Package 'clustDRM'

March 15, 2019

Type Package

Version 0.1-0

Date 2019-03-06

Title Clustering Dose-Response Curves and Fitting Appropriate Models to Them

Maintainer Vahid Nassiri <vahid.nassiri@openanalytics.eu>

Description Functions to identify the pattern of a dose-response curve. Then fit a set of appropriate models to it according to the identified pattern, followed by model averaging to estimate the effective dose.

Imports doParallel, parallel, foreach, caret, ORCME, ORIClust, multcomp, IsoGene, DoseFinding, pheatmap, shiny, readr, DT, MCPMod, RColorBrewer,

Suggests testthat,

License GPL (>= 2)

Encoding UTF-8

RoxygenNote 6.1.0

NeedsCompilation no

Author Vahid Nassiri [cre],
Yimer Wasihun [aut],
Martin Otava [aut],
Helena Geys [aut],
Fetene Tekle [aut],
Kanaka Tatikola [aut],
Ziv Shkedy [aut]

Repository CRAN

Date/Publication 2019-03-15 16:53:44 UTC

R topics documented:

clustDRMapp 2
clustDRMappSimple 2

fitDRM 3
generalPatternClustering 5
inputDataMaker 8
monotonePatternClustering 10
plotDoseResponseData 12
plotSimulDRM 14
simulEvalDRM 15

Index **18**

`clustDRMapp` *launch the shiny app for an easier use of the package*

Description

launch the shiny app for an easier use of the package

Usage

```
clustDRMapp()
```

Details

the shiny app made for an easy use of the functionalities of the clustDRM package. It can be launched using command clustDRMapp(). It imports the data (in csv format), performs the clustering on it (for monotone patterns using E2 test, and for general patterns using ORICC2 and MCT). It also can estimate EDp for different dose-response curves using appropriate models. Plotting dose-response curves are also possible for any of these operations. A simulation tab will help to decide about the design of the study using a simulation study.

Author(s)

Vahid Nassiri and Yimer Wasihun.

`clustDRMappSimple` *launch the simpler version of the shiny app for an easier use of the package*

Description

launch the simpler version of the shiny app for an easier use of the package

Usage

```
clustDRMappSimple()
```

Details

a simpler version of the shiny app made for an easy use of the functionalities of the clustDRM package. It can be launched using command clustDRMappSimple(). It imports the data (in csv format), performs the clustering on it (for monotone patterns using E2 test, and for general patterns using ORICC2 and MCT). It also can estimate EDp for different dose-response curves using appropriate models. Plotting dose-response curves are also possible for any of these operations.

Author(s)

Vahid Nassiri and Yimer Wasihun.

`fitDRM` *fitting dose-response model according to the identified pattern.*

Description

function to fit several dose-response candidate models according to the identified pattern, and combine their results using model selection and/or model averaging.

Usage

```
fitDRM(inputDataset, dose, response, ID, subsettingID = NULL,
  transform = c("none", "log", "sRoot", "qRoot", "boxcox"),
  addCovars = ~1, patternClusters, EDp = 0.5, addCovarsVar = TRUE,
  alpha = 0.05, na.rm = FALSE, imputationMethod = c("mean",
  "median"), nCores = 1)
```

Arguments

`inputDataset`	a data frame containing the input dataset, it should at least include dose, response, and ID
`dose`	either a single string or a scalar, indicating the name of the dose column or its index.
`response`	either a single string or a scalar, indicating the name of the response column or its index.
`ID`	either a single string or a scalar, indicating the name of the ID column or its index.
`subsettingID`	a vector of ID's of the subjects, in case one wants to fit the models only to a subset of the data. Default is NULL, i.e., all the subjects in the inputDataset will be used.
`transform`	single string indicating what kind of transform should be applied on the response data. It takes "none" (no transform, dafault), "log" (natural log), "sRoot (square root), and "qRoot" (cubic root), and "boxcox" (Box-Cox transformation).
`addCovars`	formula specifying extra linear covariate, e.g., ~x1+x2

`patternClusters`
: a vector of the same length as the number of rows in inputData (number of subjects) indicating a pattern for each subject. Note that the keywords which are recognized are: "increasing", "decreasing", "flat", "complete", and "up down max at x" and "down up min at x", which x is one of the doses. The "flat" and "complete" patterns would not be considered.

