Package ‘designGG’

February 19, 2015

Version 1.1

Date 19-02-2013

Title Computational tool for designing genetical genomics experiments.

Author Yang Li <yang.li@rug.nl>, Morris Swertz <m.a.swertz@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>, Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

Maintainer Yang Li <yang.li@rug.nl>

Description The package provides R scripts for designing genetical genomics experiments.

Depends R (>= 2.2.0)

License GPL

URL http://www.rug.nl/research/bioinformatics/

NeedsCompilation no

Repository CRAN

Date/Publication 2013-02-19 16:08:27

R topics documented:

`acceptanceProbability` *Compute the acceptance probability for each updated design*

Description

Compute the acceptance probability for each updated design. It depends on the current temperature value of simulated annealing process. This is a subfunction needed for `designGG`, but is not directly used.

Usage

```
acceptanceProbability( designScore, newDesignScore, method,
                       temperature )
```

Arguments

`designScore`	score of current design.
`newDesignScore`	score of updated design.
`method`	either "SA" (simulated annealing) or "MH". (Metropolis Hastings)
`temperature`	current temperature in simulated annealing process.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

E. Wit and J. McClure. Statistics for Microarrays: Design, Analysis and Inference. (2004) Chichester: Wiley.
Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009) http://gbic.biol.rug.nl/designGG

See Also

designGG

---

arrayUpdate *Update array allocation*

Description

Update the allocation of samples on the arrays. This is a subfunction needed for `updateDesign`, but is not directly used.

Usage

```
arrayUpdate(array.allocation, condition.allocation, nRILs, nSlides)
```

Arguments

`array.allocation`	matrix with nArray rows and nRIL columns. Elements of 1/0 indicate this RIL (or strain) is/not selected for this array.
`condition.allocation`	matrix with nCondition rows and nRIL columns. Elements of 1/0 indicate this RIL (or strain) is/not selected for this condition.
`nRILs`	number of RILs or strains available for the experiment.
`nSlides`	total number of slides available for experiment.

Details

This function is used only for designing a dual-channel experiment where samples need to be paired.

Value

A list with the following two elements:
`new.array.allocation`: an updated array allocation table
`new.condition.allocation`: an updated condition allocation table

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)
http://gbic.biol.rug.nl/designGG

See Also

updateDesign

---

conditionAllocation *Allocate the selected RILs into different conditions*

Description

This is a subfunction used by `initialDesign` but is not directly used. In the experiment where samples are profiled in pairs, the samples are firstly selected and paired on each array and then the selected samples are randomly allocated into different conditions.

Usage

```
conditionAllocation( selectedRILs, genotype, nConditions, nSlides, nTuple )
```

Arguments

`selectedRILs`	the index of the selected RILs or strains among all that are available for the experiment.
`genotype`	genotype data: a nMarker-by-nRILs matrix with two allels being 0 and 1 (or A and B) or three allels being 0, 0.5 and 1 (or, A, H, and B), where 0.5 (or H) represents heterozygous allele.
`nConditions`	number of all possible combination of all environmental factors. It should be larger than 1.
`nSlides`	total number of slides available for the experiment. It should be a non-zero integer.
`nTuple`	average number of RILs to be assigned onto each condition. `nTuple` should be a real number which is larger than 1. if `nTuple` < 1, the algorithm will stop and show a message as below, `warning: "The number of slides is too small to perform the experiment."`

Details

This function is only called by `initialDesign` function when `btwoColorArray` is `TRUE`.

Value

A matrix with nCondition rows and nRIL columns. Elements of 1/0 indicate that this RIL (or strain) is/not selected for this condition.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009) http://gbic.biol.rug.nl/designGG

See Also

`initialDesign`

`conditionCombination` *Generate a matrix indicating all possible levels for environmental factors*

Description

Generate a matrix indicating all possible levels for environmental factors with dimension nConditions * nEnvFactors. This is a subfunction needed for `designScore`, but is not directly used.

Usage

```
conditionCombination( nEnvFactors, nLevels, Level, envFactorNames )
```

Arguments

`nEnvFactors`	number of environmental factors, an integer bewteen 1 and 3. When `nEnvFactors` is 1 and the number of levels for the enviromental factor (`nLevels`) is 1, there is one condition in the experiment (i.e. no enviromental perturbation) and thus only genetic factor will be considered in the algorithm. When `nEnvFactors` is 1 and nLevels is larger than 1 or `nEnvFactors` is larger than 1, all main factor(s) and interacting facotr(s) will be included. Examples: If there is a temperature perturbation, then `nEnvFactors` is 1; If there is both temperature and drug treatment perturbation, then `nEnvFactors` is 2.
`nLevels`	number of levels for each factor, a vector with each component being integer. The length should be equal to `nEnvFactors`.
`Level`	a list which specifies the levels for each factor in the experiment. There are in total `nEnvFactors` elements in the list and each element correspsonds to certain envrironmental factor. The element is a vector describing all levels of the environmental factor. Default setting for the level of each factor is 1, 2, ..., nLevels[i]. (Here nLevels[i] is the *i*th element of nLevels, which tells the total number of levels for *i* environmental factor).
`envFactorNames`	a vector with names for all environmental factor(s). For example, for an experiment with two environmental factors of temperature and drug treatment: `envFactorNames <- c( "Temperature", "Dosage" )` Default = `NULL`, then the output will use `"F1"` and `"F2"` to indicate the environmental factors.

Details

Currently this function works only when `nEnvFactors` is between 1 and 3.

