset is returned. In case some crucial element is missing, the original eset is returned. If the latter is missing too a list of supported plattform is returned (thus, when using this function, it is better to be careful of what lies at the LHS of the assignement arrow).

Note

The HTML report is generated in the working directory.

Author(s)

Martino Barenco

References

M. Barenco, D. Tomescu, D. Brewer, R. Callard, J. Stark, M. Hubank (2006) Ranked predictions of p53 targets using Hidden Variable Dynamic Modelling. *Genome Biology*, **V7(3)**, R25.

See Also

HVDMcheck,training,screening,fitgene

Examples

```
data(HVDMexample)
fiveGybis<-estimerrors(eset=fiveGyMAS5,plattid="affy_HGU133A",refchips=leastnoisychips)
#leastnoisy chips is a list of three chips identifiers stored in the HVDMexample data bun
```

`fitgene.lin` *Fits the optimal kinetic parameter values for a particular gene.*

Description

This method fits the three kinetic parameter values for a particular gene. It returns a list containing the results.

Usage

```
fitgene.lin(eset,gene,tHVDM,transforms,firstguess)
```

Arguments

`eset`	an ExpressionSet object (Biobase)
`gene`	the gene identifier in character format
`transforms`	a vector containing the kinetic parameter identifiers that have to be transformed during optimisation (optional)
`tHVDM`	the output of the training set
`firstguess`	first guess for the fitting (optional, see details)

Details

An exponential transform is set by default for both the basal (Bj) and degradation (Dj) rates (through the transforms argument). This forces the values for both these parameters to be positive. It also helps to reach a better fit. To turn this off let `transforms=c()`. Even in this case the degradation rate will not be allowed to take non positive values as it causes problems with the differential operator used internally. The value in the vector indicates the parameter to be transformed: "Bj": basal rate of transcription, "Sj": sensitivity, "Dj": degrdation rate. The entry label indicates the transform to be applied; presently, only log-transforms are implemented (ie "exp").

This `fitgene()` step can only be applied after a `training()` step. The output to the `training()` step has to be fed through the `tHVDM` argument.

The `firstguess` argument is optional (a first guess is generated internally by default). However a first guess can be supplied by the user which can take several forms. It can either be a vector with three entries containing a first guess for the basal rate, the sensitivity, the degradation rate (in that order). Alternatively, another output from the `fitgene()` function (for example from a gene that has a similar expression profile) can be supplied as a `firstguess` argument.

Value

a list containing the results (see documentation for more details).

Note

Obviously, the expression set given as a `eset` argument has to be the same as the one used for the training step.

Author(s)

Martino Barenco

References

M. Barenco, D. Tomescu, D. Brewer, R. Callard, J. Stark, M. Hubank (2006) Ranked predictions of p53 targets using Hidden Variable Dynamic Modelling. *Genome Biology*, **V7(3)**, R25.

See Also

`training`,`screening`,`HVDMreport`

Examples

```
data(HVDMexample)
tHVDMp53<-training(eset=fiveGyMAS5,genes=p53traingenes,degrate=0.8,actname="p53")
sHVDMcd38<-fitgene.lin(eset=fiveGyMAS5,gene="205692_s_at",tHVDM=tHVDMp53)
```

`fitgene.nl` *Fits the optimal kinetic parameter values for a particular gene and model.*

Description

This method fits the three kinetic parameter values for a particular gene. It returns a list containing the results.

Usage

```
fitgene.nl(eset,gene,tHVDM,transforms,firstguess,model)
```

Arguments

`eset`	an ExpressionSet object (Biobase)
`gene`	the gene identifier in character format
`transforms`	a vector containing the kinetic parameter identifiers that have to be transformed during optimisation (optional)
`tHVDM`	the output of the training set
`firstguess`	first guess for the fitting (optional, see details)
`model`	model specification of the production function in character format" MM (Michelis-Menten, default) or hill (hill function)

Details

An exponential transform is set by default for both the basal (Bj), degradation (Dj) and Kj rates (through the transforms argument). This forces the values for both these parameters to be positive. It also helps to reach a better fit. To turn this off let transforsm=c(). Even in this case the degradation rate will not be allowed to take non positive values as it causes problems with the differential operator used internally. The value in the vector indicates the parameter to be transformed: "Bj": basal rate of transcription, "Vj": sensitivity, "Dj": degrdation rate. The entry label indicates the transform to be applied.

