Package ‘ecpc’

February 27, 2023

Type Package

Title Flexible Co-Data Learning for High-Dimensional Prediction

Version 3.1.1

Date 2023-02-27

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, mgcv, pracma, JOPS, quadprog, checkmate

Suggests Rsolnp, expm, foreach, doParallel, parallel, ggplot2, ggraph, igrAPH, ggpubr, scales, dplyr, magrittr, nnls

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.
Co-data may be continuous or categorical.
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.

License GPL (>= 3)

URL http://dx.doi.org/10.1002/sim.9162

RoxygenNote 7.2.0

NeedsCompilation no

Repository CRAN

Date/Publication 2023-02-27 21:12:30 UTC
R topics documented:

ecpc-package .................................................. 2
coe.ecpc ...................................................... 4
createCon ....................................................... 6
createGroupset ............................................... 7
createS .......................................................... 10
createZforGroupset .......................................... 11
createZforSplines ............................................ 12
cv.ecpc .......................................................... 13
epc ............................................................... 14
hierarchicalLasso ........................................... 22
obtainHierarchy ............................................... 23
plot.ecpc ....................................................... 24
postSelect ...................................................... 26
predict.ecpc ................................................... 28
print.ecpc ...................................................... 30
produceFolds ................................................... 32
simDat .......................................................... 33
splitMedian ..................................................... 34
visualiseGroupset ........................................... 35
visualiseGroupsetweights ................................. 36
visualiseGroupweights ...................................... 37

Index 39

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. Co-data may be continuous or categorical. 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: epc
Type: Package
Title: Flexible Co-Data Learning for High-Dimensional Prediction
Version: 3.1.1
Index of help topics:

- coef.ecpc: Obtain coefficients from 'ecpc' object
- createCon: Create a list of constraints for co-data weight estimation
- createGroupset: Create a group set (groups) of variables
- createS: Create a generalised penalty matrix
- createZforGroupset: Create a co-data matrix Z for a group set
- createZforSplines: Create a co-data matrix Z of splines
- cv.ecpc: Cross-validation for 'ecpc'
- ecpc: Fit adaptive multi-group ridge GLM with hypershrinkage
- ecpc-package: Flexible Co-Data Learning for High-Dimensional Prediction
- hierarchicalLasso: Fit hierarchical lasso using LOG penalty
- obtainHierarchy: Obtain hierarchy
- plot.ecpc: Plot an 'ecpc' object
- postSelect: Perform posterior selection
- predict.ecpc: Predict for new samples for 'ecpc' object
- print.ecpc: Print summary of 'ecpc' object
- produceFolds: Produce folds
- simDat: Simulate data
- splitMedian: Discretise continuous data in multiple granularities
- visualiseGroupset: Visualise a group set
- visualiseGroupsetweights: Visualise estimated group set weights
- visualiseGroupweights: Visualise estimated group weights

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>
**coef.ecpc**  
*Obtain coefficients from 'ecpc' object*

**Description**

Obtain regression coefficients or penalties from an existing model fit given in an 'ecpc' object, re-estimate regression coefficients for a given 'ecpc' object and ridge penalties, or obtain ridge penalties for given prior parameters and co-data.

**Usage**

```r
## S3 method for class 'ecpc'
coef(object, penalties = NULL,  
     X = NULL, Y = NULL,  
     unpen = NULL, intrcpt = TRUE,  
     model = c("linear", "logistic", "cox"),  
     est_beta_method = c("glmnet", "multiridge"), ...)

penalties(object, tauglobal=NULL, sigmahat=NULL, gamma=NULL, gamma0=NULL, w=NULL,  
          Z=NULL, groupsets=NULL,  
          unpen=NULL, datablocks=NULL)
```

**Arguments**

- `object`  
  An 'ecpc' object returned by `ecpc`.

- `penalties`  
  Ridge penalties; p-dimensional vector. If provided to `coef.ecpc`, 'X' and 'Y' should be provided too.

- `tauglobal`  
  Estimated global prior variance; scalar (or vector with datatype-specific global prior variances when multiple 'datablocks' are given). If provided to penalties, 'Z' or 'groupsets' should be provided too.

- `sigmahat`  
  (linear model) Estimated sigma^2. If provided to penalties, 'Z' or 'groupsets' should be provided too.