Value

A matrix with dimension of nConditions * nEnvFactors. Each element in the matrix indicates the levels of corresponding environmental factor.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)
http://gbic.biol.rug.nl/designGG

See Also

`designScore`

`conditionLevel` *Levels of all environmental factors*

Description

Describe the levels of all environmental factors for each RIL/strain in the experiment.
This is a subfunction needed for `designScore`, but is not directly used.

Usage

```
conditionLevel( array.allocation, condition.allocation,
                condition.combination,nEnvFactors )
```

Arguments

`array.allocation`	a matrix with nArray rows and nRIL columns. Elements of 1/0 indicates this RIL (or strain) is/not selected for this array.
`condition.allocation`	a matrix with nCondition rows and nRIL columns. Elements of 1/0 indicates this RIL (or strain) is/not selected for this condition.

`condition.combination`	a matrix indicating all possible levels for environmental factors, with dimension of nConditions by nEnvFactors.
`nEnvFactors`	number of environmental factors, an integer bewteen 1 and 3. When `nEnvFactors` is 1 and nLevels is 1, there is one condition in the experiment (i.e. no enviromental perturbation) and thus only genetic factor will be considered in the algorithm. When `nEnvFactors` is 1 and nLevels is larger than 1 or `nEnvFactors` is larger than 1, all main factor(s) and interacting facotr(s) will be included.

Details

For single-channel experiment, `array.allocation` is `NULL`. Then the `conditionLevel` is decided by `condition.allocation`. For dual-channel experiment, `array.allocation` decides which RILs are selected and then the `condition.allocation` indicates which condition this RIL will be put in for the experiment.

Value

A matrix with dimension of nRILs by nEnvFactors, each element indicates the level of a certain environmental factor to which the RIL (or strain) is exposed in the experiment.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)
http://gbic.biol.rug.nl/designGG

See Also

`designScore`, `conditionCombination`

`conditionUpdate` *Update condition allocation*

Description

Update the allocation of samples onto different conditions. This is a subfunction needed for `updateDesign`, but is not directly used.

Usage

```
conditionUpdate( condition.allocation, nTuple,bTwoColorArray )
```

Arguments

`condition.allocation`	a matrix with nCondition rows and nRIL columns. elements of 1/0 indicate this RIL (or strain) is/not selected for this condition.
`nTuple`	average number of RILs (or strains) to be assigned onto each condition `nTuple` should be a real number which is larger than 1. if `nTuple` < 1, the algorithm will stop and show the message, `warning: "The number slides is too less to perform the experiment."`
`bTwoColorArray`	binary variable indicating experiment type: `bTwoColorArray <- TRUE` \#for dual channel experiment `bTwoColorArray <- FALSE` \#for single channel experiment

Details

This function will be used both in single and dual channel experiment design.

Value

An updated `condition.allocation` table.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)
http://gbic.biol.rug.nl/designGG

See Also

`arrayUpdate`, `designGG`

designGG *Optimal design for genetical genomics experiments*

Description

Main function to search and display A- and D- optimal designs for single- or two-channel genetical genomics experiments. Simulated annealing or Metropolis Hastings used to find the best design.

Usage

```
designGG ( genotype, nSlides, nTuple, nEnvFactors, nLevels,
                 Level=NULL, bTwoColorArray=TRUE, initial=NULL, weight=1,
                region=NULL, optimality="A", method="SA", nIterations=3000,
                 n.search=2, endTemp=1e-10, startTemp=1, maxTempStep=0.9,
                 plotScores=TRUE, directory=NULL, fileName=NULL,
                 envFactorNames=NULL, writingProcess=TRUE )
```

Arguments

`genotype`	genotype data: a nMarker-by-nRILs matrix with two alleles being 0 and 1 (or A and B) or three allels being 0, 0.5 and 1 (or, A, H, and B), where 0.5 (or H) represents heterozygous allele.
`nSlides`	total number of slides available for the experiment.
`nTuple`	average number of RILs (or strains) to be assigned onto each condition. `nTuple` should be a real number which is larger than 1. If `nTuple` < 1, the algorithm will stop and show the message, `warning: "The number of slides is too small to perform the experiment."`
`nEnvFactors`	number of environmental factors, an integer bewteen 1 and 3. When `nEnvFactors` is 1 and the number of levels for the enviromental factor (`nLevels`)is 1, there is one condition in the experiment (i.e. no enviromental perturbation) and thus only genetic factor will be considered in the algorithm. When `nEnvFactors` is 1 and nLevels is larger than 1 or `nEnvFactors` is larger than 1, all main factor(s) and interacting facotr(s) will be included. Examples: If there is a temperature perturbation, then `nEnvFactors` is 1; If there is both temperature and drug treatment perturbation, then `nEnvFactors` is 2.
`nLevels`	number of levels for each factor, a vector with each component being an integer. The length of it should equal `nEnvFactors`.
`Level`	a list which specifies the levels for each factor in the experiment. There are in total `nEnvFactors` elements in the list and each element corresponds to a certain environmental factor. The element is a vector describing all levels of the environmental factor. default setting for the level of each factor is 1, 2, ..., nlevels[i]. (Here nLevels[i] is the *i*th element of nLevels, which gives the total number of levels for *i* environmental factor).
`bTwoColorArray`	binary variable indicating experiment type: `bTwoColorArray <- TRUE` \#for dual channel experiment `bTwoColorArray <- FALSE` \#for single channel experiment