This `fitgene.nl()` step can only be applied after a `training()` step. The output to the `training.nl()` step has to be fed through the `tHVDM` argument.

The `firstguess` argument is optional (a first guess is generated internally by default). However a first guess can be supplied by the user which can take several forms. It can either be a vector with three entries containing a first guess for the basal rate, the sensitivity, the degradation rate (in that order). Alternatively, another output from the `fitgene()` function (for example from a gene that has a similar expression profile) can be supplied as a `firstguess` argument.

The `model` argument is only used to choose the model (MM: Michelis-Menten, hill: Hill function).

Value

a list containing the results (see documentation for more details).

Note

Obviously, the expression set given as a `eset` argument has to be the same as the one used for the training step.

Author(s)

Martino Barenco

References

M. Barenco, D. Tomescu, D. Brewer, R. Callard, J. Stark, M. Hubank (2006) Ranked predictions of p53 targets using Hidden Variable Dynamic Modelling. *Genome Biology*, **V7(3)**, R25.

See Also

`training`,`screening`,`HVDMreport`

Examples

```
#load data and fit a linear model
data(HVDMexample)
rm(fiveGyMAS5)
data(HVDMexample2)
tp532<-training(eset=twodosesMAS5,genes=p53traingenes,degrate=0.8)
#formulate constraints
CONSTRAINTS<-c(350,35)
names(CONSTRAINTS)<-c("trfact1.5Gy.1.4","trfact1.hGy.1.4")
#specify individual gene models
GENEMODELS<-rep("MM",5)
GENEMODELS[c(1,3)]<-"hill"
names(GENEMODELS)<-p53traingenes
```

```
#run the model
tp53hyb<-training.nl(inputHVDM=tp532,constraints=CONSTRAINTS,genemodels=GENEMODELS)

#fit an individual gene
sHVDMex<-fitgene.nl(eset=twodosesMAS5,gene="213293_s_at",tHVDM=tp53hyb)
```

`fitgene` *Fits the optimal kinetic parameter values for a particular gene.*

Description

This method fits the three kinetic parameter values for a particular gene. It returns a list containing the results.

Usage

```
fitgene(eset,gene,tHVDM,transforms,firstguess,criterion)
```

Arguments

`eset`	an ExpressionSet object (Biobase)
`gene`	the gene identifier in character format
`transforms`	a vector containing the kinetic parameter identifiers that have to be transformed during optimisation (optional)
`tHVDM`	the output of the training set
`firstguess`	first guess for the fitting (optional, see details)
`criterion`	model selection in the nonlinear case

Details

An exponential transform is set by default for both the basal (Bj), degradation (Dj) rates (through the transforms argument) and possibly Kj (in case a nonlinear model is used). This forces the values for both these parameters to be positive. It also helps to reach a better fit. To turn this off let transforsm=c(). Even in this case the degradation rate will not be allowed to take non positive values as it causes problems with the differential operator used internally. The value in the vector indicates the parameter to be transformed: "Bj": basal rate of transcription, "Sj": sensitivity, "Dj": degrdation rate. The entry label indicates the transform to be applied; presently, only log-tranforms are implemented (ie "exp").

This `fitgene()` step can only be applied after a `training()` step. The output to the `training()` step has to be fed through the `tHVDM` argument.

