# Package 'ecpc'

May 3, 2021

**Type** Package

**Title** Flexible Co-Data Learning for High-Dimensional Prediction

**Version** 2.0

**Date** 2021-04-19

**Author** Mirrelijn M. van Nee [aut, cre],
Lodewyk F.A. Wessels [aut],
Mark A. van de Wiel [aut]

**Maintainer** Mirrelijn M. van Nee <m.vannee@amsterdamumc.nl>

**Depends** R (>= 3.5.0)

**Imports** glmnet, stats, Matrix, gglasso, mvtnorm, CVXR, multiridge (>= 1.5), survival, pROC

**Suggests** Rsolnp, expm, mgcv, foreach, doParallel, parallel, ggplot2, ggraph, igraph, scales, dplyr, magrittr

**Description** Fit linear, logistic and Cox survival regression models penalised with adaptive multi-group ridge penalties.
The multi-group penalties correspond to groups of covariates defined by (multiple) co-data sources.
Group hyperparameters are estimated with an empirical Bayes method of moments, penalised with an extra level of hyper shrinkage.
Various types of hyper shrinkage may be used for various co-data.
The method accommodates inclusion of unpenalised covariates, posterior selection of covariates and multiple data types.
The model fit is used to predict for new samples.
The name 'ecpc' stands for Empirical Bayes, Co-data learnt, Prediction and Covariate selection.
See Van Nee et al. (2020) <arXiv:2005.04010>.

**License** GPL (>= 3)

**URL** https://arxiv.org/abs/2005.04010

**RoxygenNote** 7.1.1

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2021-05-03 07:20:02 UTC

# R topics documented:



---

ecpc-package *Flexible Co-Data Learning for High-Dimensional Prediction*

---

## Description

Fit linear, logistic and Cox survival regression models penalised with adaptive multi-group ridge penalties. The multi-group penalties correspond to groups of covariates defined by (multiple) co-data sources. Group hyperparameters are estimated with an empirical Bayes method of moments, penalised with an extra level of hyper shrinkage. Various types of hyper shrinkage may be used for various co-data. The method accommodates inclusion of unpenalised covariates, posterior selection of covariates and multiple data types. The model fit is used to predict for new samples. The name 'ecpc' stands for Empirical Bayes, Co-data learnt, Prediction and Covariate selection. See Van Nee et al. (2020) <arXiv:2005.04010>.

## Details

The DESCRIPTION file:

| | |
|---|---|
| Package: | ecpc |
| Type: | Package |
| Title: | Flexible Co-Data Learning for High-Dimensional Prediction |
| Version: | 2.0 |
| Date: | 2021-04-19 |
| Authors@R: | c(person(c("Mirrelijn","M."), "van Nee", role = c("aut", "cre"), email = "m.vannee@amsterdamumc.nl"), per |
| Author: | Mirrelijn M. van Nee [aut, cre], Lodewyk F.A. Wessels [aut], Mark A. van de Wiel [aut] |
| Maintainer: | Mirrelijn M. van Nee <m.vannee@amsterdamumc.nl> |
| Depends: | R (>= 3.5.0) |
| Imports: | glmnet, stats, Matrix, gglasso, mvtnorm, CVXR, multiridge (>= 1.5), survival, pROC |
| Suggests: | Rsolnp, expm, mgcv, foreach, doParallel, parallel, ggplot2, ggraph, igraph, scales, dplyr, magrittr |
| Description: | Fit linear, logistic and Cox survival regression models penalised with adaptive multi-group ridge penalties. T |

License: GPL (>= 3)
URL: https://arxiv.org/abs/2005.04010
RoxygenNote: 7.1.1

Index of help topics:



See ecpc for example code.

## Author(s)


Mirrelijn M. van Nee [aut, cre], Lodewyk F.A. Wessels [aut], Mark A. van de Wiel [aut]

Maintainer: Mirrelijn M. van Nee <m.vannee@amsterdamumc.nl>


---

`createGroupset` *Creates a group set (groups) of variables*

---

## Description

Creates a group set (groups) of variables for categorical co-data (factor, character or boolean input), or for continuous co-data (numeric). Continuous co-data is discretised in non-overlapping groups.

## Usage

```
createGroupset(values,index=NULL,grsize=NULL,ngroup=10,
               decreasing=TRUE,uniform=FALSE,minGroupSize = 50)
```

## Arguments

| | |
|---|---|
| values | Factor, character or boolean vector for categorical co-data, or numeric vector for continuous co-data values. |
| index | Index of the covariates corresponding to the values supplied. Useful if part of the co-data is missing/seperated and only the non-missing/remaining part should be discretised. |
| grsize | Numeric. Size of the groups. Only relevant when values is a numeric vector and uniform=TRUE. |
| ngroup | Numeric. Number of the groups to create. Only relevant when values is a numeric vector and grsize is NOT specified. |
| decreasing | Boolean. If TRUE then values is sorted in decreasing order. |
| uniform | Boolean. If TRUE the group sizes are as equal as possible. |
| minGroupSize | Numeric. Minimum group size. Only relevant when values is a numeric vector and uniform=FALSE. |

## Details

This function is derived from CreatePartition from the GRridge-package, available on Bioconductor. Note that the function name and some variable names have been adapted to match terminology used in other functions in the ecpc-package.