`EDp`
: scalar in (0,1), indicatign with EDp should be computed, default is 0.5 (ED50).

`addCovarsVar`
: logical variable (TRUE as default), idicating whether the variance of the extra covariates preesented in addCovars (unless it is only intercept) should also be computed or not.

`alpha`
: scalar in (0,1), level of significance with default alpha = 0.05.

`na.rm`
: logical variable indicatign whether missing values should be removed (TRUE) or not (FALSE, default)

`imputationMethod`
: signle string taking calues from "mean" (default), and "median", which indicates how the missing values should be treated. "mean" would replace them with the mean of the observed ones, and "median" will use median of them for imputation.

`nCores`
: scalar, indicating the number of cores should be used to perform LRT and MCT tests. Default is 1 which means sequantial computation (no prallel computation).

Details

Note that the dose column of the inputDataset should be a numeric variable.

Value

an object fo class fittedDRM which is a list with the following objects: fittedModels the outcome of DoseFinding::fitMod for all the suitable models estAICNonmonotone: the computed AIC for the models fitted to the subjects with a non-monotone pattern estEDpNonmonotone: the computed EDp for the models fitted to the subjects with a non-monotone pattern estAICMonmonotone: the computed AIC for the models fitted to the subjects with a monotone pattern estEDpMonmonotone: the computed EDp for the models fitted to the subjects with a monotone pattern extraCovarsMonotone: if any extra covariates are added to the model their estimates and possibly standard errors (if addCovarsVar = TRUE) are gievn for subjects with monotone pattern. extraCovarsNonmonotone: if any extra covariates are added to the model their estimates and possibly standard errors (if addCovarsVar = TRUE) are gievn for subjects with non-monotone pattern.

Author(s)

Vahid Nassiri, and Yimer Wasihun

See Also

DoseFinding

Examples

```
## gnerating data
set.seed(11)
doses2Use <-  c(0, 5, 20)
numRep2Use <- c(6, 3, 3)
generatedData <- cbind(rep(1,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05), doses2Use,
numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(generatedData) <- c("ID", "dose", "response", "x1")
for (iGen in 2:15){
genData0 <- cbind(rep(iGen,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05), doses2Use,
numRep2Use, 1), matrix(rnorm(1*sum(numRep2Use)),
sum(numRep2Use), 1))
colnames(genData0) <- c("ID", "dose", "response", "x1")
generatedData <- rbind(generatedData, genData0)
}
## transforming it for clustering
toInput <- inputDataMaker(2, 3, 1, generatedData)
## general pattern clustering
generalPatternClust <- generalPatternClustering(
inputData = toInput$inputData, colsData = toInput$colsData ,
colID = toInput$colID, doseLevels = toInput$doseLevels,
numReplications = toInput$numReplicates, na.rm = FALSE,
imputationMethod = "mean", ORICC = "two", transform = "none",
plotFormat = "eps", LRT = TRUE, MCT = TRUE,
adjustMethod = "BH", nPermute = 100, useSeed = NULL,
theLeastNumberOfMethods = 2, alpha = 0.05, nCores = 1)
## fitDRM
fittedModel <- fitDRM (inputDataset = generatedData, dose = 2,
response = 3, ID = 1, subsettingID = NULL,
transform = c("none"), addCovars = ~x1,
patternClusters =
generalPatternClust$clusteringORICC2Results$clusteringResultsORICC2,
EDp = 0.5, addCovarsVar = TRUE, alpha = 0.05, na.rm = FALSE,
imputationMethod = c("mean"), nCores = 1)
```

---

generalPatternClustering

Clustering dose-response curves based on their pattern

Description

function to cluster dose-response curves based on their pattern.