The `firstguess` argument is optional (a first guess is generated internally by default). However a first guess can be supplied by the user which can take several forms. It can either be a vector with three entries containing a first guess for the basal rate, the sensitivity, the degradation rate (in that order). Alternatively, another output from the `fitgene()` function (for example from a gene that has a similar expression profile) can be supplied as a `firstguess` argument.

The `criterion` argument is only used if the training object fed through the tHVDM command is a non-linear fit and determines the type of criterion used for model selection between Michelis-Menten and Hill. Possible values fed throught this argument are "BIC" (Bayesian information criterion, default) and "AIC" (Akaike information critertion). This argument is ignored in case of linear fitting.

Value

a list containing the results (see documentation for more details).

Note

Obviously, the expression set given as a `eset` argument has to be the same as the one used for the training step.

Author(s)

Martino Barenco

References

M. Barenco, D. Tomescu, D. Brewer, R. Callard, J. Stark, M. Hubank (2006) Ranked predictions of p53 targets using Hidden Variable Dynamic Modelling. *Genome Biology*, **V7(3)**, R25.

See Also

`training`,`screening`,`HVDMreport`

Examples

```
data(HVDMexample)
tHVDMp53<-training(eset=fiveGyMAS5,genes=p53traingenes,degrate=0.8,actname="p53")
sHVDMcd38<-fitgene(eset=fiveGyMAS5,gene="205692_s_at",tHVDM=tHVDMp53)
```

`fiveGyMAS5` — *Example data for the rHVDM package (time course of irradiated human T cells)*

Description

fiveGyMAS5: the data set proper is an Expression set (Biobase). MOLT4 p53-wild type human T-cells irradiated @ 5 Gy, 7 time points. Triplicate experiment.

Format

R data (.rda)

`genestoscreen` — *Genes differentially expressed*

Description

A vector containing affymetrix identifiers of differentially expressed genes. Part of an an example dataset for the rHVDM package.

Format

R data (.rda)

`HVDMcheck` *Checks that an object of class ExpressionSet is compliant with HVDM*

Description

This method issues warnings about the compliance of the ExpressionSet object to be used with rHVDM. It just issues warnings and does not attempt to correct the problems encountered. the `pdata` argument is optional; if missing, the method will check the phenodata that is inside the ExpressionSet.

Usage

```
HVDMcheck(eset,pdata)
```

Arguments

`eset`	an ExpressionSet object (Biobase)
`pdata`	a dataframe (optional argument)

Details

The checks performed are: A) the phenoData has all the required fields. B) the time field is numeric, has a zero time point, does not have negative or repeated values. C) the row names in the phenoData are consistent with the columns names found in the ExpressionSet. D) the data are not log-transformed. E) Standard deviations have been supplied in the ExpressionSet, and they are non-negative.

Value

no value returned.

Note

If the `pdata` argument is ignored (it can be), the method checks the phenoData that is inside the ExpressionSet.

Author(s)

Martino Barenco

References

M. Barenco, D. Tomescu, D. Brewer, R. Callard, J. Stark, M. Hubank (2006) Ranked predictions of p53 targets using Hidden Variable Dynamic Modelling. *Genome Biology*, **V7(3)**, R25.

See Also

`training`

Examples

```
data(HVDMexample)
HVDMcheck(fiveGyMAS5)
```

HVDMreport *generates an HTML report using the input*

Description

This method generates report using one of the HVDM outputs (either from the `training()`, `screeening()` or `fitgene()` commands.)

Usage

```
HVDMreport(HVDM,name)
```

Arguments

`HVDM`	a list output either by a `training()`, `screening()` or `fitgene()` command.
`name`	an optional argument allowing to specify the name of the HTML file (the .HTML extension is added automatically).

Details

If the output has been generated by `training()` or `fitgene()`, the HTML report contains four sections:

1) Functions inputs.

2) Model score.

3) Fitted parameters.

4) Visual comparison of model and data.

In the `fitgene()` case comparisons with correponding results for genes in the training set are given.

If the output has been generated by the `screening()` command, the HTML report contains the following sections:

1) Inputs to the preceding `training()` command.