A convenience function to create group sets of variables from external information that is stored in values. If values is a factor then the levels of the factor define the groups. If values is a character vector then the unique names in the character vector define the groups. If values is a Boolean vector then the group set consists of two groups for True and False. If values is a numeric vector, then groups contain the variables corresponding to grsize consecutive values of values. Alternatively, the group size is determined automatically from ngroup. If uniform=FALSE, a group with rank $r$ is of approximate size mingr*(r^f), where f>1 is determined such that the total number of groups equals ngroup. Such unequal group sizes enable the use of fewer groups (and hence faster computations) while still maintaining a good 'resolution' for the extreme values in values. About decreasing: if smaller values mean 'less relevant' (e.g. test statistics, absolute regression coefficients) use decreasing=TRUE, else use decreasing=FALSE, e.g. for p-values. If index is defined, then the group set will use these variable indices corresponding to the values. Useful if the group set should be made for a subset of all variables.

## Value

A list with elements that contain the indices of the variables belonging to each of the groups.

## Author(s)

Mark A. van de Wiel

## See Also

Instead of discretising continuous co-data in a a fixed number of groups, they may be discretised adaptively to learn a discretisation that fits the data well, see: splitMedian.

## Examples

```
#SOME EXAMPLES ON SMALL NR OF VARIABLES

#EXAMPLE 1: group set based on known gene signature (boolean vector)
genset <- sapply(1:100,function(x) paste("Gene",x))
signature <- sapply(seq(1,100,by=2),function(x) paste("Gene",x))
SignatureGroupset <- createGroupset(genset%in%signature) #boolean vector

#EXAMPLE 2: group set based on factor variable
Genetype <- factor(sapply(rep(1:4,25),function(x) paste("Type",x)))
TypeGroupset <- createGroupset(Genetype)

#EXAMPLE 3: group set based on continuous variable, e.g. p-value
pvals <- rbeta(100,1,4)

#Creating a group set of 10 equally-sized groups, corresponding to increasing p-values.
PvGroupset <- createGroupset(pvals, decreasing=FALSE,uniform=TRUE,ngroup=10)

#Alternatively, create a group set of 5 unequally-sized groups,
#with minimal size at least 10. Group size
#increases with less relevant p-values.
# Recommended when nr of variables is large.
PvGroupset2 <- createGroupset(pvals, decreasing=FALSE,uniform=FALSE,
                              ngroup=5,minGroupSize=10)

#EXAMPLE 4: group set based on subset of variables,
#e.g. p-values only available for 50 genes.
genset <- sapply(1:100,function(x) paste("Gene",x))
subsetgenes <- sort(sapply(sample(1:100,50),function(x) paste("Gene",x)))
index <- which(genset%in%subsetgenes)

pvals50 <- rbeta(50,1,6)

#Returns the group set for the subset based on the indices of
#the variables in entire genset.

PvGroupsetSubset <- createGroupset(pvals50, index=index,
                                   decreasing=FALSE,uniform=TRUE, ngroup=5)
#append list with group containing the covariate indices for missing p-values
PvGroupsetSubset <- c(PvGroupsetSubset,
                      list("missing"=which(!(genset%in%subsetgenes))))

#EXAMPLE 5: COMBINING GROUP SETS

#Combines group sets into one list with named components.
#This can be used as input for the ecpc() function.

GroupsetsAll <- list(signature=SignatureGroupset, type = TypeGroupset,
                     pval = PvGroupset, pvalsubset=PvGroupsetSubset)

#NOTE: if one aims to use one group set only, then this should also be
# provided in a list as input for the ecpc() function.
```

```
GroupsetsOne <- list(signature=SignatureGroupset)
```

---

`cv.ecpc` *Cross-validation for 'ecpc'*

---

## Description

Cross-validates 'ecpc' and returns model fit, summary statistics and cross-validated performance measures.

## Usage

```
cv.ecpc(Y,X,type.measure=c("MSE","AUC"),outerfolds=10,
        lambdas=NULL,ncores=1,balance=TRUE,silent=FALSE,...)
```

## Arguments

| | |
|---|---|
| `Y` | Response data; n-dimensional vector (n: number of samples) for linear and logistic outcomes, or `Surv` object for Cox survival. |
| `X` | Observed data; (nxp)-dimensional matrix (p: number of covariates) with each row the observed high-dimensional feature vector of a sample. |
| `type.measure` | Type of cross-validated performance measure returned. |
| `outerfolds` | Number of cross-validation folds. |
| `lambdas` | A vector of global ridge penalties for each fold; may be given, else estimated. |
| `ncores` | Number of cores; if larger than 1, the outer cross-validation folds are processed in parallel over 'ncores' clusters. |
| `balance` | (logistic, Cox) Should folds be balanced in response? |
| `silent` | Should output messages be suppressed (default FALSE)? |
| `...` | Additional arguments used in `ecpc`. |

## Value

A list with the following elements:

| | |
|---|---|
| `ecpc.fit` | List with the ecpc model fit in each fold. |
| `dfPred` | Data frame with information about out-of-bag predictions. |
| `dfGrps` | Data frame with information about estimated group and group set weights across folds. |
| `dfCVM` | Data frame with cross-validated performance metric. |

## See Also

Visualise cross-validated group set weights with `visualiseGroupsetweights` or group weights with `visualiseGroupweights`.

