

# Package ‘multiridge’

May 9, 2026

**Type** Package

**Title** Fast Cross-Validation for Multi-Penalty Ridge Regression

**Version** 1.11

**Date** 2022-06-13

**Author** Mark A. van de Wiel

**Maintainer** Mark A. van de Wiel <mark.vdwiel@amsterdamumc.nl>

**Depends** R (>= 3.5.0), survival, pROC, methods, mgcv, snowfall

**Description** Multi-penalty linear, logistic and cox ridge regression, including estimation of the penalty parameters by efficient (repeated) cross-validation and marginal likelihood maximization. Multiple high-dimensional data types that require penalization are allowed, as well as unpenalized variables. Paired and preferential data types can be specified. See Van de Wiel et al. (2021), <[doi:10.48550/arXiv.2005.09301](https://doi.org/10.48550/arXiv.2005.09301)>.

**License** GPL (>= 3)

**NeedsCompilation** no

**Repository** CRAN

**Date/Publication** 2022-06-13 15:10:05 UTC

## Contents

multiridge-package . . . . .	2
augment . . . . .	4
betasout . . . . .	5
createXblocks . . . . .	7
createXXblocks . . . . .	8
CVfolds . . . . .	9
CVscore . . . . .	10
dataXXmirmeth . . . . .	11
doubleCV . . . . .	12
fastCV2 . . . . .	15
IWLSCoxridge . . . . .	16
IWLSridge . . . . .	18
mgcv_lambda . . . . .	19

mlikCV . . . . .	20
optLambdas . . . . .	23
optLambdasWrap . . . . .	25
optLambdas_mgcv . . . . .	27
optLambdas_mgcvWrap . . . . .	29
predictIWLS . . . . .	30
Scoring . . . . .	31
setupParallel . . . . .	32
SigmaFromBlocks . . . . .	33

<b>Index</b>	<b>35</b>
--------------	-----------

---

multiridge-package	<i>Fast cross-validation for multi-penalty ridge regression</i>
--------------------	-----------------------------------------------------------------

---

## Description

The package implements multi-penalty linear, logistic and cox ridge regression, including estimation of the penalty parameters by efficient (repeated) cross-validation or marginal likelihood maximization. It allows for multiple high-dimensional data types that require penalization, as well as unpenalized variables. Moreover, it allows a paired penalty for paired data types, and preferential data types can be specified.

## Details

The DESCRIPTION file:

```

Package:    multiridge
Type:      Package
Title:     Fast Cross-Validation for Multi-Penalty Ridge Regression
Version:   1.11
Date:     2022-06-13
Author:    Mark A. van de Wiel
Maintainer: Mark A. van de Wiel <mark.vdwiel@amsterdamumc.nl>
Depends:  R (>= 3.5.0), survival, pROC, methods, mgcv, snowfall
Description: Multi-penalty linear, logistic and cox ridge regression, including estimation of the penalty parameters by efficient cross-validation or marginal likelihood maximization.
License:   GPL (>=3)

```

Index of help topics:

```

CVfolds          Creates (repeated) cross-validation folds
CVscore          Cross-validated score
IWLScoxridge     Iterative weighted least squares algorithm for
                  Cox ridge regression.
IWLSridge        Iterative weighted least squares algorithm for
                  linear and logistic ridge regression.
Scoring          Evaluate predictions

```

<code>SigmaFromBlocks</code>	Create penalized sample cross-product matrix
<code>augment</code>	Augment data with zeros.
<code>betasout</code>	Coefficient estimates from (converged) IWLS fit
<code>createXXblocks</code>	Creates list of (unscaled) sample covariance matrices
<code>createXblocks</code>	Create list of paired data blocks
<code>dataXXmirmeth</code>	Contains R-object 'dataXXmirmeth'
<code>doubleCV</code>	Double cross-validation for estimating performance of 'multiridge'
<code>fastCV2</code>	Fast cross-validation per data block
<code>mgcv_lambda</code>	Maximum marginal likelihood score
<code>mlikCV</code>	Outer-loop cross-validation for estimating performance of marginal likelihood based 'multiridge'
<code>multiridge-package</code>	Fast cross-validation for multi-penalty ridge regression
<code>optLambdas</code>	Find optimal ridge penalties.
<code>optLambdasWrap</code>	Find optimal ridge penalties with sequential optimization.
<code>optLambdas_mgcv</code>	Find optimal ridge penalties with maximum marginal likelihood
<code>optLambdas_mgcvWrap</code>	Find optimal ridge penalties with sequential optimization.
<code>predictIWLS</code>	Predictions from ridge fits
<code>setupParallel</code>	Setting up parallel computing

`betasout`: Coefficient estimates from (converged) IWLS fit

`createXXblocks`: Creates list of (unscaled) sample covariance matrices

`CVscore`: Cross-validated score for given penalty parameters

`dataXXmirmeth`: Example data

`doubleCV`: Double cross-validation for estimating performance

`fastCV2`: Fast cross-validation per data block; no dependency

`IWLSCoxridge`: Iterative weighted least squares algorithm for Cox ridge regression

`IWLSridge`: Iterative weighted least squares algorithm for linear and logistic ridge regression

`mlikCV`: Cross-validation for estimating performance of marginal likelihood estimation

`optLambdasWrap`: Find optimal ridge penalties by cross-validation

`optLambdas_mgcvWrap`: Find optimal ridge penalties in terms of marginal likelihood

`predictIWLS`: Predictions from ridge fits

`setupParallel`: Setting up parallel computing

`SigmaFromBlocks`: Create penalized sample cross-product matrix

### Author(s)

Mark A. van de Wiel (mark.vdwiel@amsterdamumc.nl)

### References

Mark A. van de Wiel, Mirrelijn van Nee, Armin Rauschenberger (2021). Fast cross-validation for high-dimensional ridge regression. *J Comp Graph Stat*

**See Also**

A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

**Examples**

```

data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]

# Find initial lambdas: fast CV per data block separately.
cvperblock2 <- fastCV2(XXblocks=XXmirmeth,Y=resp,kfold=10,fixedfolds = TRUE)
lambdas <- cvperblock2$lambdas

# Create (repeated) CV-splits of the data.
leftout <- CVfolds(Y=resp,kfold=10,nrepeat=3,fixedfolds = TRUE)

# Compute cross-validated score for initial lambdas
CVscore(penalties=lambdas, XXblocks=XXmirmeth,Y=resp,folds=leftout,
score="loglik")

# Optimizes cross-validate criterion (default: log-lik)
# Increase the number of iterations for optimal results
jointlambdas <- optLambdasWrap(penaltiesinit=lambdas, XXblocks=XXmirmeth,Y=resp,
folds=leftout,score="loglik",save=T, maxItropt1=5, maxItropt2=5)

# Alternatively: optimize by using marginal likelihood criterion
## Not run:
jointlambdas2 <- optLambdas_mgcvWrap(penaltiesinit=lambdas, XXblocks=XXmirmeth,
Y=resp)

## End(Not run)

# Optimal lambdas
optlambdas <- jointlambdas$optpen

# Prepare fitting for the optimal lambdas.
XXT <- SigmaFromBlocks(XXmirmeth,penalties=optlambdas)

# Fit. fit$etas contains the n linear predictors
fit <- IWLSridge(XXT,Y=resp)

```

---

augment

*Augment data with zeros.*

---

**Description**

This function augments data with zeros to allow pairing of data on the same variables, but from DIFFERENT samples

**Usage**

```
augment(Xdata1, Xdata2)
```

**Arguments**

Xdata1	Data frame or data matrix of dimension $n_1 \times p$ .
Xdata2	Data frame or data matrix of dimension $n_2 \times p$

**Details**

Xdata1 and Xdata2 should have the same number of columns. These columns represent variables. Augments both data matrices with zeros, such that the matrices can be paired using [createXXblocks](#) on the output of this function.