2) Inputs to the `screening()` command (such as bounds used to classify genes).

3) Results.

4) List of putative targets of the transcription factor under review.

The HTML report is genereated in the current working directory. Graphic files are stored in a subdirectory with a similar name.

Value

Nothing is returned, but a message indicating the name of the HTML file and its location on the hard drive is given.

Note

The HTML report is generated in the working directory.

Author(s)

Martino Barenco

References

M. Barenco, D. Tomescu, D. Brewer, R. Callard, J. Stark, M. Hubank (2006) Ranked predictions of p53 targets using Hidden Variable Dynamic Modelling. *Genome Biology*, **V7(3)**, R25.

See Also

`training`,`screening`,`fitgene`

Examples

```
data(HVDMexample)
tHVDMp53<-training(eset=fiveGyMAS5,genes=p53traingenes,degrate=0.8,actname="p53")
#HVDMreport(tHVDMp53)
#HVDMreport(tHVDMp53, name="p53training")
```

---

leastnoisychips *Identifiers of the three least noisy arrays in the p53 experiment*

Description

Identifiers of the three least noisy arrays in the p53 experiment

Format

R data (.rda)

---

p53traingenes *Known p53 targets*

Description

A vector with five p53 targets (affymetrix identifiers). Part of an an example dataset for the rHVDM package.

Format

R data (.rda)

rHVDMplattforms *plattform-specific repository allowing to estimate measurement errors*

Description

plattform-specific repository allowing to estimate measurement errors, plattforms covered: 1 - Affymetrix Human genome 133A 3-prime array

Format

R data (.rda)

screening.nl *Fits the optimal kinetic parameter values for several genes with a non-linear model.*

Description

This method fits the kinetic parameter values for each gene in a user-supplied vector. It returns a list containing the results.

Usage

```
screening.nl(eset,genes,HVDM,transforms,cl1zscorelow,cl1modelscorehigh,cl1degrat
```

Arguments

eset	an ExpressionSet object (Biobase)
genes	a vector containing the genes identifiers to be screened (in character format)
transforms	a vector containing the kinetic parameter identifiers that have to be transformed during optimisation (optional)
HVDM	a HVDM object (see details)
cl1zscorelow	the Vj Z-score cutoff value for a gene to be classified as a putative target
cl1modelscorehigh	the model score cutoff value for a gene to be classified as a putative target
cl1degraterange	the degradation rate bounds applied for a gene to be classified as a putative target
criterion	criterion used to select the model. "BIC" (bayesian, default) or "AIC" (akaike)

Details

This screening step can only be applied if a `training.nl()` step has already been run. The output to the `training,nl()` step can be given as the "HVDM" argument. A fit of each gene in the "genes" argument is then performed and a model is selected according to the input to the `criterion` argument.

Alternatively an output to a previously run `screening()` step can be supplied as an "HVDM" argument. In this case, the fit is not performed once again. Feeding a previous output of `screening()` to that same function again is useful only if the various bounds altered in the secon run. In the case this option is used, the "eset" and "genes" arguments do not have to be supplied (they will be ignored anyway).

The output of that function is a list containing the results. The relevant data frame is in the "results" member of the output. Putative targets can be identified using the "class1" field of that data frame (see example).

Bounds determining whether a gene is a target of the transcription factor under review have to be supplied. They are:

- cl1zscorelow: lower bound for the Vj Z-score (default: 2.5)

- cl1modelscore: upper bound for the model score (default: 100.0). This default will have to be changed in most cases. As a rule of thumb, 5x the model score for the genes in the training set can be used.

- cl1degraterange: lower and upper bounds for the degradation rate (default: c(0.01,5.0)). This is to exclude those genes with an absurd degradation rate, measured in (unit time)\textasciicircum(-1). In our example the unit time is an hour. In the case the unit time is different, these default bounds will have to be altered accordingly.