- `gamma`  
  Estimated co-data variable weights; vector of dimension the total number of groups. If provided to penalties, 'Z' or 'groupsets' should be provided too.

- `gamma0`  
  Estimated co-data variable intercept; scalar. If provided to penalties, 'Z' or 'groupsets' should be provided too.

- `w`  
  Estimated group set weights; m-dimensional vector. If provided to penalties, 'Z' or 'groupsets' should be provided too.

- `X`  
  Observed data; (n*p)-dimensional matrix (p: number of covariates) with each row the observed high-dimensional feature vector of a sample.

- `Y`  
  Response data; n-dimensional vector (n: number of samples) for linear and logistic outcomes, or `Surv` object for Cox survival.

- `Z`  
  List with m co-data matrices. Each element is a (p*G)-dimensional co-data matrix containing co-data on the p variables. Co-data should either be provided in 'Z' or 'groupsets'.
coef.ecpc

**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.

**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.

**est_beta_method**
Package used for estimating regression coefficients, either "glmnet" or "multiridge".

**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 separate block, but can be omitted or included in blocks with penalised covariates. Each datatype obtains a datatype-specific ‘tauglobal’ as in multiridge.

... Other parameters

**Value**
For `coef.ecpc`, a list with:

**intercept**
If included, the estimated intercept; scalar.

**beta**
Estimated regression coefficients; p-dimensional vector.

For penalties: a p-dimensional vector with ridge penalties.

**See Also**

`penalties` for obtaining penalties for given prior parameters and co-data.