Usage

```
generalPatternClustering(inputData, colsData, colID, doseLevels,
  numReplications, na.rm = FALSE, imputationMethod = c("mean",
  "median"), ORICC = c("two", "one", "both"), transform = c("none",
  "log", "sRoot", "qRoot", "boxcox"), plotFormat = c("eps", "jpg"),
  LRT = TRUE, MCT = FALSE, adjustMethod = c("BH", "holm", "hochberg",
  "hommel", "bonferroni", "BY", "fdr", "none"), nPermute = 1000,
  useSeed = NULL, theLeastNumberOfMethods = c(1, 2, 3, 4),
  alpha = 0.05, nCores = 1)
```

Arguments

inputData	data matrix which should incluide ID's of the subjects, as well as the measurements (gene expressions, etc.) for all replications of different as columns.
colsData	vector indicating the idex of columns in the inputData which correspond to the measurement for different replications of different doses.
colID	scalar indicating the index of column corresponding to data ID.
doseLevels	vector with dose levels.
numReplications	vector wit hthe same length as doseLevels with number of replications for each dose.
na.rm	logical variable indicatign whether missing values should be removed (TRUE) or not (FALSE, default)
imputationMethod	signle string taking calues from "mean" (default), and "median", which indicates how the missing values should be treated. "mean" would replace them with the mean of the observed ones, and "median" will use median of them for imputation.
ORICC	signle string taking value "two", "one", and "both", indicating which ORICC procedure should be used. "one" refers to one-stage ORICC only, "two" (default) refers to two-stage ORICC only, and "both" will perform both of them.
transform	single string indicating what kind of transform should be applied on the response data. It takes "none" (no transform, dafault), "log" (natural log), "sRoot (square root), and "qRoot" (cubic root), and "boxcox" (Box-Cox transformation).
plotFormat	plotFormat string gets two values "eps" (default), and "jpg" indicating the format of the ouput plot.
LRT	logical indicating whether a permutation-based likelihood ratio test should be applied (TRUE) on the subjects which their trend is identified as non-flast by ORICC1 or not (FALSE).
MCT	logical indicating whether a multiple comparison test (with "UmbrellaWilliams" constrast matrix) should be applied (TRUE) on the subjects which their trend is identified as non-flast by ORICC1 or not (FALSE).
adjustMethod	The method for multiplicity adjustment for p-values. The possible values for this argument are "BH", "holm", "hochberg", "hommel", "bonferroni", "BY", "fdr", "none" with "BH" (Benjamini-Hocheberg) as default.

`nPermute` scalar indicating number of permutations in LRT.

`useSeed` scalar, indicating the seed should be used to generate LRT permutations. The default is NULL.

`theLeastNumberOfMethods`
scalar taking values from 1, 2, 3, and 4, indicating how many methods should approve a non-flat trend that it can be selected. Its value depends on how many tests are asked to be done, for the maximum happens when ORICC = "both" and both LRT and MCT are TRUE. For example, when this argument sets to 2 and ORICC = "two", LRT = TRUE, and MCT = TRUE, it means if two-stage ORICC identifies a non-flat pattern and at least one of the LRT and MCT also accepts (at the level of alpha), then that comound is selected as one with a non-flat pattern. Note that the comparison with alpha is done for adjusted p-values.

`alpha` the significance level to compare the adjusted p-value with it.

`nCores` nCores scalar, indicating the number of cores should be used to perform LRT and MCT tests. Default is 1 which means sequantial computation (no prallel computation).

Details

This function first use ORIIC1 or ORICC2 (or both) to identify the pattern of the dose-response cruve for each subject. Once the pattern is identified, for non-flat ones, a permutation-based likelihood ratio test (for exactly the identified pattern, if LRT = TRUE), and a multiple comparisons test (to test H0: flat vs. H1: non-flat, if MCT = TRUE) will be performed to further filter the flat patterns.