`initial`	the starting design matrix for the algorithm. If specified, this should be a list with 2 matrices: `condition.allocation`: allocate RILs (or strains) into different conditional (nrow = nCondition, ncol= nRILs) `array.allocation`: pair RILs (or strains) into sldies (nrow = nSlide, ncol = nRILs) However, the algorithm does not require that a starting matrix is specified. Default = `NULL`.
`weight`	a vector with length of `variableNumber` which is calculated from function `variableNumber`. Default = 1 (which means the parameters to be estimated are all equally important during optimization). See details below.
`region`	genome region of biological interest. Default = `NULL` (which means the entire genome will considered).
`optimality`	type of optimality, i.e. "A" (A-optimality) or "D" (D-optimality). A-optimality minimizes $Trace((X’X)^-1)$, which corresponds to minimum average variance of the parameter estimates. D-optimality minimizes $det(X’X)^-1$, which corresponds to minimum generalized variance of the parameter estimates.
`method`	method for searching for an optimal design. "SA" uses simulated annealing. "MH" uses Metropolis Hasting. Default = "SA".
`nIterations`	number of iterations of the simulated annealing method. Default = 3000.
`n.search`	number of times for simulated annealing optimaization with different initial design, default = 2. Here it is suggested to be between 1 and 5. It should not to be too large because of the reaching computational burden.
`endTemp`	ending temperature of simulated annealing process. An important optimization parameter. Default = $1e^-10$.
`startTemp`	starting temperature of simulated annealing process. Default = 1.
`maxTempStep`	maximum temperature decreasing step for simulated annealing process. The parameter ensures that the multiplicative cooling factor is not smaller than that. If `nIterations` is too small, the preferred final temperature (endTemp) may not be reached. See Wit and McClure (2004) for details. Default = 0.9.
`plotScores`	If `TRUE` (default) it produces a plot of the optimazation by SA using the function plotAllScores.
`directory`	It tells where the resulting optimal design tables are to be stored. If `NULL` (default), it will take currect working directory.
`fileName`	the final optimal design table(s) in `csv` format and a plot (in png format) of all scores during SA process (if `plotScores` = TRUE) will be produced. The users can specify the table and plot name by setting `fileName`. If `NULL` (default) it produces files starting with `"myDesignGG"`.
`envFactorNames`	a vector with names for all environmental factor(s). For example, for the experiment with two environmental factors of temperature and drug treatment: `envFactorNames <- c( "Temperature", "Dosage" )` Default = `NULL`, then the output will use `"F1"` and `"F2"` to indicate the environmental factors.
`writingProcess`	If TRUE, it prints how much computation work has been finished in a file called `"processing.txt"`. Default = `TRUE`.

Details

Given the genetic information of samples available for the experiment (genotype) and the information about experimental settings (`nEnvFactors`, `nSlides`,`nLevels` etc.), the algorithm searches for an A-optimal or D-optimal (see `optimality`) using simulated annealing (see `method`). A plot of the scores at each iterations can also be given using the `plotAllScores` function.
It also contains a number of the arguments:
`region` is used to specify the genome region that are of major interest to experimenters.
`weight` is used to define the weight of genetic and environmental factors, and interaction terms. Prior knowledge about expected effect sizes of interesting factors can also be incorporated as `weight` parameters for the algorithm. The weight is inversely proportional to the expected effect size of the corresponding parameter. Example parameter settings: Suppose to design an experiment with two environmental factors (F1, F2) and there are two diffferent levels for each environment. The levels are 16 and 24 for F1, and 5 and 10 for F2. Thus the following command can be used:

```
nEnvFactors  <- 2
nLevels       <- c ( 2, 2 )
levels       <- list ( c(16, 24), c(5, 10) )
```

The length of parameter `weight` is dependent on the number of environmental factors:
When `nEnvFactor` = 0,
`weight` is 1 as there is only one parameter of interest (genotype).
When `nEnvFactor` = 1,
`weight` = c(\$w_Q\$, \$w_F1\$, \$w_QF1\$)
When `nEnvFactor` = 2,
`weight` = c(\$w_Q\$, \$w_F1\$, \$w_F2\$, \$w_QF1\$, \$w_QF2\$, \$w_F1F2\$, \$w_QF1F2\$)
When `nEnvFactor` = 3,
`weight` = c(\$w_Q\$, \$w_F1\$, \$w_F2\$, \$w_F2\$, \$w_QF1\$, \$w_QF2\$, \$w_QF3\$, \$w_F1F2\$, \$w_F1F3\$, \$w_F2F3\$, \$w_QF1F2\$, \$w_QF1F3\$, \$w_QF2F3\$, \$w_QF1F2F3\$)
Here \$w_Q\$ represents the weight for genotype effect, \$w_F1\$ represent the weight for F1 effect and \$w_QF1\$ represent the weight for interaction between genotype and F1 effect, etc.
It should be noted that the simulated annealing algorithm might find a locally and not globally optimal design. Running the optimization process multiple times is recommended. When `nSearch` > 1, the simulated annealing optimization will be run nSearch times, each run starts with a different initial design and will provide a (near-)optimal design. If the optimization problem is simple, all runs will converge to the same (optimal) design. Otherwise, the best one among all near-optimal designs will be selected as the output of the function. One can run the algorithm multiple times with `nSearch` = 1 to review a few (near-)optimal designs.

Value

An array design table (arrayDesign.csv) and a condition design table (conditionDesign.csv) are generated.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009) http://gbic.biol.rug.nl/designGG
Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
E. Wit and J. McClure. Statistics for Microarrays: Design, Analysis and Inference. (2004) Chichester: Wiley.