**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
```
# Provide co-data #

continuousCodata <- abs(Dat$beta)
Z1 <- cbind(continuousCodata, sqrt(continuousCodata))

# setting 2: splines for informative continuous
Z2 <- createZforSplines(values=continuousCodata)
S1.Z2 <- createS(orderPen=2, G=dim(Z2)[2]) # create difference penalty matrix
Con2 <- createCon(G=dim(Z2)[2], shape="positive+monotone.i") # create constraints

# setting 3: 5 random groups
G <- 5
categoricalRandom <- as.factor(sample(1:G, p, TRUE))
# make group set, i.e. list with G groups:
groupsetRandom <- createGroupset(categoricalRandom)
Z3 <- createZforGroupset(groupsetRandom, p=p)
S1.Z3 <- createS(G=G, categorical = TRUE) # create difference penalty matrix
Con3 <- createCon(G=dim(Z3)[2], shape="positive") # create constraints

# fit ecpc for the three co-data matrices with following penalty matrices and constraints
# note: can also be fitted without paraPen and/or paraCon
Z.all <- list(Z1=Z1, Z2=Z2, Z3=Z3)
paraPen.all <- list(Z2=list(S1=S1.Z2), Z3=list(S1=S1.Z3))
paraCon <- list(Z2=Con2, Z3=Con3)

# Fit ecpc #

tic <- proc.time()[[3]]
fit <- ecpc(Y=Dat$Y, X=Dat$Xctd, 
            Z = Z.all, paraPen = paraPen.all, paraCon = paraCon, 
            model="linear", maxsel=c(5,10,15,20),
            Y2=Dat$Y2, X2=Dat$X2ctd)
toc <- proc.time()[[3]]-tic

# estimate coefficients for twice as large penalties
new_coefficients <- coef(fit, penalties=fit$penalties*2, X=Dat$Xctd, Y=Dat$Y)

# change some prior parameters and find penalties
gamma2 <- fit$gamma; gamma2[1:3] <- 1:3
new_penalties <- penalties(fit, gamma=gamma2, Z=Z.all)
new_coefficients2 <- coef(fit, penalties=new_penalties, X=Dat$Xctd, Y=Dat$Y)
createGroupset

Description
Create 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)

createConset

Description
Create a list of constraints to be used by ecp in estimating G co-data weights. Combine constraints with p-splines to estimate shape-constrained functions, e.g. positive, monotone increasing and/or convex functions.

Usage
createCon(G, shape = "positive+monotone.i+convex")

Arguments
- G: Number of co-data weights that should be estimated subject to constraints.
- shape: Common type of shapes, including 'positive', 'monotone.i' ('monotone.d') for monotonically increasing (decreasing), 'convex' ('concave'), or any combination thereof by attaching multiple with a '+' sign.

Value
A list of the form list(M.ineq = M.ineq, b.ineq = b.ineq) with the matrix M.ineq and vector b.ineq containing the inequality constraints corresponding to the given shape.

See Also
The relation between the prior variance and co-data may be estimated with a shape-constrained spline, see createZforSplines and createS for creating a spline basis and difference penalty matrix for a co-data variable. See ecp for an example.

Examples
# create constraints for positivity
Con1 <- createCon(G=10, shape="positive")
# create constraints for positive and monotonically increasing weights
Con2 <- createCon(G=10, shape="positive+monotone.i")
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 $\text{mng}r^*(r^f)$, 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.
createS <- list(signature=SignatureGroupset)

createS

Create a generalised penalty matrix

Description
Create a generalised penalty matrix which can be used as hypershrinkage for co-data matrix Z.

Usage
createS(orderPen=2, G=10, categorical=FALSE)

Arguments

orderPen The order of the difference penalty. If 0, then a ridge penalty matrix is returned.
G Number of co-data variables to be penalised.
categorical If TRUE, a block correlation matrix is returned.

Value
A (GxG)-dimensional penalty matrix.

References
See for an introduction on p-splines and difference penalties:

See Also
A difference penalty may be applied for p-spline basis functions created with createZforSplines or for categorical co-data created with createZforGroupset.

Examples
S1 <- createS(orderPen=2, G=10) #second difference penalty matrix
S2 <- createS(orderPen=0, G=10) #zeroth order defined as ridge penalty matrix
S3 <- createS(G=10, categorical=TRUE) #difference penalty for unordered categorical groups
createZforGroupset

Create a co-data matrix $Z$ for a group set

Description

Create a co-data matrix $Z$ for a group set as obtained for instance with createGroupset.

Usage

createZforGroupset(groupset, p=NULL)

Arguments

groupset

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

p

Number of covariates in total. If not given, taken as maximum index in 'groupset'. But in cases where some covariates are left unpenalised, the total number of covariates may be larger.

Value

A ($p \times G$)-dimensional co-data matrix.

See Also

createGroupset

Examples

# Group set: $G$ random groups
$G$ <- 5 # number of groups
$p$ <- 300 # number of covariates from which last 10 left unpenalised
# Sample random categorical co-data:
categoricalRandom <- as.factor(sample(1:$G$, (p-10), TRUE))
# Make group set, i.e. list with $G$ groups
groupsetRandom <- createGroupset(categoricalRandom)
Zcat <- createZforGroupset(groupsetRandom, p=$p$)
createZforSplines Create a co-data matrix Z of splines

Description

Create a co-data matrix Z of spline basis functions for a continuous co-data variable.

Usage

createZforSplines(values, G=10, bdeg=3, index=NULL, p=NULL)

Arguments

values A vector with continuous co-data values.
G Number of B-splines.
bdeg Degree of the B-spline basis functions.
index Index of the covariates corresponding to the values supplied. Useful when part
of the co-data is missing/seperated and only the non-missing/remaining part
should be modelled with splines.
p Number of covariates in total. If not given, taken as length of ‘values’. But in
cases where some covariates are left unpenalised, the total number of covariates
may be larger.

Value

A (pxG)-dimensional co-data matrix.

References

See for an introduction on p-splines:
University Press.

See Also

Use createS to create a difference penalty for p-splines.

Examples

#create co-data with random normally distributed values for 100 covariates
values <- rnorm(n=100)
#suppose that there is one additional covariate (the first) that should not be modelled
ind <- 2:101
p<-101
Z <- createZforSplines(values=values,G=10,index=ind,p=p)
cv.ecpc  Cross-validation for 'ecpc'

Description

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

Usage

```r
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 co-data learnt ridge penalties. The ridge penalties correspond to normal prior variances which are regressed on (multiple) co-data sources, e.g. for categorical co-data, each group of variables obtains a group-specific ridge penalty. Co-data weights are estimated with an empirical Bayes method of moments, penalised with an extra level of hypershrinkage and possibly constrained by linear constraints. Various types of hypershrinkage may be used for various co-data, including overlapping groups, hierarchical groups and continuous co-data. P-splines may be used to estimate the relation between the prior variance and continuous co-data variables. This may be combined with linear constraints to estimate shape-constrained functions.