Value

a list of the following objects:

selectedSubjects: a data frame indicating the ID's of the selected subjects in the first columns and the identified trend in the second column.

clusteringORICC1Results and/or clusteringORICC2Results: a list with four elements providing the raw data as the outcome of the ORICC procedure (rawDataORICC1 and/or rawDataORICC2), the pattern identified by the ORICC procedure (clusteringResultsORICC1 and/or clusteringResultsORICC2), results of LRT (resultsLRT) and results of MCT (resultsMCT). Both of them provide the adjusted and unadjusted p-values, but for MCT the selected contrast will be provided as well.

Author(s)

Vahid Nassiri, and Yimer Wasihun.

See Also

ORIClust ORIClust

Examples

```
## gnerating data
set.seed(11)
doses2Use <-  c(0, 5, 20)
numRep2Use <- c(3, 3, 3)
generatedData <- cbind(rep(1,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05), doses2Use,
numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(generatedData) <- c("ID", "dose", "response", "x1")
for (iGen in 2:15){
genData0 <- cbind(rep(iGen,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05), doses2Use,
numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(genData0) <- c("ID", "dose", "response", "x1")
generatedData <- rbind(generatedData, genData0)
}
## transforming it for clustering
toInput <- inputDataMaker(2, 3, 1, generatedData)
## general pattern clustering
generalPatternClust <- generalPatternClustering(inputData = toInput$inputData,
colsData = toInput$colsData ,colID = toInput$colID ,
doseLevels = toInput$doseLevels, numReplications = toInput$numReplicates,
na.rm = FALSE, imputationMethod = "mean",
ORICC = "two", transform = "none",plotFormat = "eps",
LRT = TRUE, MCT = TRUE,
adjustMethod = "BH",
nPermute = 100, useSeed = NULL,
theLeastNumberOfMethods = 2, alpha = 0.05, nCores = 1)
```

`inputDataMaker` *Creating suitable inputData for clustering of the dose-response curve patterns*

Description

function to create needed information as the input of the functions to cluster dose-response cruve patterns.

Usage

```
inputDataMaker(dose, response, ID, inputDataset)
```

Arguments

`dose`	either a single string or a scalar, indicating the name of the dose column or its index.

response	either a single string or a scalar, indicating the name of the response column or its index.
ID	either a single string or a scalar, indicating the name of the ID column or its index.
inputDataset	a data frame containing the input dataset, it should at least include dose, response, and ID

Details

Note that the output of this function can be feed into the function for clustering dose-response curve patterns.

Value

a list with the following elements:

inputDataset: includes the ID (first column), and the response for all doses with their replications for each subject as rows. doseLevels: unique dose levels numReplicatrions: number of replicatios per each unique dose level. colsData: the index of columns with responses. colID: the index of ID column.

Author(s)

Vahid Nassiri, and Yimer Wasihun

Examples

```
## gnerating data
set.seed(11)
doses2Use <-  c(0, 5, 20)
numRep2Use <- c(3, 3, 3)
generatedData <- cbind(rep(1,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05),
doses2Use, numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(generatedData) <- c("ID", "dose", "response", "x1")
for (iGen in 2:15){
genData0 <- cbind(rep(iGen,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05),
doses2Use, numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(genData0) <- c("ID", "dose", "response", "x1")
generatedData <- rbind(generatedData, genData0)
}
## transforming it for clustering
toInput <- inputDataMaker(2, 3, 1, generatedData)
```

monotonePatternClustering *clustering dose-response curves based on their pattern when it is known to be monotone. function to cluster dose-response curves based on their pattern.*

Description

clustering dose-response curves based on their pattern when it is known to be monotone. function to cluster dose-response curves based on their pattern.

Usage

```
monotonePatternClustering(inputData, colsData, colID, doseLevels,
  numReplications, transform = c("none", "log", "sRoot", "qRoot",
  "boxcox"), BHorBY = TRUE, SAM = FALSE, testType = c("E2",
  "Williams", "Marcus", "M", "ModifM"), adjustType = c("BH", "BY"),
  FDRvalue = c(0.05, 0.05), nPermute = c(1000, 1000),
  fudgeSAM = c("pooled", "none"), useSeed = c(NULL, NULL),
  theLeastNumberOfTests = 5, na.rm = FALSE,
  imputationMethod = c("mean", "median"))
```

