Package 'PReMiuM'

January 10, 2013

Type Package

Title Post processing for C++ PReMiuM

Version 3.0.4

Date 2012-19-12

Author
David I. Hastie <david.hastie@rsimony.com> and Silvia Liverani <liveranis@gmail.com>

Maintainer Silvia Liverani <liveranis@gmail.com>

Description Dirichlet process Bayesian Clustering, also known as profile regression.

License GPL (>= 3)

LazyLoad yes

Depends R (>= 2.15.1), Rcpp (>= 0.9.15), cluster (>= 1.14.3), ggplot2 (>= 0.9.2.1), grid, clue (>= 0.3-45)

LinkingTo Rcpp

SystemRequirements GNU make

R topics documented:

PReMiuM-package *Dirichlet Process Bayesian Clustering*

Description

Dirichlet Process Bayesian Clustering and functions for the post-processing of its output.

Details

Package:	PReMiuM
Type:	Package
Version:	3.0.1
Date:	2012-12-19
License:	What license is it under?
LazyLoad:	yes

~~ An overview of how to use the package, including the most important ~~ ~~ functions ~~

Author(s)

David Hastie and Silvia Liverani

Maintainer: Silvia Liverani <liveranis@gmail.com>

References

Hastie et al (2012) blah blah

Examples

```
# example for Poisson outcome and Discrete covariates
inputs <- generateSampleDataFile(clusSummaryPoissonDiscrete())
runInfoObj<-profRegr(yModel=inputs$yModel,
    xModel=inputs$xModel, nSweeps=1000,
    nBurn=1000, data=inputs$inputData, output="output",
    covNames = inputs$covNames, outcomeT = inputs$outcomeT,
    fixedEffectsNames = inputs$fixedEffectNames)

dissimObj<-calcDissimilarityMatrix(runInfoObj)
clusObj<-calcOptimalClustering(dissimObj)
```

```
riskProfileObj<-calcAvgRiskAndProfile(clusObj)
clusterOrderObj<-plotRiskProfile(riskProfileObj,"summary.png")
```

`calcAvgRiskAndProfile` *Calculation of the average risks and profiles*

Description

Calculation of the average risks and profiles.

Usage

```
calcAvgRiskAndProfile(clusObj, includeFixedEffects=F)
```

Arguments

`clusObj`	Object of type clusObj.
`includeFixedEffects`	By default this is set to 'FALSE'.

Value

Need to write this

none	dont know

Author(s)

David Hastie and Silvia Liverani

Examples

```
generateDataList <- clusSummaryBernoulliDiscrete()
inputs <- generateSampleDataFile(generateDataList)
runInfoObj<-profRegr(yModel=inputs$yModel, xModel=inputs$xModel, nSweeps=100,
    nBurn=100, data=inputs$inputData, output="output",
    covNames=inputs$covNames)

dissimObj<-calcDissimilarityMatrix(runInfoObj)
clusObj<-calcOptimalClustering(dissimObj)
riskProfileObj<-calcAvgRiskAndProfile(clusObj)
clusterOrderObj<-plotRiskProfile(riskProfileObj,"summary.png",
    whichCovariates=c(1,2))
```

calcDissimilarityMatrix *Calculates the dissimilarity matrix*

Description

Calculates the dissimilarity matrix.

Usage

```
calcDissimilarityMatrix(runInfoObj)
```

Arguments

runInfoObj	Object of type runInfoObj.

Value

Need to write this

disSimRunInfoObj	These are details regarding the run and in the same format as runInfoObj.
disSimMat	The dissimilarity matrix, in vector format. Note that it is diagonal, so this contains the upper triangle diagonal entries.
disSimMatPred	The dissimilarity matrix, again in vector format as above, for the predicted subjects.
lsOptSweep	The optimal partition among those explored by the MCMC, as defined by the least squares method. See Dahl (2006).

Author(s)

David Hastie and Silvia Liverani

Examples

```
generateDataList <- clusSummaryBernoulliDiscrete()
inputs <- generateSampleDataFile(generateDataList)
runInfoObj<-profRegr(yModel=inputs$yModel, xModel=inputs$xModel,
    nSweeps=100, nBurn=100, data=inputs$inputData, output="output",
    covNames=inputs$covNames)

dissimObj<-calcDissimilarityMatrix(runInfoObj)
clusObj<-calcOptimalClustering(dissimObj)
riskProfileObj<-calcAvgRiskAndProfile(clusObj)
clusterOrderObj<-plotRiskProfile(riskProfileObj,"summary.png",
    whichCovariates=c(1,2))
```

`calcOptimalClustering` *Calculation of the optimal clustering*

Description

Calculates the optimal clustering.