Usage
\[
\text{ecpc}(Y, X, \\
Z=NULL, \text{paraPen=NULL, paraCon=NULL, intrcpt.bam=TRUE, bam.method="ML",} \\
groupsets=NULL, \text{groupsets.groupbylvl=} NULL, \text{hypershinkage=} NULL, \\
unpen = \text{NULL, intrcpt = TRUE, model=c("linear","logistic","cox"),} \\
\text{postselection = "elnet,dense", maxsel = 10,} \\
\text{lambda = NULL, fold = 10, sigmasq = NaN, w = NULL,} \\
\text{nsplits = 100, weights = TRUE, profplotRSS = FALSE, Y2 = NULL, X2 = NULL,} \\
\text{compare = TRUE, mu = FALSE, normalise = FALSE, silent = FALSE,} \\
\text{datablocks = NULL, est_beta_method=c("glmnet","multiridge"))}
\]

Arguments
\begin{itemize}
\item \textbf{Y} \quad \text{Response data; n-dimensional vector (n: number of samples) for linear and logistic outcomes, or \textbf{Surv} object for Cox survival.}
\item \textbf{X} \quad \text{Observed data; (nxp)-dimensional matrix (p: number of covariates) with each row the observed high-dimensional feature vector of a sample.}
\item \textbf{Z} \quad \text{List with m co-data matrices. Each element is a (pxG)-dimensional co-data matrix containing co-data on the p variables. Co-data should either be provided in ‘Z’ or ‘groupsets’.}
\item \textbf{paraPen} \quad \text{A list with generalised ridge penalty matrices used as hypershrinkage in estimating co-data weights, e.g. \textbf{list}("Z2" = \textbf{list}("S1" = M1,"S2"= M2)) when the second co-data source given in ‘Z’ should be penalised by a penalty matrix ‘M1’ and ‘M2’. The names of the elements of the list should be equal to ‘Zi’ where ‘i’ matches the index of the co-data matrix. The list elements should again be lists with elements ‘Si’ for i=1,2,... different generalised ridge penalty matrices.}
\item \textbf{paraCon} \quad \text{A list with linear inequality and or equality constraints used in estimating co-data weights, e.g. \textbf{list}("Z2" = \textbf{list}("M.ineq" = M1,"b.ineq"= b.ineq, "M.eq" = M2,"b.eq"= b.eq)). The names of the elements of the list should be equal to ‘Zi’ where ‘i’ matches the index of the co-data matrix. The list elements should again be lists with elements ‘M.ineq’, ‘b.ineq’ for inequality constraints and ‘M.eq’, ‘b.eq’ for equality constraints, similar to the arguments used in \textbf{lsqlincon} of \textbf{pracma}'.}
\end{itemize}
intrcpt.bam Should an intercept be included in the co-data model? Is used only when ‘Z’ is provided, for which the function bam of ‘mgcv’ is used to fit a generalised additive model.

bam.method When ‘Z’ is provided, ‘bam.method’ indicates the method used in bam of ‘mgcv’ to estimate the hyperpenalties corresponding to the generalised ridge penalty matrices given in ‘paraPen’.

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.

hypershinkage 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 hypershinkage types are:
c("none","ridge","lasso","hierLasso","lasso,ridge","hierLasso,ridge");
"none" for no hypershinkage, "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 separate block, but can be omitted or included in blocks with penalised covariates. Each datatype obtains a datatype-specific ‘tauglobal’ as in multiridge.
est_beta_method  Package used for estimating regression coefficients, either "glmnet" or "multiridge".

Details

Model:
The response is modeled with a generalised linear model with variance \( Var(Y) = \sigma^2 * V(Y) \). For the linear model, \( \sigma^2 \) is the error variance parameter. For the logistic and Cox model, \( \sigma^2 = 1 \). The regression coefficients are independently modeled with a normal prior with prior variance \( v \) regressed on (possibly multiple sources of) co-data

\[
\beta \sim N(0, v), v = \tau_{global}^2 * \text{sum}_d[w_d * Z_d * \gamma_d]
\]

with \( \tau_{global}^2 \) the global scaling parameter, the scalar \( w_d \) the importance weight of co-data set \( d \), \( Z_d \) the co-data matrix for source \( d \) and \( \gamma_d \) the vector of co-data variable weights of source \( d \).

Co-data and hypershrinkage input:
Co-data should be provided in a list of co-data matrices given in argument ‘Z’ or in a list of group sets given in ‘groupsets’. The latter may be used only for (overlapping) groups of variables, whereas the first may be used for continuous co-data too. In most cases, providing co-data in ‘Z’ is faster, so users may want to transform co-data from a group set to a co-data matrix with createZforGroupset.