Arguments

inputData	data matrix which should incluide ID's of the subjects, as well as the measurements (gene expressions, etc.) for all replications of different as columns.
colsData	vector indicating the idex of columns in the inputData which correspond to the measurement for different replications of different doses.
colID	scalar indicating the index of column corresponding to data ID.
doseLevels	vector with dose levels.
numReplications	vector wit hthe same length as doseLevels with number of replications for each dose.
transform	single string indicating what kind of transform should be applied on the response data. It takes "none" (no transform, dafault), "log" (natural log), "sRoot (square root), and "qRoot" (cubic root), and "boxcox" (Box-Cox transformation).
BHorBY	logical inidicating whether monotonicity tests (specified in argument testType) using BH or BY modifications should be performed. Default is TRUE.
SAM	logical indicating whether a SAM procedure should be perfmored. Default is FALSE.
testType	string a subset of c("E2", "Williams", "Marcus", "M", "ModifM"), indicating the monotonicity tests which should be applied.
adjustType	method of adjustment for multi-plicity in case BHorBY = TRUE. It takes values "BH" abd "BY" with "BH" as the default.

`FDRvalue` a numerical vector of length 2 indicating the FDR values for BHorBY and SAM, the default is 0.05 for both.

`nPermute` a numerical vector of length 2 indicating number of permutatation for BHorBY and SAM, the default for both is 1000.

`fudgeSAM` single string takes value from ("pooled", "none") specified the fudge factor in SAM test statistic. The default is "pooled".

`useSeed` a vector of lkength two specifying the seed value for BHorBY and SAM, the default is NULL for both.

`theLeastNumberOfTests`
A scalar indicating the minimum number of tests which should approve a monotone trend to consider a trend montone. The default is 5, i.e., all of the tests should agree on the monotonicity.

`na.rm` logical variable indicatign whether missing values should be removed (TRUE) or not (FALSE, default)

`imputationMethod`
signle string taking calues from "mean" (default), and "median", which indicates how the missing values should be treated. "mean" would replace them with the mean of the observed ones, and "median" will use median of them for imputation.

Value

a list with the following objects:

selectedSubjects: provides the ID and indentified patterns for the subjects which are selected based on the results of various tests and theLeastNumberOfTests.

subjectsPatterns: a vector of the same length as the number of subjects in the input dataset which indicates the identified patterns for all subjects (including flat ones).

resultsBH: a list with the results of selected tests (if BHorBY = TRUE, NULL otherwise).

resultsSAM: a list with results of SAM procedure (if SAM = TRUE, NULL otherwise).

selectedSubjectsBH: a data frame of all of the subjects with then number of tests select them based on adjusted BH or BY methods.

selectedSubjectsSAM: a data frame of all of the subjects with then number of tests select them based on SAM procedure

Author(s)

Vahid Nassiri, and Yimer Wasihun.

See Also

IsoGene IsoGene ORCME

Examples

```
## gnerating data, a sample of size 20
set.seed(11)
doses2Use <-  c(0, 5, 20)
numRep2Use <- c(3, 3, 3)
generatedData <- cbind(rep(1,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05),
doses2Use, numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(generatedData) <- c("ID", "dose", "response", "x1")
for (iGen in 2:20){
genData0 <- cbind(rep(iGen,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05),
doses2Use, numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(genData0) <- c("ID", "dose", "response", "x1")
generatedData <- rbind(generatedData, genData0)
}
## transforming it for clustering
toInput <- inputDataMaker(2, 3, 1, generatedData)
## monotone pattern clustering
monotonePatternClust <- monotonePatternClustering (inputData =
toInput$inputData, colsData = toInput$colsData ,
colID = toInput$colID, doseLevels = toInput$doseLevels,
numReplications = toInput$numReplicates,
BHorBY = TRUE, SAM = FALSE, testType = c("E2"),
adjustType = "BH", FDRvalue = c(0.05, 0.05),
nPermute= c(100, 100), fudgeSAM = "pooled",
useSeed = c(NULL, NULL), theLeastNumberOfTests = 1,
na.rm = FALSE, imputationMethod = "mean")
```

---

plotDoseResponseData *plot dose-response curves*

Description

function to plot dose-response curves with the possibility of adding lines indicating average response per dose levels. Also, provided a pattern for the dose-response curve, it can estimate the expected mean values per dose level for the given pattern and add them to the plot.