An exponential transform is set by default for both the basal (Bj) and degradation (Dj) rates as well as the kinetic parameters (Vj and Kj) of the production function(through the transforms argument). This forces the values for these parameters to be positive. It also helps to reach a better fit. To turn this off let transforms=c(). Even in this case the degradation rate will not be allowed to take non positive values as it causes problems with the differential operator used internally. The value in the vector indicates the parameter to be transformed: "Bj": basal rate of transcription, "Dj": degradation rate. The entry label indicates the transform to be applied; presently,

The argument `criterion` specifies which criterion should be used for model selection.

Value

a list containing the results.

Note

Obviously, the expression set given as a `eset` argument has to be the same as the one used for the training set.

Author(s)

Martino Barenco

References

M. Barenco, D. Tomescu, D. Brewer, R. Callard, J. Stark, M. Hubank (2006) Ranked predictions of p53 targets using Hidden Variable Dynamic Modelling. *Genome Biology*, **V7(3)**, R25.

See Also

training,HVDMreport,fitgene,training,fitgene.nl

Examples

```
#load data and fit a linear model
data(HVDMexample)
rm(fiveGyMAS5)
data(HVDMexample2)
tp532<-training(eset=twodosesMAS5,genes=p53traingenes,degrate=0.8)
#formulate constraints
CONSTRAINTS<-c(350,35)
names(CONSTRAINTS)<-c("trfact1.5Gy.1.4","trfact1.hGy.1.4")
#specify individual gene models
GENEMODELS<-rep("MM",5)
GENEMODELS[c(1,3)]<-"hill"
names(GENEMODELS)<-p53traingenes
#run the model
tp53hyb<-training.nl(inputHVDM=tp532,constraints=CONSTRAINTS,genemodels=GENEMODELS)
#launch screening
screenlist<-screening.nl(eset=twodosesMAS5,genes=genestoscreen[c(1:5)],HVDM=tp53hyb)
```

---

screening *Fits the optimal kinetic parameter values for several genes.*

Description

This method fits the three kinetic parameter values for each gene in a user-supplied vector. It returns a list containing the results.

Usage

```
screening(eset,genes,HVDM,transforms,cl1zscorelow,cl1modelscorehigh,cl1degratera
```

Arguments

eset	an ExpressionSet object (Biobase)
genes	a vector containing the genes identifiers to be screened (in character format)
transforms	a vector containing the kintetic parameter identifiers that have to be transformed during optimisation (optional)
HVDM	a HVDM object (see details)
cl1zscorelow	the sensitivity Z-score cutoff value for a gene to be classified as a putative target
cl1modelscorehigh	the model score cutoff value for a gene to be classified as a putative target
cl1degraterange	the degradation rate bounds applied for a gene to be classified as a putative target

Details

This screening step can only be applied if a `training()` step has already been run. The output to the `training()` step can be given as the "HVDM" argument. A fit of each gene in the "genes" argument is then performed.

Alternatively an output to a previously run `screening()` step can be supplied as an "HVDM" argument. In this case, the fit is not performed once again. Feeding a previous output of `screening()` to that same function again is useful only if the various bounds altered in the secon run. In the case this option is used, the "eset" and "genes" arguments do not have to be supplied (they will be ignored anyway).

The output of that function is a list containing the results. The relevant data frame is in the "results" member of the output. Putative targets can be identified using the "class1" field of that data frame (see example).

Bounds determining whether a gene is a target of the transcription factor under review have to be supplied. They are:

- cl1zscorelow: lower bound for the sensitivity Z-score (default: 2.5)

- cl1modelscore: upper bound for the model score (default: 100.0). This default will have to be changed in most cases. As a rule of thumb, 5x the model score for the genes in the training set can be used.

- cl1degraterange: lower and upper bounds for the degradation rate (default: c(0.05,5.0)). This is to exclude those genes with an absurd degradation rate, measured in (unit time)\textasciicircum(-1). In our example the unit time is an hour. In the case the unit time is different, these default bounds will have to be altered accordingly.