See Also

`initialDesign`, `designScore`, `updateDesign`, `acceptanceProbability`, `experimentDesignTable`, `plotAllScores`, `exampleArrayDesignTable`,`exampleConditionDesignTable`,

Examples

```
library(designGG)
#load genotype data
data(genotype)
#Example:  single-channel experiment with 2 environmental factors,
#each with 2 levels, and there will be four samples per condition(nTuple=4).
optimalDesign <- designGG ( genotype, nSlides=NULL, nTuple=4, nEnvFactors=2,
                   nLevels=c(2,2),Level=list(c(16,24),c(5,10)),  bTwoColorArray=FALSE,
                      initial=NULL, weight=1, region=seq(1,20), optimality="A",
                      method="SA", nIterations=100, n.search=2, endTemp=1e-10,
                      startTemp=1, maxTempStep=0.9, plotScores=TRUE,
                      directory=NULL, fileName=NULL, envFactorNames=NULL,
                      writingProcess=FALSE )
#Example 2:  dual-channel experiment with 2 environmental factors,
#each with 2 levels. There are 50 slides available.
optimalDesign <- designGG ( genotype, nSlides=50, nTuple=NULL, nEnvFactors=2,
                    nLevels=c(2,2),Level=list(c(16,24),c(5,10)),  bTwoColorArray=TRUE,
                      initial=NULL, weight=1, region=seq(1,20), optimality="A",
                      method="SA", nIterations=100, n.search=2, endTemp=1e-10,
                      startTemp=1, maxTempStep=0.9, plotScores=TRUE,
                      directory=NULL, fileName=NULL, envFactorNames=NULL,
                      writingProcess=FALSE )
#result
optimalDesign$arrayDesign
optimalDesign$conditionDesign
plotAllScores(optimalDesign$plot.obj)

#Use the following commands to see example output tables:
data(exampleArrayDesignTable)
exampleArrayDesignTable
data(exampleConditionDesignTable)
exampleConditionDesignTable
```

designScore *Calculate the A- or D- optimality score based on current experimental design*

Description

According to the current experimental design, the Fisher information matrix is obtained and then either the A- or D- optimality score is computed.

Usage

```
designScore( genotype, array.allocation, condition.allocation,
             nEnvFactors, nLevels, Level, nConditions, weight=1,
             optimality="A", bTwoColorArray, envFactorNames)
```

Arguments

`genotype`	genotype data: a nMarker-by-nRILs matrix with two allels being 0 and 1 (or A and B) or three allels being 0, 0.5 and 1 (or, A, H, and B), where 0.5 (or H) represents heterozygous allele.
`array.allocation`	matrix with nArray rows and nRIL columns. Elements of 1/0 indicate this RIL (or strains) is/not selected for this array.
`condition.allocation`	matrix with nCondition rows and nRIL columns. Elements of 1/0 indicate this RIL (or strains) is/not selected for this condition.
`nEnvFactors`	number of environmental factors, an integer bewteen 1 and 3. When `nEnvFactors` is 1 and the number of levels for the enviromental factor (`nLevels`)is 1, there is one condition in the experiment (i.e. no enviromental perturbation) and thus only genetic factor will be considered in the algorithm. When `nEnvFactors` is 1 and nLevels is larger than 1 or `nEnvFactors` is larger than 1, all main factor(s) and interacting facotr(s) will be included. Examples: If there is a temperature perturbation, then `nEnvFactors` is 1; If there is both temperature and drug treatment perturbation, then `nEnvFactors` is 2.
`nLevels`	number of levels for each factor, a vector with each component being an integer. The length of it should equal `nEnvFactors`.
`Level`	a list which specifies the levels for each factor in the experiment. There are in total `nEnvFactors` elements in the list and each element correpsond to certain envrironmental factor. The emlemet is a vector describing all levels of the environmental factor. default setting for the level of each factor is 1, 2, ... nLevels[i]. (Here nLevels[i] is the *i*th element of nLevels, which gives the total number of levels for *i* environmental factor).
`nConditions`	number of all possible combination of all environmental factors.
`weight`	a vector with length of variableNumber which is calculated from function `variableNumber`. Default = 1 (which means the parameters to be estimated are equally important during optimization.)

`optimality`	type of optimality, i.e. "A" (A-optimality) or "D" (D-optimality). A-optimality minimizes $Trace((X'X)^-1)$, which corresponds to minimum average variance of the parameter estimates. D-optimality minimizes $det(X'X)^-1$, which corresponds to minimum generalized variance of the parameter estimates.
`bTwoColorArray`	binary variable indicating experiment type: `bTwoColorArray <- TRUE` \#for dual channel experiment `bTwoColorArray <- FALSE` \#for single channel experiment
`envFactorNames`	a vector with names for all environmental factor(s). For example, for the experiment with two environmental factors of temperature and drug treatment: `envFactorNames <- c( "Temperature", "Dosage" )` Default = `NULL`, then the output will use "F1" and "F2" to indicate the environmental factors.