The co-data variable weights are estimated with an extra level of hypershrinkage, i.e. with a penalised estimator (see below). The type of hypershrinkage may differ per co-data source. Providing these types depends on whether the co-data is provided in ‘Z’ or ‘groupsets’. When co-data is provided in ‘Z’, the hypershrinkage may be provided in the arguments ‘paraPen’, ‘paraCon’, ‘intercept.bam’ and ‘bam.method’ (second line above in usage). When co-data is provided in ‘groupsets’, the hypershrinkage may be provided in the arguments ‘groupsets.grouplvl’ and ‘hypershrinkage’ (third line above in usage).
Estimation:
The regression coefficients are estimated by maximising the penalised likelihood (equiv. maximum a posteriori estimate) for estimated prior parameters:

\[ \hat{\beta} = \text{argmax}_\beta \left[ \log \text{lik} + \sum_k (\beta_k^2 / (2v_k)) \right] \]

The prior parameters are estimated from the data using an empirical Bayes approach; \( \tau_{\text{global}}^2 \) is estimated by maximising the marginal likelihood (linear, default, jointly optimised with \( \sigma^2 \)) or by cross-validation (linear, logistic, Cox). \( \gamma_d \) is estimated per co-data source by finding the minimum (penalised) least squares solution corresponding to the marginal moment equations:

\[ \gamma_d = \text{argmin}_\gamma \left[ ||A\gamma - b||_2^2 + f_{\text{pen}}(\gamma; \lambda_d) \right] \]

with \( f_{\text{pen}} \) some penalty function ('hypershrinkage', see below) depending on hyperpenalty parameter \( \lambda_d \). Co-data weights \( w \) are estimated with a similar, unpenalised marginal moment estimator.

'ecpc' is the first implementation of marginal moment estimation with the additional layer of hypershrinkage. Moment-based estimates without hypershrinkage have been implemented in the R-package 'GRridge'.

Hypershinkage:
For co-data provided in the argument 'Z', a generalised ridge penalty may be used of the type:

\[ \lambda_d \gamma_d^T \mathbf{S} \gamma_d \]

with the penalty matrix \( \mathbf{S} \) possibly a sum of multiple penalty matrices and given in argument 'paraPen'. Additionally, linear (in)equality constraints may be added with the argument 'paraCon', i.e. the least squares estimate is subject to \( M_{\text{neq}} \gamma_d \leq b_{\text{neq}} \) and \( M_{\text{eq}} \gamma_d = b_{\text{eq}} \).

For co-data provided in the argument 'groupsets', the types of hypershrinkage include the ridge penalty \( (\lambda_d \gamma_d^T \gamma_d)^2 \), lasso penalty \( (\lambda_d \gamma_d^T \gamma_d) \) and hierarchical lasso penalty with hierarchy defined in 'groupsets.grouplvl'.

Value
An object of the class 'ecpc' 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.
- **gamma0**: Estimated co-data variable intercept; scalar.
- **w**: Estimated group set weights; m-dimensional vector.
- **penalties**: Estimated multi-group ridge penalties; p-dimensional 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 mutiridge penalty are given;

model Type of model fitted for the response; linear, logistic or cox.

betaridge Estimated regression coefficients for ordinary ridge (or mutiridge) penalty.

interceptridge Estimated intercept for ordinary ridge (or mutiridge) penalty.

lambdaridge Estimated (multi)ridge penalty.

Ypredridge If independent test set 'X2' is given, ordinary ridge (or mutiridge) predictions for the test set.

MSEridge If independent test set 'X2', 'Y2' is given, mean squared error of the ordinary ridge (or mutiridge) 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


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

########################################################################
# Provide co-data in 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)

########################################################################
# ..or in co-data matrices #
########################################################################

#Setting 1: some transformations of informative, continuous co-data
Z1 <- cbind(continuousCodata,sqrt(continuousCodata))

#setting 2: splines for informative continuous
Z2 <- createZforSplines(values=continuousCodata)
S1.Z2 <- createS(orderPen=2, G=dim(Z2)[2]) #create difference penalty matrix
Con2 <- createCon(G=dim(Z2)[2], shape="positive+monotone.i") #create constraints