## Examples

```
####################
# Simulate toy data #
####################
p<-300 #number of covariates
n<-100 #sample size training data set
n2<-100 #sample size test data set

#simulate all betas i.i.d. from beta_k~N(mean=0,sd=sqrt(0.1)):
muBeta<-0 #prior mean
varBeta<-0.1 #prior variance
indT1<-rep(1,p) #vector with group numbers all 1 (all simulated from same normal distribution)

#simulate test and training data sets:
Dat<-simDat(n,p,n2,muBeta,varBeta,indT1,sigma=1,model='linear')
str(Dat) #Dat contains centered observed data, response data and regression coefficients

#########################
# Make co-data group sets #
#########################
#Group set: G random groups
G <- 5 #number of groups
#sample random categorical co-data:
categoricalRandom <- as.factor(sample(1:G,p,TRUE))
#make group set, i.e. list with G groups:
groupsetRandom <- createGroupset(categoricalRandom)

######################
# Cross-validate ecpc #
######################
tic<-proc.time()[[3]]
cv.fit <- cv.ecpc(type.measure="MSE",outerfolds=2,
                  Y=Dat$Y,X=Dat$Xctd,
                  groupsets=list(groupsetRandom),
                  groupsets.grouplvl=list(NULL),
                  hypershrinkage=c("none"),
                  model="linear",maxsel=c(5,10,15,20))
toc <- proc.time()[[3]]-tic

str(cv.fit$ecpc.fit) #list containing the model fits on the folds
str(cv.fit$dfPred) #data frame containing information on the predictions
cv.fit$dfCVM #data frame with the cross-validated performance for ecpc
#with/without posterior selection and ordinary ridge
```

---

ecpc *Fit adaptive multi-group ridge GLM with hypershrinkage*

---

## Description

Fits a generalised linear (linear, logistic) or Cox survival model, penalised with adaptive multi-group ridge penalties. The multi-group penalties correspond to groups of covariates defined by (multiple) co-data sources. Group hyperparameters are estimated with an empirical Bayes method of moments, penalised with an extra level of hypershrinkage. Various types of hypershrinkage may be used for various co-data, including overlapping groups, hierarchical groups and continuous co-data.

## Usage

```
ecpc(Y, X, groupsets, groupsets.grouplvl = NULL, hypershrinkage,
unpen = NULL, intrcpt = TRUE, model=c("linear","logistic","cox"),
postselection = "elnet,dense", maxsel = 10,
lambda = NULL, fold = 10, sigmasq = NaN, w = NaN,
nsplits = 100, weights = TRUE, profplotRSS = FALSE, Y2 = NaN, X2 = NaN,
compare = TRUE, mu = FALSE, normalise = FALSE, silent = FALSE,
datablocks = NULL)
```

## Arguments

| | |
|---|---|
| `Y` | Response data; n-dimensional vector (n: number of samples) for linear and logistic outcomes, or `Surv` object for Cox survival. |
| `X` | Observed data; (nxp)-dimensional matrix (p: number of covariates) with each row the observed high-dimensional feature vector of a sample. |
| `groupsets` | Co-data group sets; list with m (m: number of group sets) group sets. Each group set is a list of all groups in that set. Each group is a vector containing the indices of the covariates in that group. |
| `groupsets.grouplvl` | (optional) Group sets on group level used in hypershrinkage; list of m elements (corresponding to 'groupsets'), with NULL if there is no structure on group level, or with a list of groups containing the indices of groups of covariates in that group. May be used for hierarchical groups and to adaptively discretise continuous co-data, see `obtainHierarchy`. |
| `hypershrinkage` | Type of shrinkage that is used on the group level; vector of m strings indicating the shrinkage type (or penalty) that is used for each of the m group sets. String may be of the simple form "type1", or "type1,type2", in which type1 is used to select groups and type2 to estimate the group weights of the selected groups. Possible hypershrinkage types are:<br>c("none","ridge","lasso","hierLasso","lasso,ridge","hierLasso,ridge");<br>"none" for no hypershrinkage, "ridge" (default), "lasso" and "hierLasso" (hierarchical lasso using a latent overlapping group lasso penalty) for group selection possibly be combined with ridge shrinkage. |
| `unpen` | Unpenalised covariates; vector with indices of covariates that should not be penalised. |
| `intrcpt` | Should an intercept be included? Included by default for linear and logistic, excluded for Cox for which the baseline hazard is estimated. |