Usage

```
calcOptimalClustering(disSimObj, maxNClusters=NULL, useLS=F)
```

Arguments

`disSimObj`	A dissimilarity matrix (in vector format, as the output of the function calcDissimilarityMatrix(), and as described in ?calcDissimilarityMatrix) or a list of dissimilarity matrix, to combine the output of several runs of the MCMC.
`maxNClusters`	Set the maximum number of clusters allowed. This is set to the maximum number explored.
`useLS`	This is set to FALSE by default. If it is set to TRUE then the least-squares method is used for the calculation of the optimal clustering, as described in Molitor et al (2010).

Value

the output is a list with the following elements.

`clusObjRunInfoObj`	Details on this run.
`clusterSizes`	Cluster sizes.
`clusteringPred`	The predicted cluster memberships for the predicted scenarios.
`clusObjDisSimMat`	Dissimilarity matrix.
`clustering`	Cluster memberships.
`nClusters`	Optimal number of clusters.
`avgSilhouetteWidth`	Not sure

Author(s)

David Hastie and Silvia Liverani

Examples

```
generateDataList <- clusSummaryBernoulliDiscrete()
inputs <- generateSampleDataFile(generateDataList)
runInfoObj<-profRegr(yModel=inputs$yModel, xModel=inputs$xModel,
    nSweeps=100, nBurn=100, data=inputs$inputData, output="output",
    covNames=inputs$covNames)

dissimObj<-calcDissimilarityMatrix(runInfoObj)
clusObj<-calcOptimalClustering(dissimObj)
```

`calcPredictions` *Calculates the predictions*

Description

Calculates the predictions.

Usage

```
calcPredictions(riskProfObj,
    predictResponseFileName=NULL,
    doRaoBlackwell=F,fullSweepPredictions=F,fullSweepLogOR=F)
```

Arguments

NOTE: THIS DESCRIPTION HAS NOT BEEN CHANGED FOR R AND THE FUNCTIONALITY HAVE NOT BEEN ADAPTED YET

There is also now an additional function calcPredictions for computing predicted responses, for various prediction scenarios. It is assumed that the predictive allocations and Rao-Blackwell predictions have already been done in C++ (using the –predict=<filename> run time option). The user can provide the function with a file through the predictResponseFileName argument. This file has the number of subjects, followed by a row for each subject, where each row contains values for the response, fixed effects and offset / number of trials (depending on the response model) where available. Missing values in this file are denoted (-999). If the file is not provided then the response, fixed effect and offset data is treated as missing for all subjects. If a subject is missing fixed effect values, then the mean value or 0 category fixed effect is used in the predictions (i.e. no fixed effect contribution to predicted response). If the offset / number of trials is missing this value is taken to be 1 when making predictions. If the response is provided for all subjects, the predicted responses are compared with the observed responses and the bias and rmse are computed.

The function can produce predicted values based on simple allocations (the default), or a Rao-Blackwellised estimate of predictions, where the probabilities of allocations are used instead of actually performing a random allocation.

An example file where the fixed effects can be provided for prediction but the observed response is missing is data/input/example_Poisson_Normal_predictW.txt (there are 2 fixed effects in this example). An example of using the function, which would do the Rao Blackwellised predictions, is given by

>calcPredictions(riskProfileObj, predictResponseFileName='../data/input/example_Poisson_Normal_pre
doRaoBlackwell=T)

An example file where both the observed response and fixed effects are present is in data/input/example_Poisson_Normal_predictYW.txt (there are no fixed effects in this example, but these would just be added as columns between the first and last columns). An example of using the function, which would do the simple predictions using the allocations produced by the C++, is given by

>calcPredictions(riskProfileObj, predictResponseFileName='../data/input/example_Poisson_Normal_pre
doRaoBlackwell=F)

Object of type riskProfObj.

`predictResponseFileName`

`riskProfObj` The file which contains the predictive scenarios, as it is produced by the code and stored.

`doRaoBlackwell` By default this is set to FALSE.

`fullSweepPredictions`

By default this is set to FALSE.

`fullSweepLogOR` By default this is set to FALSE.