**Value**

List	
Xaug1	Augmented data matrix 1
Xaug2	Augmented data matrix 2

**Examples**

```
#Example
#Simulate
n1 <- 10
n2 <- 20
p <- 100
X1 <- matrix(rnorm(p*n1),nrow=n1)
X2 <- matrix(rnorm(p*n2),nrow=n2)

#check whether column dimension is correct
ncol(X1)==ncol(X2)

#create cross-product
Xaugm <- augment(X1,X2)

#check dimensions (should be (n1+n2) x p)
dim(Xaugm[[1]])
dim(Xaugm[[2]])
```

---

betasout

*Coefficient estimates from (converged) IWLS fit*


---

**Description**

Extracts estimated regression coefficients from the final Iterative Weighted Least Squares fit, as obtained from linear, logistic, or Cox ridge regression.

**Usage**

```
betasout(IWLSfit, Xblocks, X1=NULL, penalties, pairing = NULL)
```

**Arguments**

IWLSfit	List object, see details
Xblocks	List of data frames or matrices, representing $b=1, \dots, B$ data blocks of dimensions $n \times p_b$ .
X1	Matrix. Dimension $n \times p_0$ , $p_0 < n$ , representing unpenalized covariates.
penalties	Numerical vector.
pairing	Numerical vector of length 3 or NULL.

**Details**

IWLSfit should be the output of either [IWLSridge](#) or [IWLScoxridge](#). Xblocks may be created by [createXblocks](#).

**Value**

List. Number of components equals number of components of Xblocks plus one, as the output is augmented with an intercept estimate (first component, NULL if absent). Each component is a numerical vector representing regression parameter estimates. Lengths of vectors match column dimensions of Xblocks (nr of variables for given data type)

**See Also**

[createXblocks](#). A full demo and data are available from:  
<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

**Examples**

```
data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]
lambdas <- c(100,1000)

# Prepare fitting for the specified penalties.
XXT <- SigmaFromBlocks(XXmirmeth,penalties=lambdas)

# Fit. fit$etas contains the n linear predictors
fit <- IWLSridge(XXT,Y=resp)

# Computation of the regression coefficients requires the original
# (large!) nxp data sets, available from link above
## Not run:
Xbl <- createXblocks(list(datamir,datameth))
betas <- betasout(fit, Xblocks=Xbl, penalties=lambdas)

## End(Not run)
```

---

createXblocks	<i>Create list of paired data blocks</i>
---------------	------------------------------------------

---

### Description

Create list of paired data blocks

### Usage

```
createXblocks(datablocks, which2pair = NULL)
```

### Arguments

datablocks	List of data frames or matrices representing $b=1, \dots, B$ data blocks of dimensions $n \times p_b$ .
which2pair	Integer vector of size 2 (or NULL)

### Details

Only use this function when you wish to pair two data blocks. If `which2pair = NULL` the output matches the input. If not, the function adds a paired data block, pairing the two data blocks corresponding to the elements of `which2pair`.

### Value

List. Same length as `datablocks` when `which2pair = NULL`, or augmented with one paired data block.

### See Also

[createXXblocks](#). A full demo and data are available from:  
<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

### Examples

```
n <- 43
p <- 100
fakeXbl <- createXblocks(list(X1 = matrix(rnorm(n*p), nrow=n), X2 = matrix(rnorm(n*p), nrow=n)))
```

---

createXXblocks	<i>Creates list of (unscaled) sample covariance matrices</i>
----------------	--------------------------------------------------------------

---

### Description

Creates list of (unscaled) sample covariance matrices  $X_b$  for data blocks  $b = 1, \dots, B$ .

### Usage

```
createXXblocks(datablocks, datablocksnew = NULL, which2pair = NULL)
```

### Arguments

datablocks	List of data frames or matrices
datablocksnew	List of data frames or matrices
which2pair	Integer vector of size 2 (or NULL)

### Details

The efficiency of `multiridge` for high-dimensional data relies largely on this function: all iterative calculation are performed on the output of this function, which contains  $B$  blocks of  $n \times n$  matrices. If `which2pair`  $\neq$  NULL, the function adds a paired covariance block, pairing the two data blocks corresponding to the elements of `which2pair`. If predictions for new samples are desired, one also needs to specify `datablocksnew`, which should have the exact same format as `datablocks` with matching column dimension (number of variables).

### Value

List. Same number of component as `datablocks` when `which2pair` = NULL, or augmented with one paired data block. Dimension is  $n \times n$  for all components.

### See Also

[createXblocks](#), which is required when parameter estimates are desired (not needed for prediction). A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

### Examples

```
#Example
#Simulate
Xb11 <- matrix(rnorm(1000),nrow=10)
Xb12 <- matrix(rnorm(2000),nrow=10)

#check whether dimensions are correct
ncol(Xb11)==nrow(Xb12)

#create cross-product
```

```

XXbl <- createXXblocks(list(Xbl1,Xbl2))

#suppose penalties for two data types equal 5,10, respectively
Sigma <- SigmaFromBlocks(XXbl,c(5,10))

#check dimensions (should be n x n)
dim(Sigma)

```

---

CVfolds *Creates (repeated) cross-validation folds*

---

### Description

Creates (repeated) cross-validation folds for samples

### Usage

```
CVfolds(Y, model = NULL, balance = TRUE, kfold = 10, fixedfolds = TRUE, nrepeat = 1)
```

### Arguments

Y	Response vector: numeric, binary, factor or survival.
model	Character. Any of c("linear", "logistic", "cox"). Is inferred from Y when NULL.
balance	Boolean. Should the splits be balanced in terms of response labels?
kfold	Integer. Desired fold.
fixedfolds	Boolean. Should fixed splits be used for reproducibility?
nrepeat	Numeric. Number of repeats.

### Details

Creates (repeated), possibly balanced, splits of the samples. Computing time will often largely depend on on  $kfold \times nrepeat$ , the number of training-test splits evaluated.

### Value

List object with  $kfold \times nrepeat$  elements containing the sample indices of the left-out samples per split.

### See Also

A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

### Examples

```

data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
leftout <- CVfolds(Y=resp,kfold=10,nrepeat=3,fixedfolds = TRUE)

```

---

CVscore	<i>Cross-validated score</i>
---------	------------------------------

---

### Description

Cross-validated score for given penalty parameters.