#setting 3: 5 random groups
Z3 <- createZforGroupset(groupsetRandom,p=p)
S1.Z3 <- createS(G=G, categorical = TRUE) #create difference penalty matrix
Con3 <- createCon(G=dim(Z3)[2], shape="positive") #create constraints

########################################################################
# Fit ecpc on group sets.. #
########################################################################

#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

# ..or on co-data matrices#

# fit ecpc for the three co-data matrices with following penalty matrices and constraints
# note: can also be fitted without paraPen and/or paraCon
Z.all <- list(Z1=Z1, Z2=Z2, Z3=Z3)
paraPen.all <- list(Z2=list(S1=S1.Z2), Z3=list(S1=S1.Z3))
paraCon <- list(Z2=Con2, Z3=Con3)
tic <- proc.time()[[3]]
fit <- ecpc(Y=Dat$Y, X=Dat$Xctd, Z = Z.all, paraPen = paraPen.all, paraCon = paraCon, 
            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 co-data sources)
fit$gamma0 # estimated co-data intercept

# plot contribution of one co-data source
i <- 1
groupsetNO <- c(unlist(sapply(1:length(Z.all), function(i) rep(i, dim(Z.all[[i]])[2]))))
vk <- as.vector(Z.all[[i]]%*%fit$gamma[groupsetNO==i]*fit$tauglobal)
plot(continuousCodata, vk)

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 <- ecpp(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 package.

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


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="Var 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

#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)

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


See Also

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

Examples

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

Description

Make a plot of the fitted regression coefficients versus their corresponding fitted prior variances, or fit the prior variance weight contribution of each co-data source.
Usage

## S3 method for class 'ecpc'
plot(x, show = c("coefficients", "priorweights"),
     Z = NULL, values = NULL, groupsets = NULL,
     codataweights=FALSE, ...)

Arguments

x An 'ecpc' object returned by ecpc.
show Either "coefficients" or "priorweights" to show the fitted regression coefficients or the prior variances. To plot the prior variances, co-data should be provided in either 'Z' or 'groupsets'.
Z List of m co-data matrices, as in ecpc.
values List of m elements, containing p-dimensional vectors with continuous co-data values or NULL. If provided, the prior variances will be plotted versus the provided continuous co-data. If NULL, the prior variances will be plotted per co-data variable.
groupsets Co-data provided as list of group sets, as in ecpc.
codataweights For the option 'show="priorweights"', should the prior variances include the co-data source weights?
...
...

Value

If the packages ‘ggplot2’ and ‘ggpubr’ are installed, a ‘ggplot’ object is shown and returned, else a base plot is shown.

See Also

See ecpc for model fitting.

Examples

# Simulate toy data #
# 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

# Provide co-data #
continuousCodata <- abs(Dat$beta)
Z1 <- cbind(continuousCodata, sqrt(continuousCodata))

#setting 2: splines for informative continuous
Z2 <- createZforSplines(values=continuousCodata)
S1.Z2 <- createS(orderPen=2, G=dim(Z2)[2]) # create difference penalty matrix
Con2 <- createCon(G=dim(Z2)[2], shape="positive+monotone.i") # create constraints

#setting 3: 5 random groups
G <- 5
categoricalRandom <- as.factor(sample(1:G,p,TRUE))
# make group set, i.e. list with G groups:
groupsetRandom <- createGroupset(categoricalRandom)
Z3 <- createZforGroupset(groupsetRandom, p=p)
S1.Z3 <- createS(G=G, categorical = TRUE) # create difference penalty matrix
Con3 <- createCon(G=dim(Z3)[2], shape="positive") # create constraints

# fit ecpc for the three co-data matrices with following penalty matrices and constraints
# note: can also be fitted without paraPen and/or paraCon
Z.all <- list(Z1=Z1, Z2=Z2, Z3=Z3)
paraPen.all <- list(Z2=list(S1=S1.Z2), Z3=list(S1=S1.Z3))
paraCon <- list(Z2=Con2, Z3=Con3)