An exponential transform is set by default for both the basal (Bj) and degradation (Dj) rates (through the transforms argument). This forces the values for both these parameters to be positive. It also helps to reach a better fit. To turn this off let transforms=c(). Even in this case the degradation rate will not be allowed to take non positive values as it causes problems with the differential operator used internally. The value in the vector indicates the parameter to be transformed: "Bj": basal rate of transcription, "Sj": sensitivity, "Dj": degradation rate. The entry label indicates the transform to be applied; presently, only log-tranforms are implemented (ie "exp").

Value

a list containing the results (see documentation for more details).

Note

Obviously, the expression set given as a `eset` argument has to be the same as the one used for the training set.

Author(s)

Martino Barenco

References

M. Barenco, D. Tomescu, D. Brewer, R. Callard, J. Stark, M. Hubank (2006) Ranked predictions of p53 targets using Hidden Variable Dynamic Modelling. *Genome Biology*, **V7(3)**, R25.

See Also

`training`,`HVDMreport`,`fitgene`

Examples

```
data(HVDMexample)
tHVDMp53<-training(eset=fiveGyMAS5,genes=p53traingenes,degrate=0.8,actname="p53")
screenp53<-screening(eset=fiveGyMAS5,genes=genestoscreen[1:10],HVDM=tHVDMp53)

#extracting a list of putative p53 targets
p53targets<-screenp53$results[screenp53$results$class1,]

#shifting the goal posts
screenp53B<-screening(HVDM=screenp53,cl1zscorelow=3.5)
```

---

training.nl *Performs the HVDM training step and returns a list containing the results*

Description

This method performs the nonlinear training step of the HVDM algorithm. It returns a list that will then be used in the subsequent screening step.

Usage

```
training.nl(inputHVDM,transforms,constraints,forcetransforms,genemodels,firstgue
```

Arguments

`inputHVDM`	an HVDM training object
`transforms`	a vector containing the kinetic parameter identifiers that have to be transformed during optimisation (optional)
`constraints`	"known" values for the activator signal
`forcetransforms`	Boolean, whether the transformation in argument transforms have to be applied
`genemodels`	the type of model that has to be used for each gene
`firstguess`	a first guess for all the parameters

Details

Contrary to the linear training function (without .nl suffix), this function takes as main input another training object (either a linear training object or a non-linear one, where all the genes have been fitted witha MIchelis-Menten model ("MM").

An exponential transform is set by default for the basal (Bj) and degradation (Dj) rates, as well as for the kinetic parameters (Vj and Kj) in the production function (through the transforms argument). This forces the values for these parameters to be positive For the exponent Nj, the `expp1` function ensures it is greater than one, when the hill formulation is used. To turn this off, set the `forcetransforms` switch to FALSE. Even in this case the degradation rate will not be allowed to take non-positive values as it causes problems with the differential operator used internally. The value in the vector indicates the parameter to be transformed: "Bj": basal rate of transcription, "Sj": sensitivity, "Dj": degrdation rate, etc.. The entry label indicates the transform to be applied.

The `constraints` argument is used to specify values for know values of some of the parameters, typically the strength of the activator for some time points. See example below for the syntax.

The genemodels argument is compulsory and used to specify the model used for each individual gene. MM means Michaelis-Menten model whereas hill means a hill function is used. The general form of the production function is Bj+Vj*f(t)^Nj/(Kj^Nj+f(t)^Nj) where Nj=1 for the MM model and Nj>1 in the hill case. See below for the syntax of the input.

The firstguess argument is not in use yet and will offer the possibility to enter a first guess for the fitting.

Value

a list containing the results.

Author(s)

Martino Barenco

References

M. Barenco, D. Tomescu, D. Brewer, R. Callard, J. Stark, M. Hubank (2006) Ranked predictions of p53 targets using Hidden Variable Dynamic Modelling. *Genome Biology*, **V7(3)**, R25.