### Usage

```
CVscore(penalties, XXblocks, Y, X1 = NULL, pairing = NULL, folds, intercept =
  ifelse(is(Y, "Surv"), FALSE, TRUE), frac1 = NULL, score = "loglik", model =
  NULL, eps = 1e-07, maxItr = 100, trace = FALSE, printCV = TRUE, save = FALSE,
  parallel = FALSE)
```

### Arguments

penalties	Numeric vector.
XXblocks	List of nxn matrices. Usually output of <a href="#">createXXblocks</a> .
Y	Response vector: numeric, binary, factor or survival.
X1	Matrix. Dimension $n \times p_{-0}$ , $p_{-0} < n$ , representing unpenalized covariates
pairing	Numerical vector of length 3 or NULL when pairs are absent. Represents the indices (in XXblocks) of the two data blocks involved in pairing, plus the index of the paired block.
folds	List of integer vector. Usually output of <a href="#">CVfolds</a> .
intercept	Boolean. Should an intercept be included?
frac1	Scalar. Prior fraction of cases. Only relevant for model="logistic".
score	Character. See Details.
model	Character. Any of <code>c("linear", "logistic", "cox")</code> . Is inferred from Y when NULL.
eps	Scalar. Numerical bound for IWLS convergence.
maxItr	Integer. Maximum number of iterations used in IWLS.
trace	Boolean. Should the output of the IWLS algorithm be traced?
printCV	Boolean. Should the CV-score be printed on screen?
save	Boolean. If TRUE appends the penalties and resulting CVscore to global variable <code>allscores</code>
parallel	Boolean. Should computation be done in parallel? If TRUE, requires to run <a href="#">setupParallel</a> first.

### Details

See [Scoring](#) for details on score.

**Value**

Numeric, cross-validated prediction score for given penalties

**See Also**

[doubleCV](#) for double cross-validation, used for performance evaluation

**Examples**

```
data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]

# Find initial lambdas: fast CV per data block separately.
cvperblock2 <- fastCV2(XXblocks=XXmirmeth,Y=resp,kfold=10,fixedfolds = TRUE)
lambdas <- cvperblock2$lambdas

# Create training-test splits
leftout <- CVfolds(Y=resp,kfold=10,nrepeat=3,fixedfolds = TRUE)
CVscore(penalties=lambdas, XXblocks=XXmirmeth,Y=resp,folds=leftout,score="loglik")
```

---

dataXXmirmeth	<i>Contains R-object</i> dataXXmirmeth
---------------	----------------------------------------

---

**Description**

This list object contains the binary response (control/case) and two data blocks corresponding to miRNA and methylation data

**Usage**

```
data(dataXXmirmeth)
```

**Format**

The format is a list with two components: resp: numeric (0/1) [1:43] \ XXmirmeth: list with 2 components, each a matrix [1:43,1:43] \

**Details**

The object XXmirmeth is created by applying `createXXblocks(list(datamir, datameth))`, where objects `datamir` and `datameth` are large data matrices stored in the `mirmethdata.Rdata` file, which is available from the link below.

**Source**

Snoek, B. C. et al. (2019), Genome-wide microRNA analysis of HPV-positive self-samples yields novel triage markers for early detection of cervical cancer, *International Journal of Cancer* 144(2), 372-379.

Verlaat, W. et al. (2018), Identification and validation of a 3-gene methylation classifier for hpv-based cervical screening on self-samples, *Clinical Cancer Research* 24(14), 3456-3464.

**References**

Mark A. van de Wiel, Mirrelijn van Nee, Armin Rauschenberger (2021). Fast cross-validation for multi-penalty high-dimensional ridge regression. *J Comp Graph Stat*

**See Also**

createXXblocks. Source data file is available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

**Examples**

```
data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]
```

---

doubleCV

---

*Double cross-validation for estimating performance of multiridge*


---

**Description**

Double cross-validation for estimating performance of multiridge. Outer fold is for testing, inner fold for penalty parameter tuning

**Usage**

```
doubleCV(penaltiesinit, XXblocks, Y, X1 = NULL, pairing = NULL, outfold = 5,
  infold = 10, nrepeatout = 1, nrepeatin = 1, balance = TRUE, fixedfolds =
  TRUE, intercept = ifelse(is(Y, "Surv"), FALSE, TRUE), frac1 = NULL,
  score = "loglik", model = NULL, eps = 1e-07, maxItr = 10, trace = FALSE,
  printCV = TRUE, reltol = 1e-04, optmethod1 = "SANN", optmethod2 =
  ifelse(length(penaltiesinit) == 1, "Brent", "Nelder-Mead"), maxItrOpt1 = 10,
  maxItrOpt2 = 25, save = FALSE, parallel = FALSE, pref = NULL, fixedpen = NULL)
```

**Arguments**

penaltiesinit Numeric vector. Initial values for penalty parameters. May be obtained from [fastCV2](#).

XXblocks List of nxn matrices. Usually output of [createXXblocks](#).

Y Response vector: numeric, binary, factor or survival.

X1	Matrix. Dimension $n \times p_{\theta}$ , $p_{\theta} < n$ , representing unpenalized covariates
pairing	Numerical vector of length 3 or NULL when pairs are absent. Represents the indices (in XXblocks) of the two data blocks involved in pairing, plus the index of the paired block.
outfold	Integer. Outer fold for test samples.
inifold	Integer. Inner fold for tuning penalty parameters.
nrepeatout	Integer. Number of repeated splits for outer fold.
nrepeatin	Integer. Number of repeated splits for inner fold.
balance	Boolean. Should the splits be balanced in terms of response labels?
fixedfolds	Boolean. Should fixed splits be used for reproducibility?
intercept	Boolean. Should an intercept be included?
frac1	Scalar. Prior fraction of cases. Only relevant for model="logistic".
score	Character. See Details.
model	Character. Any of c("linear", "logistic", "cox"). Is inferred from Y when NULL.
eps	Scalar. Numerical bound for IWLS convergence.
maxItr	Integer. Maximum number of iterations used in IWLS.
trace	Boolean. Should the output of the IWLS algorithm be traced?
printCV	Boolean. Should the CV-score be printed on screen?
reltol	Scalar. Relative tolerance for optimization methods.
optmethod1	Character. First, global search method. Any of the methods c("Brent", "Nelder-Mead", "Sann") may be used, but simulated annealing by "Sann" is recommended to search a wide landscape. Other unconstrained methods offered by <a href="#">optim</a> may also be used, but have not been tested.
optmethod2	Character. Second, local search method. Any of the methods c("Brent", "Nelder-Mead", "Sann") may be used, but "Nelder-Mead" is generally recommended. Other unconstrained methods offered by <a href="#">optim</a> may also be used, but have not been tested.
maxItropt1	Integer. Maximum number of iterations for optmethod1.
maxItropt2	Integer. Maximum number of iterations for optmethod2.
save	Boolean. If TRUE appends the penalties and resulting CVscore to global variable allscores
parallel	Boolean. Should computation be done in parallel? If TRUE, requires to run <a href="#">setupParallel</a> first.
pref	Integer vector or NULL. Contains indices of data types in XXblocks that are preferential.
fixedpen	Integer vector or NULL. Contains indices of data types of which penalty is fixed to the corresponding value in penaltiesinit.

## Details

**WARNING:** this function may be very time-consuming. The number of evaluations may equal  $nrepeatout * outerfold * nrepeatin * innerfold * maxItr * (maxItropt1 + maxItropt2)$ . Computing time may be estimated by multiplying computing time of `optLambdasWrap` by  $nrepeatout * outerfold$ . See [Scoring](#) for details on score.