Value

Need to write this

`none` it creates files outside

Author(s)

David Hastie and Silvia Liverani

Examples

```
inputs <- generateSampleDataFile(clusSummaryBernoulliDiscrete())

# prediction profiles
preds<-data.frame(matrix(c(0, 0, 1, 0, 0,
0, 0, 1, NA, 0),ncol=5,byrow=TRUE))
colnames(preds)<-names(inputs$inputData)[2:(inputs$nCovariates+1)]

# run profiel regression
runInfoObj<-profRegr(yModel=inputs$yModel, xModel=inputs$xModel,
    nSweeps=100, nBurn=1000, data=inputs$inputData, output="output",
    covNames=inputs$covNames,predict=preds)

# postprocessing
dissimObj <- calcDissimilarityMatrix(runInfoObj)
clusObj <- calcOptimalClustering(dissimObj)
```

```
riskProfileObj <- calcAvgRiskAndProfile(clusObj)
clusterOrderObj <- plotRiskProfile(riskProfileObj,"summary.png",
    whichCovariates=c(1,2))
output_predictions <- calcPredictions(riskProfileObj,fullSweepPredictions=TRUE)
```

`clusSummaryBernoulliDiscrete` *Definition of characteristics of sample datasets for profile regression*

Description

.

Usage

```
clusSummaryBernoulliDiscrete()
clusSummaryBernoulliDiscrete()
clusSummaryBinomialNormal()
clusSummaryCategoricalDiscrete()
clusSummaryNormalDiscrete()
clusSummaryNormalNormal()
clusSummaryPoissonDiscrete()
clusSummaryPoissonNormal()
clusSummaryVarSelectBernoulliDiscrete()
clusSummaryBernoulliMixed()
```

Value

Need to write this

`none`	List of stuff

Author(s)

David Hastie and Silvia Liverani

Examples

```
clusSummaryBernoulliDiscrete()
```

compareClustering *Function to compare different partitions*

Description

Function to compare different partitions.

Usage

```
compareClustering(riskProfileObjA,riskProfileObjB,clusterOrder=NULL)
```

Arguments

riskProfileObjA	dont know
riskProfileObjB	dont know
clusterOrder	dont know

Value

Need to look into this, I don't know how it compares them.

Author(s)

David Hastie and Silvia Liverani

Examples

```
# simulate a dataset
generateDataList <- clusSummaryBernoulliDiscrete()
inputs <- generateSampleDataFile(generateDataList)

# do clustering a first time
runInfoObjA<-profRegr(yModel=inputs$yModel, xModel=inputs$xModel,
    nSweeps=100, nBurn=100, data=inputs$inputData, output="output",
    covNames=inputs$covNames)
dissimObjA<-calcDissimilarityMatrix(runInfoObjA)
clusObjA<-calcOptimalClustering(dissimObjA)
riskProfileObjA<-calcAvgRiskAndProfile(clusObjA)

# do clustering a second time
runInfoObjB<-profRegr(yModel=inputs$yModel, xModel=inputs$xModel,
    nSweeps=100, nBurn=100, data=inputs$inputData, output="output",
    covNames=inputs$covNames)
dissimObjB<-calcDissimilarityMatrix(runInfoObjB)
clusObjB<-calcOptimalClustering(dissimObjB)
riskProfileObjB<-calcAvgRiskAndProfile(clusObjB)
```

```
# compare clustering
compareClustering(riskProfileObjA,riskProfileObjB)
```

`computeAssociatedVariable` *computeAssociatedVariable*

Description

computeAssociatedVariable.

Usage

```
computeAssociatedVariable(subjectValues,clusObj,clusterPlotOrder,latexFile=NULL)
```

Arguments

`subjectValues`	dont know
`clusObj`	dont know
`clusterPlotOrder`	dont know
`latexFile`	dont know

Value

Need to look into this, I don't know how it compares them.

Author(s)

David Hastie and Silvia Liverani

`computeRatioOfVariance` *computeRatioOfVariance*

Description

computeRatioOfVariance.

Usage

```
computeRatioOfVariance(runInfoObj)
```

Arguments

runInfoObj	Object of type runInfoObj

Value

Need to look into this, I don't know how it compares them.

Author(s)

David Hastie and Silvia Liverani

---

generateSampleDataFile

Generate sample data files for profile regression

Description

.

Usage

```
generateSampleDataFile(clusterSummary)
```

Arguments

clusterSummary	A vector of strings of the covariate names as by the column names in the data argument.