| | |
|---|---|
| model | Type of model for the response; linear, logistic or Cox. |
| postselection | Type of posterior selection method used to obtain a parsimonious model of maxsel covariates, or FALSE if no parsimonious model is needed. Possible options are "elnet,dense" (default), "elnet,sparse", "BRmarginal,dense", "BRmarginal,sparse" or "DSS". |
| maxsel | Maximum number of covariates to be selected a posteriori, in addition to all unpenalised covariates. If maxsel is a vector, multiple parsimonious models are returned. |
| lambda | Global ridge penalty; if given, numeric value to fix the global ridge penalty and equivalently, the global prior variance. When not given, for linear, by default "ML" is used for estimation for maximum marginal likelihood estimation and "CV" for other models for cross-validation. |
| fold | Number of folds used in inner cross-validation to estimate global ridge penalty lambda. |
| sigmasq | (linear model only) If given, noise level is fixed (Y~N(X*beta,sd=sqrt(sigmasq))). |
| w | Group set weights: m-dimensional vector. If given, group set weights are fixed. |
| nsplits | Number of splits used in the Residual Sum of Squares (RSS) criterion to estimate the optimal hyperlambda. |
| weights | Should weights be used in hypershrinkage to correct for group size (default TRUE)? |
| profplotRSS | Should a profile plot of the residual sum of squares (RSS) criterium be shown? |
| Y2 | (optional) Independent response data to compare with predicted response. |
| X2 | (optional) Independent observed data for which response is predicted. |
| compare | Should an ordinary ridge model be fitted to compare with? |
| mu | Should group prior means be included (default FALSE)? |
| normalise | Should group variances be normalised to sum to 1 (default FALSE)? |
| silent | Should output messages be suppressed (default FALSE)? |
| datablocks | (optional) for multiple data types, the corresponding blocks of data may be given in datablocks; a list of B vectors of the indices of covariates in 'X' that belong to each of the B data blocks. Unpenalised covariates should not be given as seperate block, but can be omitted or included in blocks with penalised covariates. Each datatype obtains a datatype-specific 'tauglobal' as in multiridge. |

## Value

A list with the following elements:

| | |
|---|---|
| beta | Estimated regression coefficients; p-dimensional vector. |
| intercept | If included, the estimated intercept; scalar. |
| tauglobal | Estimated global prior variance; scalar (or vector with datatype-specific global prior variances when multiple 'datablocks' are given).) |
| gammatilde | Estimated group weights before truncating negative weights to 0; vector of dimension the total number of groups. |

| | |
|---|---|
| gamma | Final estimated group weights; vector of dimension the total number of groups. |
| w | Estimated group set weights; m-dimensional vector. |
| penalties | Estimated multi-group ridge penalties; p-dimensinional vector. |
| hyperlambdas | Estimated hyperpenalty parameters used in hypershrinkage; m-dimensional vector. |
| Ypred | If independent test set 'X2' is given, predictions for the test set. |
| MSEecpc | If independent test set 'X2', 'Y2' is given, mean squared error of the predictions. |
| sigmahat | (linear model) Estimated sigma^2. |

If 'compare'=TRUE, ordinary ridge estimates and predictions are given. If in addition multiple 'datablocks' are given, the estimates and predictions for multiridge penalty are given;

| | |
|---|---|
| betaridge | Estimated regression coefficients for ordinary ridge (or multiridge) penalty. |
| interceptridge | Estimated intercept for ordinary ridge (or multiridge) penalty. |
| lambdaridge | Estimated (multi)ridge penalty. |
| Ypredridge | If independent test set 'X2' is given, ordinary ridge (or multiridge) predictions for the test set. |
| MSEridge | If independent test set 'X2', 'Y2' is given, mean squared error of the ordinary ridge (or multiridge) predictions. |

If posterior selection is performed;

| | |
|---|---|
| betaPost | Estimated regression coefficients for parsimonious models. If 'maxsel' is a vector, 'betaPost' is a matrix with each column the vector estimate corresponding to the maximum number of selected covariates given in 'maxsel'. |
| interceptPost | Estimated intercept coefficient for parsimonious models. |
| YpredPost | If independent test set 'X2' is given, posterior selection model predictions for the test set. |
| MSEPost | If independent test set 'X2', 'Y2' is given, mean squared error of the posterior selection model predictions. |

## Author(s)

Mirrelijn van Nee, Lodewyk Wessels, Mark van de Wiel

## References

Van Nee, Mirrelijn M., Lodewyk FA Wessels, and Mark A. van de Wiel. "Flexible co-data learning for high-dimensional prediction." arXiv preprint arXiv:2005.04010 (2020).

Van de Wiel, Mark A., Mirrelijn M. van Nee, and Armin Rauschenberger. "Fast cross-validation for multi-penalty ridge regression." arXiv preprint arXiv:2005.09301 (2020).