Details

Example parameter settings:
Suppose to design an experiment with two environmental factors (F1, F2) and there are two diff-ferent levels for each environment. The levels are 16 and 24 for F1, and 5 anf 10 for F2. Thus the following command can be used:

```
nEnvFactors <- 2
nLevels <- c ( 2, 2 )
levels <- list ( c(16, 24), c(5, 10) )
```

The length of parameter `weight` is dependent on the number of environmental factors:
When `nEnvFactor` = 0,
`weight` is 1 as there is only one parameter of interest (genotype).
When `nEnvFactor` = 1,
`weight` = c($w_Q$, $w_F1$, $w_QF1$)
When `nEnvFactor` = 2,
`weight` = c($w_Q$, $w_F1$, $w_F2$, $w_QF1$, $w_QF2$, $w_F1F2$, $w_QF1F2$)
When `nEnvFactor` = 3,
`weight` = c($w_Q$, $w_F1$, $w_F2$, $w_F2$, $w_QF1$, $w_QF2$, $w_QF3$, $w_F1F2$, $w_F1F3$, $w_F2F3$, $w_QF1F2$, $w_QF1F3$, $w_QF2F3$, $w_QF1F2F3$)
Here $w_Q$ represents the weight for genotype effect, $w_F1$ represent the weight for F1 effect and $w_QF1$ represent the weight for interaction between genotype and F1 effect, etc.

Value

The score is defined as the "double" sum of the variances, summed over all parameters and over all markers.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)

http://gbic.biol.rug.nl/designGG
Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
E. Wit and J. McClure. Statistics for Microarrays: Design, Analysis and Inference. (2004) Chichester: Wiley.

See Also

`designGG`

`exampleArrayDesignTable` *Example output of ArrayDesignTable data*

Description

`exampleArrayDesignTable`: Example data of `exampleArrayDesignTable` for a hypothetical dual-channel microarray experiment in which there are 100 strains (e.g. recombinant inbred lines) and 27 arrays available. Two environmental factors (temperature and cell type) are considered in this experiment. There are three levels for temperature (15, 24 and 29) and four levels for cell types (A,B,C,D). This table tells how to pair samples into arrays.

```
data(exampleArrayDesignTable)
exampleArrayDesignTable[1:5,]
```

	Channel 1	Channel 2
array1	Strain28	Strain92
array2	Strain70	Strain47
array3	Strain22	Strain89
array4	Strain45	Strain15
array5	Strain52	Strain41

Usage

```
data(exampleArrayDesignTable)
```

Format

`exampleArrayDesignTable`: 27 arrays by two channels.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

Examples

```
##load the data
data(exampleArrayDesignTable)

##view part of the the data
exampleArrayDesignTable[1:5,]
```

`exampleConditionDesignTable` *Example ConditionDesignTable data*

Description

`exampleConditionDesignTable`: Example data of `exampleConditionDesignTable` for a hypothetical dual-channel microarray experiment in which there are 100 strains (e.g. recombinant inbred lines) and 27 arrays available. Two environmental factors (temperature and cell type) are considered in this experiment. There are three levels for temperature (15, 24 and 29) and four levels for cell types (A, B, C, D). This table tells how to allocate samples into 12 (=3*4) different conditions. On average there are 4.5 (27*2/12) samples per condition.

```
> data(exampleConditionDesignTable)
> exampleConditionDesignTable[1:5,]
```

	Temperature	Cell Type			Selected Strains		
condition1	15	A	Strain28	Strain81	Strain18	Strain61	
condition2	24	A	Strain72	Strain40	Strain83	Strain44	Strain10
condition3	29	A	Strain22	Strain89	Strain3	Strain30	Strain58
condition4	15	B	Strain70	Strain47	Strain4	Strain59	
condition5	24	B	Strain93	Strain97	Strain49	Strain14	

Usage

```
data(exampleConditionDesignTable)
```

Format

`exampleConditionDesignTable`: 12 combination of conditions from three temepratures and four cell types.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

Examples

```
##load the data
data(exampleConditionDesignTable)

##view part of the the data
exampleConditionDesignTable[1:5,]
```

`examplePlotObj` *Example PlotObj data*

Description

`examplePlotObj`: Example data of `examplePlotObj` for plot all scores and cooling at each iteration during simulated annealing process.

```
data(examplePlotObj)
plotAllScores(examplePlotObj)
```

Usage

```
data(examplePlotObj)
```

Format

`examplePlotObj`: a list which contains the following elements: (1) scores (2) cooling (3) startTemp (4) temperature (5) temperature.step (6) nIterations (7) optimality.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

Examples

```
##load the data
data(examplePlotObj)

##make plot based on the data
plotAllScores(examplePlotObj)
```

experimentDesignTable *Make experiment table based design matrix*

Description

This function generates two `.csv` files which descibe how samples are allocated samples into different conditions and paired on arrays.

Usage

```
experimentDesignTable( array.allocation, condition.allocation,
                       nEnvFactors, nLevels, Level, fileName,envFactorNames,
                       directory )
```

Arguments

`array.allocation`	matrix with nArray rows and nRIL columns. Elements of 1/0 indicate this RIL is/not selected for this array.
`condition.allocation`	matrix with nCondition rows and nRIL columns. Elements of 1/0 indicate this RIL is/not selected for this condition.
`nEnvFactors`	number of environmental factors, an integer bewteen 1 and 3. When `nEnvFactors` is 1 and the number of levels for the enviromental factor (`nLevels`)is 1, there is one condition in the experiment (i.e. no enviromental perturbation) and thus only genetic factor will be considered in the algorithm. When `nEnvFactors` is 1 and nLevels is larger than 1 or `nEnvFactors` is larger than 1, all main factor(s) and interacting facotr(s) will be included. Examples: If there is a temperature perturbation, then `nEnvFactors` is 1; If there is both temperature and drug treatment perturbation, then `nEnvFactors` is 2.
`nLevels`	number of levels for each factor, a vector with each component being integer. The length of it should equal `nEnvFactors`.
`Level`	a list which specifies the levels for each factor in the experiment. There are in total `nEnvFactors` elements in the list and each element correpsond to certain envrironmental factor. The emlemet is a vector describing all levels of the environmental factor. default setting for the level of each factor is 1, 2, ...nLevels[i]. (Here nLevels[i] is the *i*th element of nLevels, which gives the total number of levels for *i* environmental facotor).
`fileName`	the final optimal design table(s) in `csv` format and a plot (in `png` format) of the all scores during SA process (if `plotScores` = T) will be produced. The users can specify the table and plot name by setting `fileName`. If `NULL` (default) it produces files starting with `"myDesignGG"`.
`envFactorNames`	a vector with names for all environmental factor(s). For example, for the experiment with two environmental factors of temperature and drug treatment: `envFactorNames <- c( "Temperature", "Dosage" )`