Value

The output of this function is a list with the following elements

yModel	The outcome model according to which the data has been generated.
xModel	The covariate model according to which the data has been generated.
inputData	The data.frame that contains the data.
covNames	The names of the covariates.
fixedEffectNames	The names of the fixed effects.

Author(s)

David Hastie and Silvia Liverani

Examples

```
# generation of data for clustering

generateDataList <- clusSummaryBernoulliDiscrete()
inputs <- generateSampleDataFile(generateDataList)
profRegr(yModel=inputs$yModel, xModel=inputs$xModel, nSweeps=100,
    nBurn=100, data=inputs$inputData, output="output",
    covNames=inputs$covNames)
```

`is.wholenumber` *Function to check if a number is a whole number*

Description

Function to check if a number is whole, accounting for a rounding error.

Usage

```
is.wholenumber(x, tol = .Machine$double.eps^0.5)
```

Arguments

`x`	The number to be checked.
`tol`	Tolerance level.

Value

The default method for 'is.wholenumber' returns 'TRUE' if the number provided is a whole number.

Author(s)

David Hastie and Silvia Liverani

Examples

```
is.wholenumber(4) # TRUE
is.wholenumber(3.4) # FALSE
```

margModelPosterior *Marginal Model Posterior*

Description

Compute the marginal model posterior.

Usage

```
margModelPosterior(runInfoObj)
```

Arguments

runInfoObj	boh.

Value

It returns a number of files in the output directory as well as a list with the following elements.

Author(s)

Silvia Liverani

Examples

```
inputs <- generateSampleDataFile(clusSummaryBernoulliDiscrete())

runInfoObj<-profRegr(yModel=inputs$yModel,
         xModel=inputs$xModel, nSweeps=10,
         nBurn=10, data=inputs$inputData, output="output",
         covNames = inputs$covNames,
         fixedEffectsNames = inputs$fixedEffectNames,seed=12567)

margModelPosterior(runInfoObj)
```

plotRiskProfile *Plot the Risk Profiles*

Description

Plots the risk profiles for a profile regression model.

Usage

```
plotRiskProfile(riskProfObj, outFile, showRelativeRisk=F,
    orderBy=NULL, whichClusters=NULL,
    whichCovariates=NULL, useProfileStar=F)
```

Arguments

riskProfObj	An object of type riskProfObj
outFile	Path and file name to save the plot.
showRelativeRisk	Whether to show the relative risk.
orderBy	Order by?
whichClusters	Which clusters to show?
whichCovariates	Optional. Either a vector of indeces or a vector of strings that corresponds to the covariates that are to be displayed. The length of this vector must be greater than 1.
useProfileStar	Whether to use Profile Star??

Value

Need to write this

meanSortIndex

Author(s)

David Hastie and Silvia Liverani

Examples

```
# generation of data for clustering
## generation of fixed effects
fe1<-rnorm(200,0,1)
fe2<-runif(200,0,1)
## generation of the outcome
beta<-c(2,3)
W <- cbind(fe1,fe2)
theta<- c(-7,0,3)
clusterIndex<-c(rep(1,80),rep(2,60),rep(3,60))
mu<-theta[clusterIndex]+W
p<-1/(1+exp(-mu))
outcome<-vector()
for (i in 1:200){
    if(runif(1)<p[i]){
        outcome[i]<-1
    }else{
        outcome[i]<-0
    }
}
## generation of the covariates
covariateProbs<-list(list(c(0.8,0.1,0.1),
    c(0.8,0.1,0.1),
    c(0.8,0.1,0.1),
    c(0.8,0.1,0.1),
    c(0.8,0.1,0.1)),
```

```
    list(c(0.1,0.8,0.1),
    c(0.1,0.8,0.1),
    c(0.1,0.8,0.1),
    c(0.1,0.8,0.1),
    c(0.8,0.1,0.1)),
    list(c(0.8,0.1,0.1),
    c(0.1,0.1,0.8),
    c(0.1,0.1,0.8),
    c(0.1,0.1,0.8),
    c(0.1,0.1,0.8)))
X<-data.frame(Var1=rep(NA,200),Var2=rep(NA,200),
    Var3=rep(NA,200),Var4=rep(NA,200),Var5=rep(NA,200))
for (i in 1:200){
    for (j in 1:5){
        u<-runif(1)
        for(kk in 1:3){
            if(u<cumsum(covariateProbs[[clusterIndex[i]]][[j]])[kk]){
                X[i,j]<-kk-1
                break
            }
        }
    }
}

inputData<-data.frame(cbind(outcome,X,fe1,fe2))

runInfoObj<-profRegr(yModel="Bernoulli", xModel="Discrete",
    nSweeps=100, nBurn=100, data=inputData, output="output",
    covNames=c("Var1","Var2","Var3","Var4","Var5"),
    fixedEffectsNames=c("fe1","fe2"))

dissimObj<-calcDissimilarityMatrix(runInfoObj)
clusObj<-calcOptimalClustering(dissimObj)
riskProfileObj<-calcAvgRiskAndProfile(clusObj)
clusterOrderObj<-plotRiskProfile(riskProfileObj,"summary.png",
    whichCovariates=c(1,2))
```