## Examples

```
#####################
# Simulate toy data #
#####################
p<-300 #number of covariates
n<-100 #sample size training data set
n2<-100 #sample size test data set

#simulate all betas i.i.d. from beta_k~N(mean=0,sd=sqrt(0.1)):
muBeta<-0 #prior mean
varBeta<-0.1 #prior variance
indT1<-rep(1,p) #vector with group numbers all 1 (all simulated from same normal distribution)

#simulate test and training data sets:
Dat<-simDat(n,p,n2,muBeta,varBeta,indT1,sigma=1,model='linear')
str(Dat) #Dat contains centered observed data, response data and regression coefficients

############################
# Make co-data group sets #
############################
#Group set 1: G random groups
G <- 5 #number of groups
#sample random categorical co-data:
categoricalRandom <- as.factor(sample(1:G,p,TRUE))
#make group set, i.e. list with G groups:
groupsetRandom <- createGroupset(categoricalRandom)

#Group set 2: informative hierarchical group set
continuousCodata <- abs(Dat$beta) #use the magnitude of beta as continuous co-data
#Use adaptive discretisation to find a good discretisation of the continuous co-data;
# discretise in groups of covariates of various sizes:
groupsetHierarchical <- splitMedian(values=continuousCodata,index = 1:p,
                        minGroupSize = 50,split="both")
# and obtain group set on group level that defines the hierarchy:
hierarchy.grouplevel <- obtainHierarchy(groupset = groupsetHierarchical)
#visualise hierarchical groups:
#visualiseGroupset(Groupset = groupsetHierarchical,groupset.grouplvl = hierarchy.grouplevel)

############
# Fit ecpc #
############

#fit ecpc for the two group sets, with ridge hypershrinkage for group set 1,
# and hierarchical lasso and ridge for group set 2.
tic<-proc.time()[[3]]
fit <- ecpc(Y=Dat$Y,X=Dat$Xctd,groupsets=list(groupsetRandom,groupsetHierarchical),
           groupsets.grouplvl=list(NULL,hierarchy.grouplevel),
           hypershrinkage=c("ridge","hierLasso,ridge"),
           model="linear",maxsel=c(5,10,15,20),
           Y2=Dat$Y2,X2=Dat$X2ctd)
toc <- proc.time()[[3]]-tic
```

```
fit$tauglobal #estimated global prior variance
fit$gamma #estimated group weights (concatenated for the group sets)
fit$w #estimated group set weights
summary(fit$beta) #estimated regression coefficients
summary(fit$betaPost) #estimated regression coefficients after posterior selection

c(fit$MSEecpc,fit$MSEridge) #mean squared error on test set for ecpc and ordinary ridge
fit$MSEPost #MSE on the test set of ecpc after posterior selection


###################################
# Fit ecpc for multiple datatypes #
###################################
rankBeta<-order(abs(Dat$beta)) #betas ranked in order of magnitude

#with multiple datatypes (given in datablocks) and informative groups
fit2 <- ecpc(Y=Dat$Y,X=Dat$Xctd[,rankBeta],groupsets=list(list(1:75,76:150,151:225,226:300)),
            groupsets.grouplvl=list(NULL),
            hypershrinkage=c("none"),
            model="linear",maxsel=c(5,10,15,20),
            Y2=Dat$Y2,X2=Dat$X2ctd[,rankBeta],
            datablocks = list(1:floor(p/2),(floor(p/2)+1):p))
```

---

`hierarchicalLasso` *Fit hierarchical lasso using LOG penalty*

---

## Description

Fits a linear regression model penalised with a hierarchical lasso penalty, using a latent overlapping group (LOG) lasso penalty.

## Usage

```
hierarchicalLasso(X, Y, groupset, lambda=NULL)
```

## Arguments

| | |
|---|---|
| `X` | nxp matrix with observed data |
| `Y` | nx1 vector with response data |
| `groupset` | list with hierarchical group indices |
| `lambda` | Scalar. Penalty parameter for the latent overlapping group penalty. |

## Details

The LOG penalty can be used to impose hierarchical constraints in the estimation of regression coefficients (Yan, Bien et al. 2007), e.g. a group of covariates (child node in the hierarchical tree) may be selected only if another group is selected (parent node in the hierarchical tree). This function uses the simple implementation for the LOG penalty described in (Jacob, Obozinski and Vert, 2009). Faster and more scalable algorithms may be available but not yet used in this pacakage.

## Value

A list with the following elements;

| | |
|---|---|
| `betas` | Estimated regression coefficients. |
| `a0` | Estimated intercept. |
| `lambdarange` | Range of penalty parameter used for CV (if lambda was not given). |
| `lambda` | Estimated penalty parameter. |
| `group.weights` | Fixed group weights used in the LOG-penalty. |

## References

Yan, X., Bien, J. et al. (2017). Hierarchical sparse modeling: A choice of two group lasso formulations. Statistical Science 32 531-560.

Jacob, L., Obozinski, G. and Vert, J.-P. (2009). Group lasso with overlap and graph lasso. In: Proceedings of the 26th annual international conference on machine learning 433-440. ACM.

## Examples

```
# Simulate toy data
p<-60 #number of covariates
n<-30 #sample size training data set
n2<-100 #sample size test data set

#simulate all betas i.i.d. from beta_k~N(mean=0,sd=sqrt(0.1)):
muBeta<-c(0,0) #prior mean
varBeta<-c(0.0001,0.1) #prior variance
#vector with group numbers all 1 (all simulated from same normal distribution)
indT1<-rep(c(1,2),each=p/2)

#simulate test and training data sets:
Dat<-simDat(n,p,n2,muBeta,varBeta,indT1,sigma=1,model='linear')
str(Dat) #Dat contains centered observed data, response data and regression coefficients

#hierarchical grouping: e.g. covariates (p/4+1):(p/2) can only be selected when
#covariates 1:(p/4) are selected
groupset <- list(1:(p/2),(p/2+1):p,1:(p/4),(3*p/4+1):p)

#Fit hierarchical lasso, perform CV to find optimal lambda penalty
res <- hierarchicalLasso(X=Dat$Xctd,Y=Dat$Y,groupset = groupset )
res$lambdarange
plot(res$betas)
```

```
#Fit hierarchical lasso for fixed lambda
res2 <- hierarchicalLasso(X=Dat$Xctd,Y=Dat$Y,groupset = groupset,lambda=res$lambdarange[2] )
plot(res2$betas)
```

---

`obtainHierarchy` *Obtain hierarchy*

---

## Description

This function obtains the group set on group level that defines the hierarchy; if a group of covariates g is a subset of group h, then group h is an ancestor of group g (higher up in the hierarchy). This hierarchy is used in adaptively discretising continuous co-data.