	Default = `NULL`, then the output will use "F1" and "F2" to indicate the environmental factors.
`directory`	It tells where the resulting optimal design tables are to be stored. If `NULL` (default), it will use the currect working directory.

Details

Based on `nEnvFactors` and `nLevels`, `nConditions` is calculated.

Value

Two tables report the results: table "pair design" which is only used for two-channel experiments and describes how samples are paired together on the slide (e.g. microarray chip), and table "environment design" which is used when there are more environments evolved in the experiment. With these two tables, the experimenters can set up the environmental treatment and follow-up profiling measurement.
Examples:
1. conditionDesign.csv

	Temperature	Cell Type			Selected Samples		
condition1	15	A	RIL28	RIL81	RIL18	RIL61	
condition2	24	A	RIL72	RIL40	RIL83	RIL44	RIL10
condition3	29	A	RIL22	RIL89	RIL3	RIL30	RIL58
condition4	15	B	RIL70	RIL47	RIL4	RIL59	
condition5	24	B	RIL93	RIL97	RIL49	RIL14	

2. arrayDesign.csv

	Channel 1	Channel 2
array1	RIL28	RIL92
array2	RIL70	RIL47
array3	RIL22	RIL89
array4	RIL45	RIL15
array5	RIL52	RIL41

Note

The optimal design results are described in two tables. One is called "array design" which is only used for two-channel experiments. It describes how samples are paired together on the slide (e.g. microarray chip). The other table is called "condition design" which is used when there is more than one environmental factor involved in the experiment. Each cell in condition design table represents a combination of different levels of environmental factors and the selected sample names (e.g. RIL names) for this condition are shown. Based on these two tables, the experimenters can set up the environmental treatment and follow-up profiling measurement.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)
http://gbic.biol.rug.nl/designGG

See Also

`designGG`, `exampleArrayDesignTable`, `exampleConditionDesignTable`

`genotype` *Example genotype data*

Description

`genotype`: example data of `genotypes` for each marker (rownames) and 100 strains such as recombinant inbred lines (RIL) (columnnames), with numeric values 1 and 0 (or A and B).

```
data(genotype)
genotype[1:5,1:5]
```

	Strain1	Strain2	Strain3	Strain4	Strain5
C1M1	1	0	0	0	1
C1M2	1	0	0	0	1
C1M3	1	0	0	0	1
C1M4	1	0	0	1	1
C1M5	1	0	0	1	1

Usage

```
data(genotype)
```

Format

`genotype`: 120 markers by 100 samples (Strains).

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

Examples

```
##load the data
data(genotype)

##view part of the the data
genotype[1:5,1:5]
```

`initialDesign` *Initialize an experiment design matrix*

Description

Allocate RILs (or strains) into different conditional and pair RILs (or strains) into slides.

Usage

```
initialDesign( genotype, nRILs, nSlides, nConditions, nTuple,
               bTwoColorArray )
```

Arguments

`genotype`	genotype data: a nMarker-by-nRILs matrix with two alleles being 0 and 1 (or A and B) or three alleles being 0, 0.5 and 1 (or, A, H, and B), where 0.5 (or H) represents heterozygous allele.
`nRILs`	total number of RILs ((or strains) available for the experiment.
`nSlides`	total number of slides available for the experiment.
`nConditions`	number of all possible combination of all environmental factors.
`nTuple`	average number of RILs (or strains) to be assigned onto each condition `nTuple` should be a real number which is larger than 1. if `nTuple` < 1, the algorithm will stop and shw the message below, `warning: "The number of slides is too small to perform the experiment."`
`bTwoColorArray`	binary variable indicating experiment type: `bTwoColorArray <- TRUE` \#for dual channel experiment `bTwoColorArray <- FALSE` \#for single channel experiment

Details

For two-color array experiments, randomly choose a RIL (or strain) and pair it with the genetically most different RIL (or strain) on one array.
For one-color array experiments, array.allocation is `NULL` as there is no need to pair samples.

Value

a list with 2 matrices:
`condition.allocation`: allocate RILs (or strains) into different conditional (nCondition * nRILs)
`array.allocation`: pair RILs (or strains) into sldies (nSlides * nRILs)

Note

This function calls `conditionAllocation` function to allocate selected RILs (or strains) into different conditions.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)
http://gbic.biol.rug.nl/designGG

See Also

`designGG`

Examples

```
library(designGG)
data(genotype)
nRILs <-100
nEnvFactors <- 2
nConditions <-2
nLevels <- c( 2, 2 )
levels <- list ( c(16, 24), c(5, 10) )
nSlides <- 100
nTuple <- 25
bTwoColorArray <- TRUE
initialDesign( genotype, nRILs, nSlides, nConditions, nTuple, bTwoColorArray )
```

`interactionLevel` *Generate levels for all interacting factors*

Description

Generate levels for all interacting factors for all RILs (or strains). This is a subfunction needed for `designScore`, but is not directly used.