# Fit ecpc #
tic <- proc.time()[[3]]
fit <- ecpc(Y=Dat$Y, X=Dat$Xctd,
            Z = Z.all, paraPen = paraPen.all, paraCon = paraCon,
            model="linear", maxsel=c(3,10,15,20),
            Y2=Dat$Y2, X2=Dat$X2ctd)
toc <- proc.time()[[3]]-tic

values <- list(NULL, continuousCodata, NULL)

plot(fit, show="coefficients")
plot(fit, show="priorweights", Z=Z.all, values=values)
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(object, X, Y, beta=NULL, intrcpt = 0, penfctr=NULL, 
postselection = c("elnet,dense","elnet,sparse","BRmarginal,dense", "BRmarginal,sparse","DSS"), maxsel = 30, penalties=NULL, 
model=c("linear","logistic","cox"), tauglobal=NULL, sigmahat = NULL, 
muhatp = 0, X2 = NULL, Y2 = NULL, silent=FALSE)

Arguments

object  
An 'ecpc' object returned by ecpc.

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(fit, Y=Dat$Y, X=Dat$Xctd, maxsel = c(5,10,15,20))
summary(fit$betaPost[,1]); summary(fitPost$betaPost[,1])

predict.ecpc  Predict for new samples for 'ecpc' object

Description

Predict the response for new samples based on an 'ecpc' object.

Usage

## S3 method for class 'ecpc'
predict(object, X2, X=NULL, Y=NULL, ...)

### predict.ecpc
Predict for new samples for 'ecpc' object

### Description

Predict the response for new samples based on an 'ecpc' object.

### Usage

## S3 method for class 'ecpc'
predict(object, X2, X=NULL, Y=NULL, ...)

Arguments

- **object**: An 'ecpc' object returned by `ecpc`.
- **X2**: Independent observed data for which response is predicted.
- **X**: Observed data used in fitting the 'object'; (nxp)-dimensional matrix (p: number of covariates) with each row the observed high-dimensional feature vector of a sample.
- **Y**: Response data used in fitting the 'object'; n-dimensional vector (n: number of samples) for linear and logistic outcomes, or `Surv` object for Cox survival.
- **...**: Other parameters

Value

Vector with predicted values. Note that for Cox response, the relative risks are provided, unless training data X and Y is provided to compute the Breslow estimator.

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

# Provide co-data #
continuousCodata <- abs(Dat$beta)
Z1 <- cbind(continuousCodata,sqrt(continuousCodata))

#setting 2: splines for informative continuous
Z2 <- createZforSplines(values=continuousCodata)
S1.Z2 <- createS(orderPen=2, G=dim(Z2)[2]) #create difference penalty matrix
Con2 <- createCon(G=dim(Z2)[2], shape="positive+monotone.i") #create constraints