## Value

List with the following components:

<code>sampleindex</code>	Numerical vector: sample indices
<code>true</code>	True responses
<code>linpred</code>	Cross-validated linear predictors

## See Also

[optLambdas](#), [optLambdasWrap](#) which optimize the penalties. [Scoring](#) which may applied to output of this function to obtain overall cross-validated performance score. A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

## Examples

```
data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]

# Find initial lambdas: fast CV per data block separately.
cvperblock2 <- fastCV2(XXblocks=XXmirmeth,Y=resp,kfold=10,fixedfolds = TRUE)
lambdas <- cvperblock2$lambdas

# Double cross-validation
## Not run:
perf <- doubleCV(penaltiesinit=lambdas,XXblocks=XXmirmeth,Y=resp,
score="loglik",outfold=10, infold=10, nrepeatout=1, nrepeatin=3, parallel=TRUE)

# Performance metrics
Scoring(perf$linpred,perf$true,score="auc",print=TRUE)
Scoring(perf$linpred,perf$true,score="brier",print=TRUE)
Scoring(perf$linpred,perf$true,score="loglik",print=TRUE)

## End(Not run)
```

---

fastCV2	<i>Fast cross-validation per data block</i>
---------	---------------------------------------------

---

### Description

Fast cross-validation for high-dimensional data. Finds optimal penalties separately per data block. Useful for initialization.

### Usage

```
fastCV2(XXblocks, Y, X1 = NULL, kfold = 10, intercept =
  ifelse(is(Y, "Surv"), FALSE, TRUE), parallel = FALSE, fixedfolds = TRUE,
  model = NULL, eps = 1e-10, reltol = 0.5, lambdamax= 10^6, traceCV=TRUE)
```

### Arguments

XXblocks	List of data frames or matrices, representing $b=1, \dots, B$ data blocks of dimensions $n \times p_b$ .
Y	Response vector: numeric, binary, factor or survival.
X1	Matrix. Dimension $n \times p_0$ , $p_0 < n$ , representing unpenalized covariates.
kfold	Integer. Desired fold.
intercept	Boolean. Should an intercept be included?
parallel	Boolean. Should computation be done in parallel? If TRUE, requires to run <a href="#">setupParallel</a> first.
fixedfolds	Boolean. Should fixed splits be used for reproducibility?
model	Character. Any of <code>c("linear", "logistic", "cox")</code> . Is inferred from Y when NULL.
eps	Scalar. Numerical bound for IWLS convergence.
reltol	Scalar. Relative tolerance for optimization method.
lambdamax	Numeric. Upperbound for lambda.
traceCV	Boolean. Should the CV results be traced and printed?

### Details

This function is basically a wrapper for applying [optLambdas](#) per data block separately using Brent optimization.

### Value

Numerical vector containing penalties optimized separately per data block. Useful for initialization.

### See Also

[optLambdas](#), [optLambdasWrap](#) which optimize the penalties jointly. A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

**Examples**

```

data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]

cvperblock2 <- fastCV2(XXblocks=XXmirmeth,Y=resp,kfold=10,fixedfolds = TRUE)
lambdas <- cvperblock2$lambdas

```

---

 IWLS Coxridge

*Iterative weighted least squares algorithm for Cox ridge regression.*


---

**Description**

Iterative weighted least squares algorithm for Cox ridge regression. Updates the weights and linear predictors until convergence.

**Usage**

```

IWLS Coxridge(XXT, Y, X1 = NULL, intercept = FALSE, eps = 1e-07, maxItr = 25,
trace = FALSE, E0 = NULL)

```

**Arguments**

XXT	Matrix. Dimensions $n \times n$ . Sample cross-product from penalized variables, usually computed by <a href="#">SigmaFromBlocks</a> .
Y	Response vector: class survival.
X1	Matrix. Dimension $n \times p_0$ , $p_0 < n$ , representing unpenalized covariates.
intercept	Boolean. Should an intercept be included?
eps	Scalar. Numerical bound for IWLS convergence.
maxItr	Integer. Maximum number of iterations used in IWLS.
trace	Boolean. Should the output of the IWLS algorithm be traced?
E0	Numerical vector or NULL. Optional initial values for linear predictor. Same length as Y. Usually NULL, which initializes linear predictor with 0.

**Details**

Usually, Cox ridge regression does not use an intercept, as this is part of the baseline hazard. The latter is estimated using the Breslow estimator. To keep the function computationally efficient it returns the linear predictors (which suffice for predictions), instead of parameter estimates. These may be obtained by applying the [betasout](#) function to the output of this function.

**Value**

List, containing:

etas	Numerical vector: Final linear predictors
Ypred	Predicted survival
convergence	Boolean: has IWLS converged?
nIt	Number of iterations
Hres	Auxiliary list object. Passed on to other functions
linearized	Linearized predictions
unpen	Boolean: are there any unpenalized covariates involved? Passed on to other functions
intercept	Boolean: Is an intercept included?
eta0	Numerical vector: Initial linear predictors
X1	Matrix: design matrix unpenalized variables

**References**

Mark A. van de Wiel, Mirrelijn van Nee, Armin Rauschenberger (2021). Fast cross-validation for high-dimensional ridge regression. *J Comp Graph Stat*

**See Also**

[IWLStridge](#) for linear and logistic ridge. [betasout](#) for obtaining parameter estimates. [predictIWLS](#) for predictions on new samples. A full demo and data are available from:  
<https://drive.google.com/open?id=1NUfeOtN8-KZ8A2HZzveG506nBwgW64e4>

**Examples**

```
data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]
lambdas <- c(100,1000)

# Create fake survival data
respsurv <- Surv(rexp(length(resp)),resp)

# Prepare fitting for the specified penalties.
XXT <- SigmaFromBlocks(XXmirmeth,penalties=lambdas)

# Fit. fit$etas contains the n linear predictors
fit <- IWLSCoxridge(XXT,Y=respsurv)
```

---

IWLSridge	<i>Iterative weighted least squares algorithm for linear and logistic ridge regression.</i>
-----------	---------------------------------------------------------------------------------------------

---

### Description

Iterative weighted least squares algorithm for linear and logistic ridge regression. Updates the weights and linear predictors until convergence.

### Usage

```
IWLSridge(XXT, Y, X1 = NULL, intercept = TRUE, frac1 = NULL, eps = 1e-07,
maxItr = 25, trace = FALSE, model = NULL, E0 = NULL)
```

### Arguments

XXT	Matrix. Dimensions nxn. Sample cross-product from penalized variables, usually computed by <a href="#">SigmaFromBlocks</a> .
Y	Response vector: numeric, binary, or two-class factor
X1	Matrix. Dimension $n \times p_0$ , $p_0 < n$ , representing unpenalized covariates.
intercept	Boolean. Should an intercept be included?
frac1	Scalar. Prior fraction of cases. Only relevant for <code>model="logistic"</code> .
eps	Scalar. Numerical bound for IWLS convergence.
maxItr	Integer. Maximum number of iterations used in IWLS.
trace	Boolean. Should the output of the IWLS algorithm be traced?
model	Character. Any of <code>c("linear", "logistic")</code> . Is inferred from Y when NULL. Note that the cox model for survival is covered by the function <a href="#">IWLScoxridge</a> .
E0	Numerical vector or NULL. Optional initial values for linear predictor. Same length as Y. Usually NULL, which initializes linear predictor with 0.

### Details

An (unpenalized) intercept is included by default. To keep the function computationally efficient it returns the linear predictors (which suffice for predictions), instead of parameter estimates. These may be obtained by applying the [betasout](#) function to the output of this function.