## Usage

```
obtainHierarchy(groupset, penalty = "LOG")
```

## Arguments

| | |
|---|---|
| `groupset` | Group set of groups of covariates with nested groups. |
| `penalty` | Default: "LOG" for a latent overlapping group approach (currently the only option in ecpc) |

## Details

We use the latent overlapping group (LOG) lasso penalty to define the hierarchical constraints as described in (Yan, Bien et al. 2007); for each group g of covariates, we make a group on group level with group number g and the group numbers of its ancestors in the hierarchical tree. This way, group g can be selected if and only if all its ancestors are selected. This function assumes that if group g is a subset of group h, then group h is an ancestor of group g. Note that this assumption does not necessarily hold for all hierarchies. The group set on group level should then be coded manually.

## Value

A group set on group level defining the hierarchy.

## References

Yan, X., Bien, J. et al. (2017). Hierarchical sparse modeling: A choice of two group lasso formulations. Statistical Science 32 531-560.

## See Also

`splitMedian` to obtain a group set of nested groups for continuous co-data.

## Examples

```
cont.codata <- seq(0,1,length.out=20) #continuous co-data
#only split at lower continous co-data group
groupset <- splitMedian(values=cont.codata,split="lower",minGroupSize=5)
#obtain groups on group level defining the hierarchy
groupset.grouplvl <- obtainHierarchy(groupset)
```

---

`postSelect` *Perform posterior selection*

---

## Description

Given data and estimated parameters from a previously fit multi-group ridge penalised model, perform posterior selection to find a parsimonious model.

## Usage

```
postSelect(X, Y, beta, intrcpt = 0, penfctr, postselection = c("elnet,dense",
           "elnet,sparse","BRmarginal,dense","BRmarginal,sparse","DSS"),
           maxsel = 30, penalties, model=c("linear","logistic","cox"),
          tauglobal, sigmahat = 1, muhatp = 0, X2 = NaN, Y2 = NaN,silent=FALSE)
```

## Arguments

| | |
|---|---|
| `X` | Observed data: data of p penalised and unpenalised covariates on n samples; (nxp)-dimensional matrix. |
| `Y` | Response data; n-dimensional vector (linear, logistic) or `Surv` object (Cox survival). |
| `beta` | Estimated regression coefficients from the previously fit model. |
| `intrcpt` | Estimated intercept from the previously fit model. |
| `penfctr` | As in glmnet penalty.factor; p-dimensional vector with a 0 if covariate is not penalised, 1 if covariate is penalised. |
| `postselection` | Posterior selection method to be used. |
| `maxsel` | Maximum number of covariates to be selected a posteriori, in addition to all unpenalised covariates. If maxsel is a vector, multiple parsimonious models are returned. |
| `penalties` | Estimated multi-group ridge penalties for all penalised covariates from the previously fit model; vector of length the number of penalised covariates. |
| `model` | Type of model for the response. |
| `tauglobal` | Estimated global prior variance from the previously fit model. |
| `sigmahat` | (linear model only) estimated variance parameter from the previously fit model. |
| `muhatp` | (optional) Estimated multi-group prior means for the penalised covariates from the previously fit model. |

| | |
|---|---|
| X2 | (optional) Independent observed data. |
| Y2 | (optional) Independent response data. |
| silent | Should output messages be suppressed (default FALSE)? |

## Value

A list with the following elements:

| | |
|---|---|
| betaPost | Estimated regression coefficients for parsimonious models. If 'maxsel' is a vector, 'betaPost' is a matrix with each column the vector estimate corresponding to the maximum number of selected covariates given in 'maxsel'. |
| a0 | Estimated intercept coefficient for parsimonious models. |
| YpredPost | If independent test set 'X2' is given, posterior selection model predictions for the test set. |
| MSEPost | If independent test set 'X2', 'Y2' is given, mean squared error of the posterior selection model predictions. |

## Examples

```
#####################
# Simulate toy data #
#####################
p<-300 #number of covariates
n<-100 #sample size training data set
n2<-100 #sample size test data set

#simulate all betas i.i.d. from beta_k~N(mean=0,sd=sqrt(0.1)):
muBeta<-0 #prior mean
varBeta<-0.1 #prior variance
indT1<-rep(1,p) #vector with group numbers all 1 (all simulated from same normal distribution)

#simulate test and training data sets:
Dat<-simDat(n,p,n2,muBeta,varBeta,indT1,sigma=1,model='linear')
str(Dat) #Dat contains centered observed data, response data and regression coefficients

#####################################
# Fit ecpc and perform post-selection #
#####################################
fit <- ecpc(Y=Dat$Y,X=Dat$Xctd,groupsets=list(list(1:p)),
            groupsets.grouplvl=list(NULL),
            hypershrinkage=c("none"),
            model="linear",maxsel=c(5,10,15,20),
            Y2=Dat$Y2,X2=Dat$X2ctd)

fitPost <- postSelect(Y=Dat$Y,X=Dat$Xctd, beta=fit$beta, intrcpt = fit$intercept,
                      maxsel = c(5,10,15,20), penalties=fit$penalties,
                      tauglobal=fit$tauglobal, sigmahat = fit$sigmahat)
summary(fit$betaPost[,1]); summary(fitPost$betaPost[,1])
```

produceFolds *Produce folds*

## Description

Produce folds for cross-validation.