See Also

HVDMcheck,screening.nl,fitgene,HVDMreport,training

Examples

```
#load data and fit a linear model
data(HVDMexample)
rm(fiveGyMAS5)
data(HVDMexample2)
tp532<-training(eset=twodosesMAS5,genes=p53traingenes,degrate=0.8)
#formulate constraints
CONSTRAINTS<-c(350,35)
names(CONSTRAINTS)<-c("trfact1.5Gy.1.4","trfact1.hGy.1.4")
#specify individual gene models
GENEMODELS<-rep("MM",5)
GENEMODELS[c(1,3)]<-"hill"
names(GENEMODELS)<-p53traingenes
#run the model
tp53hyb<-training.nl(inputHVDM=tp532,constraints=CONSTRAINTS,genemodels=GENEMODELS)
```

---

training *Performs the HVDM training step and returns a list containing the results*

Description

This method performs the training step of the HVDM algorithm. It returns a list that will then be used in the subsequent screening step.

Usage

```
training(eset,genes,transforms,degrate,actname,pdata,forcetransforms)
```

Arguments

eset	an ExpressionSet object (Biobase)
genes	a vector containg the gene identifiers of the training genes
transforms	a vector containing the kintetic parameter identifiers that have to be transformed during optimisation (optional)
degrate	value of the anchoring gene degradation rate, expressed in inverse unit time (optional)
pdata	a data frame, phenoData to be used for the training (optional)
actname	name of the transcription factor of interest (optional)
forcetransforms	Boolean, whether the transformation in argument transforms have to be applied

Details

The first entry in the genes vector is the anchoring gene. This means that the sensitivity (Sj) for this genes is set at 1.0 by default and that if a degradation rate is supplied it applies to that gene.

An exponential transform is set by default for both the basal (Bj) and degradation (Dj) rates (through the transforms argument). This forces the values for both these parameters to be positive. It also helps to reach a better fit. To turn this off, set the `forcetransforms` switch to FALSE. Even in this case the degradation rate will not be allowed to take non-positive values as it causes problems with the differential operator used internally. The value in the vector indicates the parameter to be transformed: "Bj": basal rate of transcription, "Sj": sensitivity, "Dj": degrdation rate. The entry label indicates the transform to be applied; presently, only log-tranforms are implemented (ie "exp").

The `degrate` argument is optional, but it is recommended to provide the algorithm with an externally measured degradation rate, as this greatly improves the accuracy and robustness of the outcome.

The `pdata` argument is also optional. By default the method will use the phenoData contained in the expression set. This argument can be used for excluding a time point, or an entire replicate. To extract the phenoData from the expression set, use `dataframe<-pData(eset)`. The `dataframe` object obtained can then be manipulated as desired.

The default name of the transcription factor is "trfact1".

Value

a list containing the results (see documentation for more details).

Note

It is recommended to run the HVDMcheck method before running this command.

Author(s)

Martino Barenco

References

M. Barenco, D. Tomescu, D. Brewer, R. Callard, J. Stark, M. Hubank (2006) Ranked predictions of p53 targets using Hidden Variable Dynamic Modelling. *Genome Biology*, **V7(3)**, R25.

See Also

HVDMcheck,screening,fitgene,HVDMreport

Examples

```
data(HVDMexample)
tHVDMp53<-training(eset=fiveGyMAS5,genes=p53traingenes,degrate=0.8,actname="p53")
```

`twodosesMAS5` *Example data for the rHVDM package (time course of irradiated human T cells, two different doses)*

Description

fiveGyMAS5: the data set proper is an Expression set (Biobase). MOLT4 p53-wild type human T-cells irradiated @ 5 Gy, 7 time points. Triplicate experiment. Another single time course, same cells and conditions @ 0.5 Gy, with 2 extra points (16 and 20 hours after irradiation)

Format

R data (.rda)

Index