### Value

List, containing:

etas	Numerical vector: Final linear predictors
Ypred	Predicted survival
convergence	Boolean: has IWLS converged?
nIt	Number of iterations

Hres	Auxiliary list object. Passed on to other functions
linearized	Linearized predictions
unpen	Boolean: are there any unpenalized covariates involved? Passed on to other functions
intercept	Boolean: Is an intercept included?
eta0	Numerical vector: Initial linear predictors
X1	Matrix: design matrix unpenalized variables

## References

Mark A. van de Wiel, Mirrelijn van Nee, Armin Rauschenberger (2021). Fast cross-validation for high-dimensional ridge regression. *J Comp Graph Stat*

## See Also

[IWLScoxridge](#) for Cox ridge. [betasout](#) for obtaining parameter estimates. [predictIWLS](#) for predictions on new samples. A full demo and data are available from:  
<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

## Examples

```
data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]
lambdas <- c(100,1000)

# Prepare fitting for the specified penalties.
XXT <- SigmaFromBlocks(XXmirmeth,penalties=lambdas)

# Fit. fit$etas contains the n linear predictors
fit <- IWLSridge(XXT,Y=resp)
```

---

mgcv_lambda	<i>Maximum marginal likelihood score</i>
-------------	------------------------------------------

---

## Description

Computed maximum marginal likelihood score for given penalty parameters using mgcv.

## Usage

```
mgcv_lambda(penalties, XXblocks,Y, model=NULL, printscore=TRUE, pairing=NULL, sigmasq = 1,
  opt.sigma=ifelse(model=="linear",TRUE, FALSE))
```

**Arguments**

penalties	Numeric vector.
XXblocks	List of nxn matrices. Usually output of <a href="#">createXXblocks</a> .
Y	Response vector: numeric, binary, factor or survival.
model	Character. Any of c("linear", "logistic", "cox"). Is inferred from Y when NULL.
printscore	Boolean. Should the score be printed?
pairing	Numerical vector of length 3 or NULL when pairs are absent. Represents the indices (in XXblocks) of the two data blocks involved in pairing, plus the index of the paired block.
sigmasq	Default error variance.
opt.sigma	Boolean. Should the error variance be optimized as well? Only relevant for model="linear".

**Details**

See [gam](#) for details on how the marginal likelihood is computed.

**Value**

Numeric, marginal likelihood score for given penalties

**References**

Wood, S. N. (2011), Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models, *J. Roy. Statist. Soc., B* 73(1), 3-36.

**See Also**

[CVscore](#) for cross-validation alternative. A full demo and data are available from:  
<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

---

mlikCV

*Outer-loop cross-validation for estimating performance of marginal likelihood based multiridge*

---

**Description**

Outer-loop cross-validation for estimating performance of marginal likelihood based `multiridge`. Outer fold is for testing; penalty parameter tuning is performed by marginal likelihood estimation

**Usage**

```
mlikCV(penaltiesinit, XXblocks, Y, pairing = NULL, outfold = 5, nrepeatout = 1,
balance = TRUE, fixedfolds = TRUE, model = NULL, intercept =
ifelse(is(Y, "Surv"), FALSE, TRUE), reltol = 1e-04, trace = FALSE, optmethod1 = "SANN",
optmethod2 = ifelse(length(penaltiesinit) == 1, "Brent", "Nelder-Mead"),
maxItropt1 = 10, maxItropt2 = 25, parallel = FALSE, pref = NULL,
fixedpen = NULL, sigmasq = 1, opt.sigma=ifelse(model=="linear",TRUE, FALSE))
```

**Arguments**

penaltiesinit	Numeric vector. Initial values for penalty parameters. May be obtained from <a href="#">fastCV2</a> .
XXblocks	List of nxn matrices. Usually output of <a href="#">createXXblocks</a> .
Y	Response vector: numeric, binary, factor or survival.
pairing	Numerical vector of length 3 or NULL when pairs are absent. Represents the indices (in XXblocks) of the two data blocks involved in pairing, plus the index of the paired block.
outfold	Integer. Outer fold for test samples.
nrepeatout	Integer. Number of repeated splits for outer fold.
balance	Boolean. Should the splits be balanced in terms of response labels?
fixedfolds	Boolean. Should fixed splits be used for reproducibility?
intercept	Boolean. Should an intercept be included?
model	Character. Any of c("linear", "logistic", "cox"). Is inferred from Y when NULL.
trace	Boolean. Should the output of the IWLS algorithm be traced?
reltol	Scalar. Relative tolerance for optimization methods.
optmethod1	Character. First, global search method. Any of the methods c("Brent", "Nelder-Mead", "Sann") may be used, but simulated annealing by "Sann" is recommended to search a wide landscape. Other unconstrained methods offered by <a href="#">optim</a> may also be used, but have not been tested.
optmethod2	Character. Second, local search method. Any of the methods c("Brent", "Nelder-Mead", "Sann") may be used, but "Nelder-Mead" is generally recommended. Other unconstrained methods offered by <a href="#">optim</a> may also be used, but have not been tested.
maxItropt1	Integer. Maximum number of iterations for optmethod1.
maxItropt2	Integer. Maximum number of iterations for optmethod2.
parallel	Boolean. Should computation be done in parallel? If TRUE, requires to run <a href="#">setupParallel</a> first.
pref	Integer vector or NULL. Contains indices of data types in XXblocks that are preferential.
fixedpen	Integer vector or NULL. Contains indices of data types of which penalty is fixed to the corresponding value in penaltiesinit.
sigmasq	Default error variance.
opt.sigma	Boolean. Should the error variance be optimized as well? Only relevant for model="linear".

## Details

**WARNING:** this function may be very time-consuming. The number of evaluations may equal  $nrepeatout * outerfold * (maxItropt1 + maxItropt2)$ . Computing time may be estimated by multiplying computing time of `optLambdas_mgcvWrap` by  $nrepeatout * outerfold$ .

## Value

List with the following components:

sampleindex	Numerical vector: sample indices
true	True responses
linpred	Cross-validated linear predictors

## See Also

`optLambdas_mgcv`, `optLambdas_mgcvWrap` which optimize the penalties. `Scoring` which may applied to output of this function to obtain overall cross-validated performance score. `doubleCV` for double cross-validation counterpart. A full demo and data are available from: <https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

## Examples

```
data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]

# Find initial lambdas: fast CV per data block separately.
cvperblock2 <- fastCV2(XXblocks=XXmirmeth,Y=resp,kfold=10,fixedfolds = TRUE)
lambdas <- cvperblock2$lambdas

# Outer cross-validation, inner marginal likelihood optimization
## Not run:
perfmlik <- mlikCV(penaltiesinit=lambdas,XXblocks=XXmirmeth,Y=resp,outfold=10,
nrepeatout=1)

# Performance metrics
Scoring(perfmlik$linpred,perfmlik$true,score="auc",print=TRUE)
Scoring(perfmlik$linpred,perfmlik$true,score="brier",print=TRUE)
Scoring(perfmlik$linpred,perfmlik$true,score="loglik",print=TRUE)

## End(Not run)
```

---

optLambdas *Find optimal ridge penalties.*

---

### Description

Optimizes a cross-validated score w.r.t. ridge penalties for multiple data blocks.