---

profRegr *Profile Regression*

Description

Fit a profile regression model.

Usage

```
profRegr(covNames, fixedEffectsNames, outcome="outcome",
        outcomeT=NA, data, output="output", hyper, predict,
        nSweeps=1000, nBurn=1000, nProgress=500, nFilter=1,
        nClusInit, seed, yModel="Bernoulli", xModel="Discrete",
```

```
sampler="SliceDependent", alpha=-1, excludeY,
extraYVar, varSelectType, entropy,reportBurnIn=FALSE,
run=TRUE, discreteCovs, continuousCovs)
```

Arguments

covNames	A vector of strings of the covariate names as by the column names in the data argument.
fixedEffectsNames	A vector of strings of the fixed effect names as by the column names in the data argument.
outcome	A string of column of the data argument that contains the outcome. The outcome cannot have missing values - you could consider predicting the value of the outcome for those subjects for which it has not been observed.
outcomeT	A string of column of the data argument that contains the offset (for Poisson outcome) or the number of trials (for Binomial outcome).
data	A data frame which has as columns the outcome, the covariates, the fixed effects if any and the offset (for Poisson outcome) or the number of trials (for Binomial outcome). The outcome cannot have missing values - you could consider predicting the value of the outcome for those subjects for which it has not been observed.
output	Path to folder to save all output files. The covariates can have missing values, which must be coded as 'NA'. There cannot be missing values in the fixed effects - if there are, use an imputation method before using profile regression.
hyper	Path to file with hyperparameters specifications. This is optional: default values are provided for all hyperparameters.
predict	Data frame containing the predictive scenarios. See ?calcPredictions for more details. This is only required if predictions are requested.
nSweeps	Number of iterations of the MCMC after the burn-in period. By default this is 1000.
nBurn	Number of initial iterations of the MCMC to be discarded. By default this is 1000.
reportBurnIn	If TRUE then the burn in iterations are reported in the output files, if set to FALSE they are not. It is set to FALSE by default.
nProgress	The number of sweeps at which to print a progress update. By default this is 500.
nFilter	The frequency (in sweeps) with which to write the output to file. The default value is 1.
nClusInit	The number of clusters individuals should be initially randomly assigned to (Unif[50,60]).
seed	The value for the seed for the random number generator. The default value is the current time.
yModel	The model type for the outcome variable. The options currently available are "Bernoulli", "Poisson", "Binomial", "Categorical" and "Normal". The default value is Bernoulli.

xModel	The model type for the covariates. The options currently available are "Discrete", "Normal" and "Mixed". The default value is "Discrete".
sampler	The sampler type to be used. Options are "SliceDependent", "SliceIndependent" and "Truncated". The default value is "SliceDependent".
alpha	The value to be used if alpha is to remain fixed. If a negative value is used then alpha is updated. The default value is -1.
excludeY	If included only the covariate data X is modelled. By default this is not included.
extraYVar	If included extra Gaussian variance is included in the response model. By default the extra Gaussian variance is not included.
varSelectType	The type of variable selection to be used "None", "BinaryCluster" or "Continuous". The default value is "None".
entropy	If included then we compute allocation entropy. By default the allocation entropy is not included.
run	Logical. If TRUE then the MCMC is run. Set run=FALSE if the MCMC has been run already and it is only required to collect information about the run.
discreteCovs	The names of the discrete covariates among the covariate names, if xModel="Mixed". This and continuousCovs must be defined if xModel="Mixed", while covNames is ignored.
continuousCovs	The names of the discrete covariates among the covariate names, if xModel="Mixed". This and continuousCovs must be defined if xModel="Mixed", while covNames is ignored.

