Package ‘pengls’

July 10, 2025

Type Package

Title Fit Penalised Generalised Least Squares models

Version 1.14.0

Description Combine generalised least squares methodology from the nlme package for dealing with autocorrelation with penalised least squares methods from the glmnet package to deal with high dimensionality. This pengls packages glues them together through an iterative loop. The resulting method is applicable to high dimensional datasets that exhibit autocorrelation, such as spatial or temporal data.

License GPL-2

Encoding UTF-8

RoxygenNote 7.2.1

Imports glmnet, nlme, stats, BiocParallel

Suggests knitr,rmarkdown,testthat

VignetteBuilder knitr

Depends R (>= 4.2.0)

biocViews Transcriptomics, Regression, TimeCourse, Spatial

BugReports `https://github.com/sthawinke/pengls`

git_url https://git.bioconductor.org/packages/pengls

git_branch RELEASE_3_21

git_last_commit 784f036

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-07-09

Author Stijn Hawinkel [cre, aut] (ORCID: <`https://orcid.org/0000-0002-4501-5180`>)

Maintainer Stijn Hawinkel <`stijn.hawinkel@psb.ugent.be`>

Contents

`coef.cv.pengls` *Extract coefficients from a cv.pengls model*

Description

Extract coefficients from a cv.pengls model

Usage

```
## S3 method for class 'cv.pengls'
coef(object, which = "lambda.1se", ...)
```

Arguments

`object`	A cv.pengls object
`which`	a character string, for which lambda should coefficients be returned
`...`	further arguments, currently ignored

Value

The vector of coefficients

`coef.pengls` *Extract coefficients from a pengls model*

Description

Extract coefficients from a pengls model

Usage

```
## S3 method for class 'pengls'
coef(object, ...)
```

Arguments

`object`	A pengls object
`...`	further arguments, currently ignored

Value

The vector of coefficients

`cv.pengls` *Peform cross-validation pengls*

Description

Peform cross-validation pengls

Usage

```
cv.pengls(
  data,
  glsSt,
  xNames,
  outVar,
  corMat,
  nfolds,
  foldid,
  scale = FALSE,
  center = FALSE,
  cvType = "blocked",
  lambdas,
  transFun = "identity",
  transFunArgs = list(),
  loss = c("R2", "MSE"),
  ...
)
```

Arguments

`data`	A data matrix or data frame
`glsSt`	a covariance structure, as supplied to nlme::gls as "correlation"
`xNames`	names of the regressors in data
`outVar`	name of the outcome variable in data
`corMat`	a starting value for the correlation matrix. Taken to be a diagonal matrix if missing
`nfolds`	an integer, the number of folds used in cv.glmnet to find lambda
`foldid`	An optional vector deffining the fold
`scale`, `center`	booleans, should regressors be scaled to zero mean and variance 1? Defaults to TRUE
`cvType`	A character vector defining the type of cross-validation. Either "random" or "blocked", ignored if foldid is provided
`lambdas`	an optional lambda sequence
`transFun`	a transformation function to apply to predictions and outcome in the cross-validation
`transFunArgs`	Additional arguments passed onto transFun
`loss`	a character vector, currently either 'R2' or 'MSE' indicating the loss function (although R2 is not a proper loss...)
`...`	passed onto glmnet::glmnet

Value

A list with components

`lambda`	The series of lambdas
`cvm`	The vector of mean R2's
`cvsd`	The standard error of R2 at the maximum
`cvOpt`	The R2 according to the 1 standard error rule
`coefs`	The matrix of coefficients for every lambda value
`bestFit`	The best fitting pengls model according to the 1 standard error rule
`lambda.min`	Lambda value with maximal R2
`lambda.1se`	Smallest lambda value within 1 standard error from the maximum
`foldid`	The folds
`glsSt`	The nlme correlation object
`loss`	The loss function used

Examples

```
library(nlme)
library(BiocParallel)
n <- 20 #Sample size
p <- 50 #Number of features
g <- 10 #Size of the grid
#Generate grid
Grid <- expand.grid("x" = seq_len(g), "y" = seq_len(g))
# Sample points from grid without replacement
GridSample <- Grid[sample(nrow(Grid), n, replace = FALSE),]
#Generate outcome and regressors
b <- matrix(rnorm(p*n), n , p)
a <- rnorm(n, mean = b %*% rbinom(p, size = 1, p = 0.2)) #20% signal
#Compile to a matrix
df <- data.frame("a" = a, "b" = b, GridSample)
# Define the correlation structure (see ?nlme::gls), with initial nugget 0.5 and range 5
corStruct = corGaus(form = ~ x + y, nugget = TRUE,
value = c("range" = 5, "nugget" = 0.5))
#Fit the pengls model, for simplicity for a simple lambda
register(MulticoreParam(3)) #Prepare multithereading
penglsFitCV = cv.pengls(data = df, outVar = "a", xNames = grep(names(df),
pattern = "b", value = TRUE),
glsSt = corStruct, nfolds = 5)
penglsFitCV$lambda.1se #Lambda for 1 standard error rule
penglsFitCV$cvOpt #Corresponding R2
coef(penglsFitCV)
penglsFitCV$foldid #The folds used
#With MSE as loss function
penglsFitCVmse = cv.pengls(data = df, outVar = "a",
xNames = grep(names(df), pattern = "b", value =TRUE),
glsSt = corStruct, nfolds = 5, loss = "MSE")
penglsFitCVmse$lambda.1se #Lambda for 1 standard error rule
penglsFitCVmse$cvOpt #Corresponding MSE
coef(penglsFitCVmse)
predict(penglsFitCVmse)
```