### Usage

```
optLambdas(penaltiesinit = NULL, XXblocks, Y, X1 = NULL, pairing = NULL, folds,
  intercept = ifelse(is(Y, "Surv"), FALSE, TRUE), frac1 = NULL, score = "loglik",
  model = NULL, epsIWLS = 0.001, maxItrIWLS = 25, traceCV = TRUE, reltol = 1e-04,
  optmethod = ifelse(length(penaltiesinit) == 1, "Brent", "Nelder-Mead"), maxItropt = 500,
  save = FALSE, parallel = FALSE, fixedpen = NULL, fixedseed = TRUE)
```

### Arguments

penaltiesinit	Numeric vector. Initial values for penalty parameters. May be obtained from <a href="#">fastCV2</a> .
XXblocks	List of nxn matrices. Usually output of <a href="#">createXXblocks</a> .
Y	Response vector: numeric, binary, factor or survival.
X1	Matrix. Dimension $n \times p_0$ , $p_0 < n$ , representing unpenalized covariates
pairing	Numerical vector of length 3 or NULL when pairs are absent. Represents the indices (in XXblocks) of the two data blocks involved in pairing, plus the index of the paired block.
folds	List, containing the splits of the samples. Usually obtained by <a href="#">CVfolds</a>
intercept	Boolean. Should an intercept be included?
frac1	Scalar. Prior fraction of cases. Only relevant for model="logistic".
score	Character. See Details.
model	Character. Any of c("linear", "logistic", "cox"). Is inferred from Y when NULL.
epsIWLS	Scalar. Numerical bound for IWLS convergence.
maxItrIWLS	Integer. Maximum number of iterations used in IWLS.
traceCV	Boolean. Should the output of the IWLS algorithm be traced?
reltol	Scalar. Relative tolerance for optimization methods.
optmethod	Character. Optimization method. Any of the methods c("Brent", "Nelder-Mead", "Sann") may be used, but "Nelder-Mead" is generally recommended. Other unconstrained methods offered by <a href="#">optim</a> may also be used, but have not been tested.
maxItropt	Integer. Maximum number of iterations for optmethod.
save	Boolean. If TRUE appends the penalties and resulting CVscore to global variable allscores

parallel	Boolean. Should computation be done in parallel? If TRUE, requires to run <a href="#">setupParallel</a> first.
fixedpen	Integer vector or NULL. Contains indices of data types of which penalty is fixed to the corresponding value in <code>penaltiesinit</code> .
fixedseed	Boolean. Should the initialization be fixed? For reproducibility.

### Details

See [Scoring](#) for details on score. We highly recommend to use smooth scoring functions, in particular "loglik". For ranking-based criteria like auc and cindex we advise to use repeated CV (see [CVfolds](#)) to avoid ending up in any of the many local optima.

### Value

List, with components:

optres	Output of the optimizer
optpen	Vector with determined optimal penalties
allsc	Matrix with CV scores for all penalty parameter configurations used by the optimizer

### See Also

[optLambdasWrap](#) for i) (recommended) optimization in two steps: first global, then local; and ii) sequential optimization when some data types are preferred over others. [fastCV2](#) for initialization of penalties. A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

### Examples

```
data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]

# Find initial lambdas: fast CV per data block separately.
cvperblock2 <- fastCV2(XXblocks=XXmirmeth,Y=resp,kfold=10,fixedfolds = TRUE)
lambdas <- cvperblock2$lambdas

# Create (repeated) CV-splits of the data.
leftout <- CVfolds(Y=resp,kfold=10,nrepeat=3,fixedfolds = TRUE)

# One-pass optimization
# Increase the number of iterations for optimal results
jointlambdas <- optLambdas(penaltiesinit=lambdas, XXblocks=XXmirmeth,Y=resp,
folds=leftout,score="loglik",save=T,maxItropt=5)
```

---

optLambdasWrap      *Find optimal ridge penalties with sequential optimization.*

---

### Description

Sequentially optimizes a cross-validated score w.r.t. ridge penalties for multiple data blocks. Also implements preferential ridge, which allows to first optimize for the preferential data types.

### Usage

```
optLambdasWrap(penaltiesinit = NULL, XXblocks, Y, X1 = NULL, pairing = NULL,
  folds, intercept = ifelse(is(Y, "Surv"), FALSE, TRUE), frac1 = NULL,
  score = "loglik", model = NULL, epsIWLS = 0.001, maxItrIWLS = 25,
  traceCV = TRUE, reltol = 1e-04, optmethod1 = "SANN", optmethod2 =
  ifelse(length(penaltiesinit) == 1, "Brent", "Nelder-Mead"), maxItropt1 = 10,
  maxItropt2 = 25, save = FALSE, parallel = FALSE, pref = NULL, fixedpen = NULL)
```

### Arguments

penaltiesinit	Numeric vector. Initial values for penalty parameters. May be obtained from <a href="#">fastCV2</a> .
XXblocks	List of nxn matrices. Usually output of <a href="#">createXXblocks</a> .
Y	Response vector: numeric, binary, factor or survival.
X1	Matrix. Dimension $n \times p_0$ , $p_0 < n$ , representing unpenalized covariates
pairing	Numerical vector of length 3 or NULL when pairs are absent. Represents the indices (in XXblocks) of the two data blocks involved in pairing, plus the index of the paired block.
folds	List, containing the splits of the samples. Usually obtained by <a href="#">CVfolds</a>
intercept	Boolean. Should an intercept be included?
frac1	Scalar. Prior fraction of cases. Only relevant for model="logistic".
score	Character. See Details.
model	Character. Any of c("linear", "logistic", "cox"). Is inferred from Y when NULL.
epsIWLS	Scalar. Numerical bound for IWLS convergence.
maxItrIWLS	Integer. Maximum number of iterations used in IWLS.
traceCV	Boolean. Should the output of the IWLS algorithm be traced?
reltol	Scalar. Relative tolerance for optimization methods.
optmethod1	Character. First, global search method. Any of the methods c("Brent", "Nelder-Mead", "Sann") may be used, but simulated annealing by "Sann" is recommended to search a wide landscape. Other unconstrained methods offered by <a href="#">optim</a> may also be used, but have not been tested.

optmethod2	Character. Second, local search method. Any of the methods c("Brent", "Nelder-Mead", "Sann") may be used, but "Nelder-Mead" is generally recommended. Other unconstrained methods offered by <code>optim</code> may also be used, but have not been tested.
maxItropt1	Integer. Maximum number of iterations for optmethod1.
maxItropt2	Integer. Maximum number of iterations for optmethod2.
save	Boolean. If TRUE appends the penalties and resulting CVscore to global variable allscores
parallel	Boolean. Should computation be done in parallel? If TRUE, requires to run <code>setupParallel</code> first.
pref	Integer vector or NULL. Contains indices of data types in XXblocks that are preferential.
fixedpen	Integer vector or NULL. Contains indices of data types of which penalty is fixed to the corresponding value in penaltiesinit.

### Details

As opposed to `optLambdas` this function first searches globally, then locally. Hence, more time-consuming, but better guarded against multiple local optima.

See `Scoring` for details on score. We highly recommend to use smooth scoring functions, in particular "loglik". For ranking-based criteria like "auc" and "cindex" we advise to use repeated CV (see `CVfolds`) to avoid ending up in any of the many local optima.

### Value

List, with components:

res	Outputs of all optimizers used
lambdas	List of penalties found by the optimizers
optpen	Numerical vector with final, optimal penalties

### See Also

`optLambdas` for one-pass optimization. `fastCV2` for initialization of penalties. A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

### Examples

```
data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]

# Find initial lambdas: fast CV per data block separately.
cvperblock2 <- fastCV2(XXblocks=XXmirmeth,Y=resp,kfold=10,fixedfolds = TRUE)
lambdas <- cvperblock2$lambdas

# Create (repeated) CV-splits of the data.
```

```
leftout <- CVfolds(Y=resp,kfold=10,nrepeat=3,fixedfolds = TRUE)

# Optimizes cross-validate criterion (default: log-lik)
# Increase the number of iterations for optimal results
jointlambdas <- optLambdasWrap(penaltiesinit=lambdas, XXblocks=XXmirmeth,Y=resp,
folds=leftout,score="loglik",save=T,maxItropt1=5, maxItropt2=5)
```

---

optLambdas\_mgcv

*Find optimal ridge penalties with maximum marginal likelihood*


---

## Description

Optimizes a marginal likelihood score w.r.t. ridge penalties for multiple data blocks.