Usage

```
plotDoseResponseData(inputDataset, dose, response, ID, subjectID,
  xlab = "Dose", ylab = "Response", addMean = TRUE,
  drcPattern = NULL)
```

Arguments

inputDataset	a data frame containing the input dataset, it should at least include dose, response, and ID
dose	either a single string or a scalar, indicating the name of the dose column or its index.
response	either a single string or a scalar, indicating the name of the response column or its index.
ID	either a single string or a scalar, indicating the name of the ID column or its index.
subjectID	single input as the same type as given ID column with the ID of the subject to plot.
xlab	single string with default "dose", the label on x axis.
ylab	single string with default "response", the label on y axis.
addMean	logical variable indicating whether mean values (connecting with lines) should be plotted or not.
drcPattern	single string showing the idetified pattern using clustering algorithms. The default is NULL. In such case, no extra line will be added to the plot regarding the estimated means via the identified pattern.

Details

with addMean = TRUE, a line will be added to the plot, connecting the averaged response per dose level. But when a pattern is provided for the dose-response curve via drcPattern, then a line will be added to the data with the means estimated assuming the identified pattern. If both addMEan = TRUE and drcPattern != NULL, then two lines will be added to the plot. The line in purplish-colored with cross signs as points is the averaged response value per dose level, and the bluish-colored line with circled cross signs as points represents the estimated mean based on the pattern.

Value

make a plot.

Author(s)

Vahid Nassiri and Yimer Wasihun.

Examples

```
## gnerating data, a sample of size 20
set.seed(11)
doses2Use <-  c(0, 5, 20)
numRep2Use <- c(3, 3, 3)
generatedData <- cbind(rep(1,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05), doses2Use,
numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(generatedData) <- c("ID", "dose", "response", "x1")
```

```
for (iGen in 2:15){
genData0 <- cbind(rep(iGen,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05),
doses2Use, numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(genData0) <- c("ID", "dose", "response", "x1")
generatedData <- rbind(generatedData, genData0)
}
## plotting dose response relation
plotDoseResponseData(generatedData, 2, 3, 1, 2)
## transforming it for clustering
plotDoseResponseData(generatedData, 2, 3, 1, 2,
addMean = FALSE,
drcPattern = "increasing")
```

`plotSimulDRM` *plot results of the simulation study*

Description

a function to make a heatmap of the simulation results for tyhe given measure.

Usage

```
plotSimulDRM(simulDRMobj, quantity2Plot = c("mean", "bias", "mse",
  "variance", "relativeBias", "absBias", "absRelativeBias"))
```

Arguments

`simulDRMobj`	output of simulEvalDRM function
`quantity2Plot`	single string, the measure which should be plotted. Available choices are: c("mean", "bias", "mse", "variance", "relativeBias", "absBias", "absRelativeBias")

Value

a heatmap

Author(s)

Vahid Nassiri and Yimer Wasihun.

Examples

```
## gnerating data, a sample of size 20
set.seed(11)
doses2Use <-  c(0, 5, 20)
numRep2Use <- c(3, 3, 3)
generatedData <- cbind(rep(1,sum(numRep2Use)),
```

```
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05), doses2Use,
 numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(generatedData) <- c("ID", "dose", "response", "x1")
for (iGen in 2:20){
genData0 <- cbind(rep(iGen,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05), doses2Use,
numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(genData0) <- c("ID", "dose", "response", "x1")
generatedData <- rbind(generatedData, genData0)
}
simRes <- simulEvalDRM (pilotData =
generatedData[generatedData$ID == 2, c(2,3)],
doseLevels = c(0, 4, 20),
numReplications = c(6, 3, 3), numSim = 10,
standardDeviation = 1, EDp = 0.5,
funcList = c("linlog", "emax", "sigEmax", "logistic"))
# plot the simulated results
plotSimulDRM(simRes, quantity2Plot = "mse")
```

`simulEvalDRM` *simulation-based evalutation of a dose-response model*

Description

a function to simulate data based different dose-response model using parameters estimated from a provided pilot study. The function then simulate data from the estimated model for the given dose levels and number of replications per dose. Some criteria will be compauted which then can be used to compare different settings.