Usage

```
  interactionLevel( genotype.level, condition.level, markerIndex,
                    nEnvFactors )
```

Arguments

`genotype.level`	levels of genetic factor for each RIL (or strain) in the experiment.
`condition.level`	levels of all environmental factors for each RIL (or strain)in the experiment.
`markerIndex`	indicate which genome position that level of genetic factor corresponds to.
`nEnvFactors`	number of environmental factors, an integer bewteen 1 and 3. When `nEnvFactors` is 1 and the number of levels for the enviromental factor (`nLevels`)is 1, there is one condition in the experiment (i.e. no enviromental perturbation) and thus only genetic factor will be considered in the algorithm. When `nEnvFactors` is 1 and nLevels is larger than 1 or `nEnvFactors` is larger than 1, all main factor(s) and interacting facotr(s) will be included. Examples: If there is a temperature perturbation, then `nEnvFactors` is 1; If there is both temperature and drug treatment perturbation, then `nEnvFactors` is 2.

Details

`markerIndex` indicates the genome position that `genotype.level` corresponds to.
An experiment design is defined to be optimal over all markers if the sum of scores, e.g. A-optimality criterion over all markers is minimized.

Value

a matrix with nRILs rows. The number columns depends on `nEnvFactors`. For example:
If `nEnvFactors` = 1, there is only one interaction term.
If `nEnvFactors` = 2, there are three pair-wise two-way interaction terms and one three-way interaction term.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)
http://gbic.biol.rug.nl/designGG

See Also

`designScore`, `conditionLevel`

`pairLevel` *Pair levels for paired RILs (or strains)*

Description

Pair levels for two RILs (or strains) allocated into one slide (`bTwoColorArray=TRUE`). It is a sub-function needed for `designScore` function, but is not directly used.

Usage

```
pairLevel( xxx, rilNames )
```

Arguments

`xxx`	can be `genotype.level`, `condition.level` or `interaction.level`
`rilNames`	names for all RILs (or strains) that have been selected for the experiment

Details

This function is used only for two-color array.

Value

Pair levels for two RILs (or strains) allocated into one slide.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)
http://gbic.biol.rug.nl/designGG

See Also

See Also `designScore`

`plotAllScores` *Plot scores profiles*

Description

Plot all scores and the temperature at each iteration during the simulated annealing process.

Usage

```
plotAllScores(plot.obj,fileName=NULL)
```

Arguments

`plot.obj`	a list containing: scores, cooling, startTemp, temperature, temperature.step, nIterations and optimality. Details can be found below. scores: A- or D- optimality score of all accepted designs during optimization process. cooling: describes the cooling step in the Simulated Annealing, defined as (new.score \$-\$ now.score)/ now.score. startTemp:starting temperature of the simulated annealing process. temperature:final temperature that the simulated annealing reaches. temperatureStep:temperature decreasing step in the simulated annealing (SA) process. nIterations:number of iterations in the simulated annealing method. optimality:type of optimality, i.e. "A" (A-optimality) or "D" (D-optimality). A-optimality minimizes \$Trace((X'X)^-1)\$, which corresponds to minimum average variance of the parameter estimates. D-optimality minimizes \$det(X'X)^-1\$, which corresponds to minimum generalized variance of the parameter estimates.
`fileName`	the final optimal design table(s) in `csv` format and a plot (in `png` format) of the all scores during SA process (if `plotScores` = TRUE) will be produced. The users can specify the table and plot name by setting `fileName`. If `NULL` (default) it produces files starting with `"myDesignGG"`.

Value

Draw a plot that visualizeds the scores (y-axis) at each iteration during the simulated annealing process (x-axis is time of moving)

Note

The calculation of score is dependent on the choice of optimality.
Cooling is defined as (newScore \$-\$ nowScore)/nowScore.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009) http://gbic.biol.rug.nl/designGG

Examples

```
data(examplePlotObj)
plotAllScores(examplePlotObj)
```

---

temperatureStep — *Calculate the temperature decreasing step for simulated annealing process*

Description

Calculate the temperature decreasing step for simulated annealing process. This is a subfunction needed for `designGG`, but is not directly used.

Usage

```
temperatureStep(startTemp, maxTempStep, endTemp, nIterations)
```

Arguments

`startTemp`	starting temperature of simulated annealing process.
`maxTempStep`	maximum temperature decreasing step for simulated annealing process. The parameter ensures that the multiplicative cooling factor is not smaller than this value. If nIterations is too small, the preferred final temperature (endTemp) may not be reached. See Wit and McClure (2004) for details.
`endTemp`	ending temperature of simulated annealing process. An important optimization parameter. Setting this parameter closer to zero. See Wit and McClure (2004) for details
`nIterations`	number of iterations in the simulated annealing method.

Value

A temperature decreasing step in the simulated annealing process.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009) http://gbic.biol.rug.nl/designGG
Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
E. Wit and J. McClure. Statistics for Microarrays: Design, Analysis and Inference. (2004) Chichester: Wiley.

See Also

`designGG`

`updateDesign` *Updates current design*

Description

Updates current experimental design (including `array.allocation` and `condition.allocation`).