## Usage

```
optLambdas_mgcv(penaltiesinit=NULL, XXblocks,Y, pairing=NULL, model=NULL, reltol=1e-4,
optmethod=ifelse(length(penaltiesinit)==1,"Brent", "Nelder-Mead"),maxItropt=500,
tracescore=TRUE, fixedpen=NULL, fixedseed =TRUE, sigmasq = 1,
opt.sigma=ifelse(model=="linear",TRUE, FALSE))
```

## Arguments

penaltiesinit	Numeric vector. Initial values for penaltyparameters. May be obtained from <a href="#">fastCV2</a> .
XXblocks	List of nxn matrices. Usually output of <a href="#">createXXblocks</a> .
Y	Response vector: numeric, binary, factor or survival.
pairing	Numerical vector of length 3 or NULL when pairs are absent. Represents the indices (in XXblocks) of the two data blocks involved in pairing, plus the index of the paired block.
model	Character. Any of c("linear", "logistic", "cox"). Is inferred from Y when NULL.
reltol	Scalar. Relative tolerance for optimization methods.
optmethod	Character. Optimization method. Any of the methods c("Brent", "Nelder-Mead", "Sann") may be used, but "Nelder-Mead" is generally recommended. Other unconstrained methods offered by <a href="#">optim</a> may also be used, but have not been tested.
maxItropt	Integer. Maximum number of iterations for optmethod.
tracescore	Boolean. Should the output of the scores be traced?
fixedpen	Integer vector or NULL. Contains indices of data types of which penalty is fixed to the corresponding value in penaltiesinit.
fixedseed	Boolean. Should the initialization be fixed? For reproducibility.
sigmasq	Default error variance.
opt.sigma	Boolean. Should the error variance be optimized as well? Only relevant for model="linear".

**Details**

See [gam](#) for details on how the marginal likelihood is computed.

**Value**

List, with components:

optres	Output of the optimizer
optpen	Vector with determined optimal penalties
allsc	Matrix with marginal likelihood scores for all penalty parameter configurations used by the optimizer

**See Also**

[optLambdas\\_mgcvWrap](#) for i) (recommended) optimization in two steps: first global, then local; and ii) sequential optimization when some data types are preferred over others. A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

**Examples**

```
data(dataXXmirmeth)
resp <- dataXXmirmeth[[1]]
XXmirmeth <- dataXXmirmeth[[2]]

# Find initial lambdas: fast CV per data block separately.
cvperblock2 <- fastCV2(XXblocks=XXmirmeth,Y=resp,kfold=10,fixedfolds = TRUE)
lambdas <- cvperblock2$lambdas

# Create (repeated) CV-splits of the data.
leftout <- CVfolds(Y=resp,kfold=10,nrepeat=3,fixedfolds = TRUE)

# Compute cross-validated score for initial lambdas
CVscore(penalties=lambdas, XXblocks=XXmirmeth,Y=resp,folds=leftout,
score="loglik")

# Optimize by using marginal likelihood criterion
jointlambdas2 <- optLambdas_mgcvWrap(penaltiesinit=lambdas, XXblocks=XXmirmeth,
Y=resp)

# Optimal lambdas
optlambdas <- jointlambdas2$optpen
```

---

optLambdas\_mgcvWrap    *Find optimal ridge penalties with sequential optimization.*

---

### Description

Sequentially optimizes a marginal likelihood score w.r.t. ridge penalties for multiple data blocks.

### Usage

```
optLambdas_mgcvWrap(penaltiesinit=NULL, XXblocks,Y, pairing=NULL, model=NULL, reltol=1e-4,
  optmethod1= "SANN", optmethod2 =ifelse(length(penaltiesinit)==1,"Brent", "Nelder-Mead"),
  maxItropt1=10,maxItropt2=25, tracescore=TRUE, fixedseed =TRUE, pref=NULL, fixedpen=NULL,
  sigmasq = 1, opt.sigma=ifelse(model=="linear",TRUE, FALSE))
```

### Arguments

penaltiesinit	Numeric vector. Initial values for penaltyparameters. May be obtained from <a href="#">fastCV2</a> .
XXblocks	List of nxn matrices. Usually output of <a href="#">createXXblocks</a> .
Y	Response vector: numeric, binary, factor or survival.
pairing	Numerical vector of length 3 or NULL when pairs are absent. Represents the indices (in XXblocks) of the two data blocks involved in pairing, plus the index of the paired block.
model	Character. Any of c("linear", "logistic", "cox"). Is inferred from Y when NULL.
reltol	Scalar. Relative tolerance for optimization methods.
optmethod1	Character. First, global search method. Any of the methods c("Brent", "Nelder-Mead", "Sann") may be used, but simulated annealing by "Sann" is recommended to search a wide landscape. Other unconstrained methods offered by <a href="#">optim</a> may also be used, but have not been tested.
optmethod2	Character. Second, local search method. Any of the methods c("Brent", "Nelder-Mead", "Sann") may be used, but "Nelder-Mead" is generally recommended. Other unconstrained methods offered by <a href="#">optim</a> may also be used, but have not been tested.
maxItropt1	Integer. Maximum number of iterations for optmethod1.
maxItropt2	Integer. Maximum number of iterations for optmethod2.
tracescore	Boolean. Should the output of the scores be traced?
fixedseed	Boolean. Should the initialization be fixed? For reproducibility.
pref	Integer vector or NULL. Contains indices of data types in XXblocks that are preferential.
fixedpen	Integer vector or NULL. Contains indices of data types of which penalty is fixed to the corresponding value in penaltiesinit.
sigmasq	Default error variance.
opt.sigma	Boolean. Should the error variance be optimized as well? Only relevant for model="linear".

**Details**

As opposed to `optLambdas_mgcv` this function first searches globally, then locally. Hence, more time-consuming, but better guarded against multiple local optima. See `gam` for details on how the marginal likelihood is computed.

**Value**

List, with components:

<code>res</code>	Outputs of all optimizers used
<code>lambdas</code>	List of penalties found by the optimizers
<code>optpen</code>	Numerical vector with final, optimal penalties

**References**

Wood, S. N. (2011), Fast stable restricted maximum likelihood and marginal likelihood estimation of semiparametric generalized linear models, *J. Roy. Statist. Soc., B* 73(1), 3-36.

**See Also**

`optLambdas_mgcv` for one-pass optimization. A full demo and data are available from:  
<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

---

predictIWLS	<i>Predictions from ridge fits</i>
-------------	------------------------------------

---

**Description**

Produces predictions from ridge fits for new data.

**Usage**

```
predictIWLS(IWLSfit, X1new = NULL, Sigmanew)
```

**Arguments**

<code>IWLSfit</code>	List, containing fits from either <code>IWLSridge</code> (linear, logistic ridge) or <code>IWLScoxridge</code>
<code>X1new</code>	Matrix. Dimension $n_{\text{new}} \times p_{-\theta}$ , representing unpenalized covariates for new data.
<code>Sigmanew</code>	Matrix. Dimensions $n_{\text{new}} \times n$ . Sample cross-product from penalized variables, usually computed by first applying <code>createXXblocks</code> and then <code>SigmaFromBlocks</code> .