Usage

```
simulEvalDRM(pilotData, doseLevels, numReplications, numSim,
  standardDeviation, EDp = 0.5, funcList = c("linear", "linlog",
  "exponential", "emax", "sigEmax", "logistic", "betaMod", "quadratic"))
```

Arguments

`pilotData`	a dataset presenting dose-response data from a pilot study. The first column should give the doses and the second one should give the response values.
`doseLevels`	the dose levels which should be used in the simulation study.
`numReplications`	number of replications for each of thes dose levels for the simulated data.
`numSim`	number of times that the simulation study should be replicated.
`standardDeviation`	standard deviation of the generated response.

EDp	scalar in (0,1), indicatign with EDp should be computed to compare different models, default is 0.5 (ED50).
funcList	string vector with models for data generation and fitting, should be selected from c("linear", "linlog", "exponential", "emax", "sigEmax", "logistic", "betaMod","quadratic").

Value

a list with the following elements

estEDp a list of length of funcList providing the estimated EDp from models fitted to data generated from each model in funcList realEDp a vector of length funcList, the EDp's computed based on the estimated parameters from different models fitted to pilotData bestModel a list of length funcList, a frequency table of best selected model for data generated from each model in funcList meanEDp a matrix showing mean of estimated EDp's averaged over numSim replications. biasEDp a matrix showing bias of estimated EDp's averaged over numSim replications. mseEDp a matrix showing MSE of estimated EDp's averaged over numSim replications. varEDp a matrix showing variance of estimated EDp's averaged over numSim replications. relativeBiasEDp a matrix showing relative bias of estimated EDp's averaged over numSim replications. absBiasEDp a matrix showing absolute bias of estimated EDp's averaged over numSim replications. absRelativeBiasEDp a matrix showing absolute bias of estimated EDp's averaged over numSim replications. averagaedAIC a matrix showing AIC's of different models averaged over numSim replications. quantity2Plot which if needed will be passed to plot method.

The output of simulEvalDRM can be passed to the function plotSimulDRM to plot a heatmap for the desired the quantity of interest. Possible quantities are ("mean", "bias", "mse", "variance", "relativeBias", "absBias", "absRelativeBias")

Author(s)

Vahid Nassiri and Yimer Wasihun.

Examples

```
## gnerating data, a sample of size 20
set.seed(11)
doses2Use <-  c(0, 5, 20)
numRep2Use <- c(3, 3, 3)
generatedData <- cbind(rep(1,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05), doses2Use,
 numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(generatedData) <- c("ID", "dose", "response", "x1")
for (iGen in 2:20){
genData0 <- cbind(rep(iGen,sum(numRep2Use)),
MCPMod::genDFdata("logistic",c(5, 3, 10, 0.05), doses2Use,
numRep2Use, 1),
matrix(rnorm(1*sum(numRep2Use)), sum(numRep2Use), 1))
colnames(genData0) <- c("ID", "dose", "response", "x1")
generatedData <- rbind(generatedData, genData0)
}
simRes <- simulEvalDRM (pilotData =
```

```
    generatedData[generatedData$ID == 2, c(2,3)],
    doseLevels = c(0, 4, 20),
    numReplications = c(6, 3, 3), numSim = 10,
    standardDeviation = 1, EDp = 0.5,
    funcList = c("linlog", "emax", "sigEmax", "logistic"))
```

Index

clustDRMapp, 2
clustDRMappSimple, 2

fitDRM, 3

generalPatternClustering, 5

inputDataMaker, 8

monotonePatternClustering, 10

plotDoseResponseData, 12
plotSimulDRM, 14

simulEvalDRM, 15