#setting 3: 5 random groups
G <- 5
categoricalRandom <- as.factor(sample(1:G,p,TRUE))
#make group set, i.e. list with G groups:
groupsetRandom <- createGroupset(categoricalRandom)
```
Z3 <- createZforGroupset(groupsetRandom,p=p)
S1,Z3 <- createS(G=G, categorical = TRUE)  #create difference penalty matrix
Con3 <- createCon(G=dim(Z3)[2], shape="positive")  #create constraints

# fit ecpc for the three co-data matrices with following penalty matrices and constraints
# note: can also be fitted without paraPen and/or paraCon
Z.all <- list(Z1=Z1,Z2=Z2,Z3=Z3)
paraPen.all <- list(Z2=list(S1=S1.Z2), Z3=list(S1=S1.Z3))
paraCon <- list(Z2=Con2, Z3=Con3)

############
# Fit ecpc #
############
tic<-proc.time()[[3]]
fit <- ecpc(Y=Dat$Y, X=Dat$Xctd,
            Z = Z.all, paraPen = paraPen.all, paraCon = paraCon,
            model="linear", maxsel=c(5,10,15,20),
            Y2=Dat$Y2, X2=Dat$X2ctd)
toc <- proc.time()[[3]]-tic
predictions <- predict(fit, X2=Dat$X2ctd)

print.ecpc

---

print.ecpc

Print summary of 'ecpc' object

Description
Print summary of the fitted model given in an 'ecpc' object.

Usage

## S3 method for class 'ecpc'
print(x, ...)

## S3 method for class 'ecpc'
summary(object, ...)

Arguments

x An 'ecpc' object returned by ecpc.

object An 'ecpc' object returned by ecpc.

... ...

See Also
See ecpc for model fitting.
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

########################################################
# Provide co-data #
########################################################
continuousCodata <- abs(Dat$beta)
Z1 <- cbind(continuousCodata,sqrt(continuousCodata))

#setting 2: splines for informative continuous
Z2 <- createZforSplines(values=continuousCodata)
S1.Z2 <- createS(orderPen=2, G=dim(Z2)[2]) #create difference penalty matrix
Con2 <- createCon(G=dim(Z2)[2], shape="positive+monotone.i") #create constraints

#setting 3: 5 random groups
G <- 5
categoricalRandom <- as.factor(sample(1:G,p,TRUE))
#make group set, i.e. list with G groups:
groupsetRandom <- createGroupset(categoricalRandom)
Z3 <- createZforGroupset(groupsetRandom,p=p)
S1.Z3 <- createS(G=G, categorical = TRUE) #create difference penalty matrix
Con3 <- createCon(G=dim(Z3)[2], shape="positive") #create constraints

#fit ecpc for the three co-data matrices with following penalty matrices and constraints
#note: can also be fitted without paraPen and/or paraCon
Z.all <- list(Z1=Z1,Z2=Z2,Z3=Z3)
paraPen.all <- list(Z2=list(S1=S1.Z2), Z3=list(S1=S1.Z3))
paraCon <- list(Z2=Con2, Z3=Con3)

########################################################
# Fit ecpc #
########################################################
tic<-proc.time()[[3]]
fit <- ecpc(Y=Dat$Y,X=Dat$Xctd,
    Z = Z.all, paraPen = paraPen.all, paraCon = paraCon,
    model="linear",maxsel=c(5,10,15,20),
    Y2=Dat$Y2,X2=Dat$X2ctd)
produceFolds

Produce folds

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)
Simulate toy data with linear or logistic response.

Usage

\[
\text{simDat}(n, p, n2 = 20, \text{muGrp}, \text{varGrp}, \text{indT}, \text{sigma} = 1, \\
\text{model} = \text{c("linear","logistic")}, \text{flag} = \text{FALSE})
\]

Arguments

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

Value

A list with

- \text{beta} \quad \text{Simulated regression coefficients}
- \text{Xctd} \quad \text{Simulated observed data for training set}
- \text{Y} \quad \text{Simulated response data for test set}
- \text{X2ctd} \quad \text{Simulated observed data for test set}
- \text{Y2} \quad \text{Simulated response data for test set}

Examples

\[
\begin{align*}
n &< 10 \\
p &< 30 \\
# \text{simulate beta from two normal distributions; } &\text{beta}_k \sim \text{N}(\text{mu}_k,\text{tau}^2_k) \\
\text{muGrp} &<- \text{c}(0,0.1) \quad \# \text{mean } (\mu_1,\mu_2) \\
\text{varGrp} &<- \text{c}(0.05,0.01) \quad \# \text{variance } (\tau^2_1,\tau^2_2) \\
# \text{group number of each covariate; first half in group 1, second half in group 2} \\
\text{indT} &<- \text{rep}(\text{c}(1,2), \text{each}=15) \\
\text{dataLin} &<- \text{simDat}(n, p, n2 = 20, \text{muGrp}, \text{varGrp}, \text{indT}, \text{sigma} = 1, \text{model} = \text{"linear"}, \\
\end{align*}
\]
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.
visualiseGroupset

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 continuous 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
    (optional): groups on group level, e.g. defining a hierarchical structure.
groupset.grouplvl 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)

# continuous co-data groups

codata <- seq(0,1,length.out=40) # continuous co-data
# only split at lower continuous co-data group

groupset2 <- splitMedian(values=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=codata)
Index

c coef.ecpc, 4
createCon, 6
createGroupset, 7, 11
createS, 7, 10, 12
createZforGroupset, 10, 11, 17
createZforSplines, 7, 10, 12
cv.ecpc, 13
e cpc, 3, 4, 7, 13, 14, 25, 27, 29, 30
e cpc-package, 2
hierarchicalLasso, 22
obtainHierarchy, 16, 23, 34
penalties, 5
penalties(coef.ecpc), 4
plot.ecpc, 24
postSelect, 26
predict.ecpc, 28
print.ecpc, 30
produceFolds, 32
simDat, 33
splitMedian, 8, 24, 34
summary.ecpc(print.ecpc), 30
Surv, 4, 13, 15, 27, 29
visualiseGroupset, 35, 36, 37
visualiseGroupsetweights, 13, 35, 36, 37
visualiseGroupweights, 13, 35, 36, 37