**Details**

Predictions rely purely on the linear predictors, and do not require producing the parameter vector.

**Value**

Numerical vector of linear predictor for the test samples.

**See Also**

[IWLsridge](#) ([IWLSCoxridge](#)) for fitting linear and logistic ridge (Cox ridge). [betasout](#) for obtaining parameter estimates. [Scoring](#) to evaluate the predictions. A full demo and data are available from: <https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

**Examples**

```
#Example below shows how to create the input argument Sigmanew (for simulated data)
#Simulate
Xb11 <- matrix(rnorm(1000),nrow=10)
Xb12 <- matrix(rnorm(2000),nrow=10)
Xb11new <- matrix(rnorm(200),nrow=2)
Xb12new <- matrix(rnorm(400),nrow=2)

#check whether dimensions are correct
nrow(Xb11)==nrow(Xb11new)
nrow(Xb12)==nrow(Xb12new)
ncol(Xb11)==nrow(Xb12)
ncol(Xb11new)==ncol(Xb12new)

#create cross-product
XXbl <- createXXblocks(list(Xb11,Xb12),list(Xb11new,Xb12new))

#suppose penalties for two data types equal 5,10, respectively
Sigmanew <- SigmaFromBlocks(XXbl,c(5,10))

#check dimensions (should be nnew x n)
dim(Sigmanew)
```

---

Scoring

*Evaluate predictions*

---

**Description**

Evaluates predictions by a score suitable for the corresponding response

**Usage**

```
Scoring(lp, Y, model = NULL, score = ifelse(model == "linear", "mse", "loglik"),
  print = TRUE)
```

**Arguments**

lp	Numerical vector. Linear predictor.
Y	Response vector: numeric, binary, factor or survival.
score	Character. See Details.
model	Character. Any of c("linear", "logistic", "cox"). Is inferred from Y when NULL.
print	Boolean. Should the score be printed on screen.

**Details**

Several scores are allowed, depending on the type of output. For model = "linear", score equals any of c("loglik", "mse", "abserror", "cor", "kendall", "spearman"), denoting CV-ed log-likelihood, mean-squared error, mean absolute error, Pearson (Kendall, Spearman) correlation with response. For model = "logistic", score equals any of c("loglik", "auc", "brier"), denoting CV-ed log-likelihood, area-under-the-ROC-curve, and brier score a.k.a. MSE. For model = "cox", score equals any of c("loglik", "cindex"), denoting CV-ed log-likelihood, and c-index.

**Value**

Numerical value.

**See Also**

[CVscore](#) for obtaining the cross-validated score (for given penalties), and [doubleCV](#) to obtain doubly cross-validated linear predictors to which `Scoring` can be applied to estimated predictive performance by double cross-validation. A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

---

setupParallel	<i>Setting up parallel computing</i>
---------------	--------------------------------------

---

**Description**

This function sets up parallel computing by the package `snowfall`.

**Usage**

```
setupParallel(ncpus = 2, sourcefile = NULL, sourcelibraries =
c("multiridge", "survival", "pROC", "risksetROC"))
```

**Arguments**

ncpus	Integer. Number of cpus to use. Should be $\geq 2$ .
sourcefile	Character. Additional source files to be loaded in parallel. Only required when parallel computing is also desired for functions not available in <code>multiridge</code> .
sourcelibraries	Character vector. Libraries to be loaded in parallel. Defaults to the libraries <code>multiridge</code> depends on.

**Details**

Parallel computing is available for several functions that rely on cross-validation. If double CV is used, parallel computing is applied to the outer loop, to optimize efficiency.

**Value**

No return value, called for side effects

**See Also**

Snowfall package for further documentation on parallel computing. A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

**Examples**

```
## Not run:
setupParallel(ncpus=4)

## End(Not run)
```

---

SigmaFromBlocks	<i>Create penalized sample cross-product matrix</i>
-----------------	-----------------------------------------------------

---

**Description**

Creates penalized sample cross-product matrix, dimension  $n \times n$ .

**Usage**

```
SigmaFromBlocks(XXblocks, penalties, pairing = NULL)
```

**Arguments**

XXblocks	List of $n \times n$ matrices. Usually output of <a href="#">createXXblocks</a> .
penalties	Numeric vector, representing penalty parameters.
pairing	Numerical vector of length 3 or NULL when pairs are absent. Represents the indices (in XXblocks) of the two data blocks involved in pairing, plus the index of the paired block.

**Value**

Matrix of size  $n \times n$ .

**See Also**

A full demo and data are available from:

<https://drive.google.com/open?id=1NUfe0tN8-KZ8A2HZzveG506nBwgW64e4>

**Examples**

```
#Example
#Simulate
Xb11 <- matrix(rnorm(1000),nrow=10)
Xb12 <- matrix(rnorm(2000),nrow=10)

#check whether dimensions are correct
ncol(Xb11)==nrow(Xb12)

#create cross-product
XXb1 <- createXXblocks(list(Xb11,Xb12))

#suppose penalties for two data types equal 5,10, respectively
Sigma <- SigmaFromBlocks(XXb1,c(5,10))

#check dimensions (should be n x n)
dim(Sigma)
```

# Index

- \* **datasets**
  - dataXXmirmeth, [11](#)
- \* **package**
  - multiridge-package, [2](#)
- augment, [4](#)
- betasout, [3](#), [5](#), [16–19](#), [31](#)
- createXblocks, [6](#), [7](#), [8](#)
- createXXblocks, [3](#), [5](#), [7](#), [8](#), [10](#), [12](#), [20](#), [21](#), [23](#),  
[25](#), [27](#), [29](#), [30](#), [33](#)
- CVfolds, [9](#), [10](#), [23–26](#)
- CVscore, [3](#), [10](#), [20](#), [32](#)
- dataXXmirmeth, [3](#), [11](#)
- doubleCV, [3](#), [11](#), [12](#), [22](#), [32](#)
- fastCV2, [3](#), [12](#), [15](#), [21](#), [23–27](#), [29](#)
- gam, [20](#), [28](#), [30](#)
- IWLSCoxridge, [3](#), [6](#), [16](#), [18](#), [19](#), [30](#), [31](#)
- IWLSridge, [3](#), [6](#), [17](#), [18](#), [30](#), [31](#)
- mgcv\_lambda, [19](#)
- mlikCV, [3](#), [20](#)
- multiridge (multiridge-package), [2](#)
- multiridge-package, [2](#)
- optim, [13](#), [21](#), [23](#), [25–27](#), [29](#)
- optLambdas, [14](#), [15](#), [23](#), [26](#)
- optLambdas\_mgcv, [22](#), [27](#), [30](#)
- optLambdas\_mgcvWrap, [3](#), [22](#), [28](#), [29](#)
- optLambdasWrap, [3](#), [14](#), [15](#), [24](#), [25](#)
- predictIWLS, [3](#), [17](#), [19](#), [30](#)
- Scoring, [10](#), [14](#), [22](#), [24](#), [26](#), [31](#), [31](#)
- setupParallel, [3](#), [10](#), [13](#), [15](#), [21](#), [24](#), [26](#), [32](#)
- SigmaFromBlocks, [3](#), [16](#), [18](#), [30](#), [33](#)