## Usage

```
produceFolds(nsam, outerfold, response, model = c("logistic","cox","other"),
balance = TRUE)
```

## Arguments

| | |
|---|---|
| nsam | Number of samples |
| outerfold | Number of folds. |
| response | Response data. |
| model | Type of model for the response. |
| balance | Should folds be balanced in response? |

## Value

A list with 'outerfold' elements containing a vector of sample indices in each fold.

## Examples

```
n<-100
outerfold <- 10

#linear model
resp <- rnorm(n)
folds <- produceFolds(nsam=n, outerfold=outerfold, response=resp)

#logistic model: keep 0/1 balanced across folds
resp <- as.factor(rnorm(n)>0.5)
folds <- produceFolds(nsam=n, outerfold=outerfold, response=resp, balance = TRUE)
```

simDat *Simulate data*

## Description

Simulate toy data with linear or logistic response.

## Usage

```
simDat(n, p, n2 = 20, muGrp, varGrp, indT, sigma = 1,
  model = c("linear","logistic"), flag = FALSE)
```

## Arguments

| | |
|---|---|
| n | Number of samples for the training set. |
| p | Number of covariates. |
| n2 | Number of independent samples for the test set. |
| muGrp | Prior mean for different groups. |
| varGrp | Prior variance for different groups. |
| indT | True group index of each covariate; p-dimensional vector. |
| sigma | Variance parameter for linear model. |
| model | Type of model. |
| flag | Should linear predictors and true response be plotted? |

## Value

A list with

| | |
|---|---|
| beta | Simulated regression coefficients |
| Xctd | Simulated observed data for training set |
| Y | Simulated response data for test set |
| X2ctd | Simulated observed data for test set |
| Y2 | Simulated response data for test set |

## Examples

```
n<-10
p<-30
#simulate beta from two normal distributions; beta_k ~ N(mu_k,tau^2_k)
muGrp <- c(0,0.1) #mean (mu_1,mu_2)
varGrp <- c(0.05,0.01) #variance (tau^2_1,tau^2_2)
#group number of each covariate; first half in group 1, second half in group 2
indT <- rep(c(1,2),each=15)

dataLin <- simDat(n, p, n2 = 20, muGrp, varGrp, indT, sigma = 1, model = "linear",
```

```
    flag = TRUE)
dataLog <- simDat(n, p, n2 = 20, muGrp, varGrp, indT, model = "logistic",
    flag = TRUE)
```

---

splitMedian *Discretise continuous data in multiple granularities*

---

## Description

Discretise continuous co-data by making groups of covariates of various size. The first group is the group with all covariates. Each group is then recursively split in two at the median co-data value, until some user-specified minimum group size is reached. The discretised groups are used for adaptive discretisation of continuous co-data.

## Usage

```
splitMedian(values, index=NULL, depth=NULL, minGroupSize = 50, first = TRUE,
  split = c("both","lower","higher"))
```

## Arguments

| | |
|---|---|
| values | Vector with the continuous co-data values to be discretised. |
| index | Index of the covariates corresponding to the values supplied. Useful if part of the continuous co-data is missing and only the non-missing part should be discretised. |
| depth | (optional): if given, a discretisation is returned with 'depth' levels of granularity. |
| minGroupSize | Minimum group size that each group of covariates should have. |
| split | "both", "lower" or "higher": should both split groups of covariates be further split, or only the group of covariates that corresponds to the lower or higher continuous co-data group? |
| first | Do not change, recursion help variable. |

## Value

A list with groups of covariates, which may be used as group set in ecpc.

## See Also

Use obtainHierarchy to obtain a group set on group level defining the hierarchy for adaptive discretisation of continuous co-data.

## Examples

```
cont.codata <- seq(0,1,length.out=20) #continuous co-data
#full tree with minimum group size 5
groupset1 <- splitMedian(values=cont.codata,minGroupSize=5)
#only split at lower continous co-data group
groupset2 <- splitMedian(values=cont.codata,split="lower",minGroupSize=5)

part <- sample(1:length(cont.codata),15) #discretise only for a part of the continuous co-data
cont.codata[-part] <- NaN #suppose rest is missing
#make group set of non-missing values
groupset3 <- splitMedian(values=cont.codata[part],index=part,minGroupSize=5)
groupset3 <- c(groupset3,list(which(is.nan(cont.codata)))) #add missing data group
```

---

`visualiseGroupset` *Visualise a group set*

---

## Description

Visualises a group set in a graph, with directed edges indicating the hierarchy.