Usage

```
updateDesign( array.allocation, condition.allocation, nRILs,
              nSlides, nEnvFactors, nTuple, bTwoColorArray )
```

Arguments

`array.allocation`	matrix with nArray rows and nRIL columns. Elements of 1/0 indicate this RIL (or strain) is/not selected for this array.
`condition.allocation`	matrix with nCondition rows and nRIL columns. Elements of 1/0 indicate this RIL (or strain) is/not selected for this condition.
`nRILs`	number of RILs (or strains) available for the experiment.
`nSlides`	total number of slides available for experiment.
`nEnvFactors`	number of environmental factors, an integer bewteen 1 and 3. When `nEnvFactors` is 1 and the number of levels for the enviromental factor (`nLevels`)is 1, there is one condition in the experiment (i.e. no enviromental perturbation) and thus only genetic factor will be considered in the algorithm. When `nEnvFactors` is 1 and nLevels is larger than 1 or `nEnvFactors` is larger than 1, all main factor(s) and interacting facotr(s) will be included. Examples: If there is a temperature perturbation, then `nEnvFactors` is 1; If there is both temperature and drug treatment perturbation, then `nEnvFactors` is 2.

`nTuple`	average number of RILs (or strains) to be assigned onto each condition. `nTuple` should be a real number which is larger than 1. If `nTuple` < 1, the algorithm will stop and show the message, `warning: "The number of slides is too small to perform the experiment."`
`bTwoColorArray`	binary variable indicating experiment type: `bTwoColorArray <- TRUE` \#for dual channel experiment `bTwoColorArray <- FALSE` \#for single channel experiment

Details

This function calls two subfunctions: `conditionUpdate` and `arrayUpdate`.

Value

a list with two elements, `array.allocation` and `condition.allocation`.

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)
http://gbic.biol.rug.nl/designGG

`variableNames` *Generate variable names for all factors*

Description

Generate variable names for genetic, environmental factors and interacting terms.

Usage

```
variableNames(nEnvFactors, envFactorNames=NULL)
```

Arguments

`nEnvFactors`	number of environmental factors, an integer bewteen 1 and 3. When `nEnvFactors` is 1 and the number of levels for the enviromental factor (`nLevels`)is 1, there is one condition in the experiment (i.e. no enviromental perturbation) and thus only genetic factor will be considered in the algorithm. When `nEnvFactors` is 1 and nLevels is larger than 1 or `nEnvFactors` is larger than 1, all main factor(s) and interacting facotr(s) will be included. Examples: If there is a temperature

	perturbation, then `nEnvFactors` is 1; If there is both temperature and drug treatment perturbation, then `nEnvFactors` is 2.
`envFactorNames`	a vector with names for all environmental factor(s). For example, for the experiment with two environmental factors of temperature and drug treatment: `envFactorNames <- c( "Temperature", "Dosage" )` Default = `NULL`, then the output will use `"F1"` and `"F2"` to indicate the environmental factors.

Details

generates names for variables, a vector with the length of (variableNumber+1).

Value

When `nEnvFactors` = 1 and `nLevels` = 1, there is no environmetal pertubation in the experimental. Then we re-define `nEnvFactors` to be 0 within the algorithm. Accordingly, `variableNumber` = 1, and `variableNames` is one genetic factor "Q".
When `nEnvFactors` = 1, `variableNumber` = 3, and `variableNames` are one genetic factor "Q", one environmental factor "F", and one interacting factor "QxF".
When `nEnvFactors` = 2, `variableNumber` = 7, and `variableNames` are one genetic factor "Q", two environmental factors "F1" and "F2", three two-way interacting factors "QF1", "QF2", "F1F2", and one three way interacting factors "QxF1xF2".
When `nEnvFactors` = 3, `variableNumber` = 15, and `variableNames` are one genetic factor "Q", three environmental factors "F1", "F2" and "F3", six two-way interacting factors "QF1", "QF2", "QF3", "F1F2", "F2F3" and "F1F3", four three-way interacting factors "QxF1xF2", "QxF1xF3", "QxF2xF3", "F1xF2xF3" and one four-way interacting factors "QxF1xF2xF3".

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009) http://gbic.biol.rug.nl/designGG

See Also

`variableNumber`

`variableNumber` *Compute the number of variables in the experiment*

Description

When `nEnvFactors` = 1 and `nLevels` = 1, there is no environmetal pertubation in the experimental. Then we re-define `nEnvFactors` to be 0 within the algorithm. `nEnvFactors` = 0, only genetic factor is considered.
`nEnvFactors` > 1, genetic and environmental facotrs, and all possible interacting factors are considered.

Usage

```
variableNumber( nEnvFactors )
```

Arguments

`nEnvFactors`	number of environmental factors, an integer. When nEnvFactors is between 0 and 3, all main factors and interacting factors will be included.

Value

`nEnvFactors` = 1, `variableNumber` = 3 (one genetic factor Q, one environmental factor F, and one interacting factor QxF)
`nEnvFactors` = 2, `variableNumber` = 7
`nEnvFactors` = 3, `variableNumber` = 15

Author(s)

Yang Li <yang.li@rug.nl>, Gonzalo Vera <gonzalo.vera.rodriguez@gmail.com>
Rainer Breitling <r.breitling@rug.nl>, Ritsert Jansen <r.c.jansen@rug.nl>

References

Y. Li, R. Breitling and R.C. Jansen. Generalizing genetical genomics: the added value from environmental perturbation, Trends Genet (2008) 24:518-524.
Y. Li, M. Swertz, G. Vera, J. Fu, R. Breitling, and R.C. Jansen. designGG: An R-package and Web tool for the optimal design of genetical genomics experiments. BMC Bioinformatics 10:188(2009)
http://gbic.biol.rug.nl/designGG

See Also

`variableNames`

Index