Value

It returns a number of files in the output directory as well as a list with the following elements.

directoryPath	String. Directory path of the output files.
fileStem	String. The
inputFileName	String. Location and file name of input dataset as created by this function for the C++ routines
nSweeps	Integer. The number of sweeps of the MCMC after the burn-in.
nBurn	Integer. The number of iterations in the burn-in period of the MCMC.
reportBurnIn	Logical. Whether the output of the burn-in report should be included.
nFilter	Integer. The frequency (in sweeps) with which to write the output to file. The default value is 1.
nSubjects	Integer. The number of subjects.
nPredictSubjects	Integer. The number of subjects for which to run predictions.
covNames	A vector of strings with the names of the covariates.
xModel	String. The model type for the covariates.
includeResponse	Logical. If FALSE only the covariate data X is modelled.
yModel	String. The model type for the outcome.

varSelect	Logical. If FALSE no variable selection is performed.
varSelectType	String. It specifies what type of variable selection has been performed, if any.
nCovariates	Integer. The number of covariates.
nFixedEffects	Integer. The number of fixed effects.
nCategoriesY	Integer. The number of categories of the outcome, if yModel = "Categorical". It is 1 otherwise.
nCategories	Vector of integers. The number of categories of each covariate, if xModel = "Discrete". It is 1 otherwise.
xMat	A matrix of the covariate data.
yMat	A matrix of the outcome data, including the offset if the outcome is Poisson and the number of trials if the outcome is Binomial.
wMat	A matrix of the fixed effect data.

Author(s)

David Hastie and Silvia Liverani

Examples

```
# example for Poisson outcome and Discrete covariates
inputs <- generateSampleDataFile(clusSummaryPoissonDiscrete())
runInfoObj<-profRegr(yModel=inputs$yModel,
    xModel=inputs$xModel, nSweeps=1000,
    nBurn=1000, data=inputs$inputData, output="output",
    covNames = inputs$covNames, outcomeT = inputs$outcomeT,
    fixedEffectsNames = inputs$fixedEffectNames)

dissimObj<-calcDissimilarityMatrix(runInfoObj)
clusObj<-calcOptimalClustering(dissimObj)
riskProfileObj<-calcAvgRiskAndProfile(clusObj)
clusterOrderObj<-plotRiskProfile(riskProfileObj,"summary.png")

# example with Bernoulli outcome and Mixed covariates
inputs <- generateSampleDataFile(clusSummaryBernoulliMixed())
runInfoObj<-profRegr(yModel=inputs$yModel,
    xModel=inputs$xModel, nSweeps=1000,
    nBurn=1000, data=inputs$inputData, output="output",
    discreteCovs = inputs$discreteCovs,
    continuousCovs = inputs$continuousCovs, nClusInit=10)

dissimObj<-calcDissimilarityMatrix(runInfoObj)
clusObj<-calcOptimalClustering(dissimObj)
riskProfileObj<-calcAvgRiskAndProfile(clusObj)
clusterOrderObj<-plotRiskProfile(riskProfileObj,"summary.png",
    whichCovariates=c(1,2,4,5))
```

summariseVarSelectRho *summariseVarSelectRho*

Description

This function summarises the posterior distribution of rho, a parameter for variable selection only.

Usage

```
summariseVarSelectRho(runInfoObj)
```

Arguments

runInfoObj	Object of type runInfoObj

Value

A list with the following elements.

rho	A matrix that has as many columns as the number of covariates and as many rows as the number of sweeps. This matrix records the samples from the posterior distribution of rho for each covariate at each sweep.
rhoMean	Vector with the column means of the matrix rho above. Each value corresponds to the posterior mean of rho for each covariate.
rhoMedian	Vector with the column medians of the matrix rho above. Each value corresponds to the posterior median of rho for each covariate.
rhoLowerCI	Vector with the column lower confidence intervals of the matrix rho above. Each value corresponds to the lower confidence interval of the posterior distribution of rho for each covariate.
rhoUpperCI	Vector with the column upper confidence intervals of the matrix rho above. Each value corresponds to the upper confidence interval of the posterior distribution of rho for each covariate.

Author(s)

David Hastie and Silvia Liverani

Examples

```
inputs <- generateSampleDataFile(clusSummaryVarSelectBernoulliDiscrete())

runInfoObj<-profRegr(yModel=inputs$yModel,
    xModel=inputs$xModel, nSweeps=100,
    nBurn=1000, data=inputs$inputData, output="output",
    covNames = inputs$covNames, varSelect="Continuous")

rho<-summariseVarSelectRho(runInfoObj)
```

Index