## Usage

```
visualiseGroupset(Groupset, groupweights, groupset.grouplvl, nodeSize = 10, ls = 1)
```

## Arguments

| | |
|---|---|
| `Groupset` | List of G groups of covariates. |
| `groupweights` | (optional) vector with G group weights; if given, group weights are visualised too. |
| `groupset.grouplvl` | List of G_2 groups defining a hierarchy. |
| `nodeSize` | Size of the nodes in the visualisation; scalar. |
| `ls` | Line size; scalar. |

## Value

A ggplot object.

## See Also

`visualiseGroupsetweights` to plot estimated group set weights. and `visualiseGroupweights` to plot estimated group weights.

## Examples

```
#groups without hierarchical constraints
groupset <- list("Group1"=c(1:20),"Group2"=c(15,30))
visualiseGroupset(groupset,c(0.5,2))

#hierarchical groups
cont.codata <- seq(0,1,length.out=20) #continuous co-data
#only split at lower continous co-data group
hierarchicalgroupset <- splitMedian(values=cont.codata,split="lower",minGroupSize=5)
#obtain groups on group level defining the hierarchy
groupset.grouplvl <- obtainHierarchy(hierarchicalgroupset)
visualiseGroupset(hierarchicalgroupset, groupset.grouplvl=groupset.grouplvl)
```

---

# visualiseGroupsetweights

*Visualise estimated group set weights*

---

## Description

Plot group set weights from multiple cross-validation folds.

## Usage

```
visualiseGroupsetweights(dfGrps, GroupsetNames, hist = FALSE, boxplot = TRUE,
                         jitter = TRUE, ps = 1.5, width = 0.5)
```

## Arguments

| | |
|---|---|
| dfGrps | Data frame containing the following variables; 'Groupset': factor with group set names; 'Groupset.weight': group set weight of each group set; 'Fold': number indicating which fold in the cross-validation is used. |
| GroupsetNames | Vector with names of the group sets. |
| hist | Should histogram be plotted? |
| boxplot | Should boxplot be used or points? |
| jitter | Should group set weights be jittered? |
| ps | Point size. |
| width | Width of jitter. |

## Value

Plot in ggplot object.

## See Also

visualiseGroupset to visualise group sets and visualiseGroupweights to plot estimated group weights.

## Examples

```
dfGrps <- data.frame(Groupset=rep(c(1,2),each=10),
                     Groupset.weight=c(rnorm(10,0,0.01),rnorm(10,1,0.05)),
                     Fold=rep(1:10,2))
GroupsetNames <- c("Groupset1","Groupset2")
visualiseGroupsetweights(dfGrps, GroupsetNames, hist = FALSE, boxplot = TRUE,jitter=TRUE)
```

---

`visualiseGroupweights` *Visualise estimated group weights*

---

## Description

Plot group weights from multiple cross-validation folds.

## Usage

```
visualiseGroupweights(dfGrps, Groupset, groupset.grouplvl, values,
                      widthBoxplot = 0.05, boxplot = TRUE, jitter = TRUE,
                      ps = 1.5, ls = 1)
```

## Arguments

| | |
|---|---|
| `dfGrps` | Data frame containing the following variables; 'Group': factor with group names; 'Group.weight': group weight of each group; 'Fold': number indicating which fold in the cross-validation is used. |
| `Groupset` | List of G elements containing covariate indices for each group |
| `groupset.grouplvl` | (optional): groups on group level, e.g. defining a hierarchical structure. |
| `values` | (optional): values of continuous co-data. If given, group weights are plotted against these value. |
| `widthBoxplot` | Width of boxplot. |
| `boxplot` | Should a boxplot be plotted? |
| `jitter` | Should point estimates be jittered? |
| `ps` | Point size. |
| `ls` | Line size. |

## Value

Plot in ggplot object.

## See Also

`visualiseGroupset` to visualise group sets and `visualiseGroupsetweights` to plot estimated group set weights.

## Examples

```
#discrete groups
groupset1 <- list(1:20,21:40)
dfGrps1 <- data.frame(Group=as.factor(rep(c(1,2),each=10)),
                    Group.weight=c(rnorm(10,0.5,0.01),rnorm(10,2,0.05)),
                    Fold=rep(1:10,2))
visualiseGroupweights(dfGrps1, Groupset=groupset1)

#continous co-data groups
cont.codata <- seq(0,1,length.out=40) #continuous co-data
#only split at lower continous co-data group
groupset2 <- splitMedian(values=cont.codata,split="lower",minGroupSize=10)
#obtain groups on group level defining the hierarchy
groupset.grouplvl <- obtainHierarchy(groupset2)

#simulate random group weights around 1
dfGrps2 <- data.frame(Group=as.factor(rep(1:length(groupset2),each=10)),
                     Group.weight=c(rnorm(10*length(groupset2),1,0.01)),
                     Fold=rep(1:10,length(groupset2)))
#plot group weights per group
visualiseGroupweights(dfGrps2, Groupset=groupset2, groupset.grouplvl=groupset.grouplvl)
#plot group weights per leaf group in the hierarchical tree
visualiseGroupweights(dfGrps2, Groupset=groupset2, groupset.grouplvl=groupset.grouplvl,
                     values=cont.codata)
```

# Index