`getCorMat` *Get the (square root of the inverse of the) correlation matrix*

Description

Get the (square root of the inverse of the) correlation matrix

Usage

```
getCorMat(data, glsSt, Coef = c(coef(glsSt)), control, outVar)
```

Arguments

`data`	The data frame
`glsSt`	The correlation object for gls
`Coef`	optional vector of coefficients to glsSt
`control`	the list of control arguments for gls
`outVar`	the name of the outcome variable

Value

A list with components

`corMat`	The square root of the inverse correlation matrix
`Coef`	The coefficients of the correlation object

`getLoss` *Calculate the loss given predicted and observed values*

Description

Calculate the loss given predicted and observed values

Usage

```
getLoss(preds, obs, loss)
```

Arguments

`preds`	Matrix of predicted values
`obs`	vector of observed values
`loss`	a character vector indicating the loss type, see ?cv.pengls

Value

the evaluated loss

`makeFolds` *Divide observations into folds*

Description

Divide observations into folds

Usage

```
makeFolds(nfolds, data, cvType, coords)
```

Arguments

`nfolds`	The number of folds
`data`	the dataset
`cvType`	a character vector, indicating the type of cross-validation required, either blocked or random
`coords`	the names of the coordinates in data

Value

the vector of folds

Examples

```
nfolds <- 10
data <- expand.grid("x" = seq_len(10), "y" = seq_len(10))
randomFolds <- makeFolds(nfolds = nfolds, data, "random", c("x", "y"))
blockedFolds <- makeFolds(nfolds = nfolds, data, "blocked", c("x", "y"))
```

`pengls` *Iterative estimation of penalised generalised least squares*

Description

Iterative estimation of penalised generalised least squares

Usage

```
pengls(
  data,
  glsSt,
  xNames,
  outVar,
  corMat,
  lambda,
  foldid,
  maxIter = 30,
  tol = 0.05,
  verbose = FALSE,
  scale = FALSE,
  center = FALSE,
 optControl = lmeControl(opt = "optim", maxIter = 500, msVerbose = verbose, msMaxIter =
    500, niterEM = 1000, msMaxEval = 1000),
  nfolds = 10,
  penalty.factor = c(0, rep(1, length(xNames))),
  ...
)
```

Arguments

`data`	A data matrix or data frame
`glsSt`	a covariance structure, as supplied to nlme::gls as "correlation"
`xNames`	names of the regressors in data
`outVar`	name of the outcome variable in data
`corMat`	a starting value for the correlation matrix. Taken to be a diagonal matrix if missing
`lambda`	The penalty value for glmnet. If missing, the optimal value of vanilla glmnet without autocorrelation component is used
`foldid`	An optional vector deffining the fold
`maxIter`	maximum number of iterations between glmnet and gls
`tol`	A convergence tolerance
`verbose`	a boolean, should output be printed?
`scale, center`	booleans, should regressors be scaled to zero mean and variance 1? Defaults to TRUE
`optControl`	control arguments, passed onto nlme::gls' control argument
`nfolds`	an integer, the number of folds used in cv.glmnet to find lambda
`penalty.factor`	passed onto glmnet:glmnet. The first entry is zero by default for the intercept, which is not shrunk
`...`	passed onto glmnet::glmnet

Value

A list with components

`glmnet`	The glmnet fit, which can be manipulated as such
`gls`	A list with info on the estimated correlation matrix
`iter`	The iterations needed
`conv`	A boolean, indicating whether the iteration between mean model and covariance estimation converged
`xNames`, `data`, `glsSt`, `outVar`	As provided
`lambda`	The lambda penalty parameter used

See Also

cv.pengls

Examples

```
### Example 1: spatial data
# Define the dimensions of the data
library(nlme)
n <- 50 #Sample size
p <- 100 #Number of features
g <- 10 #Size of the grid
#Generate grid
Grid <- expand.grid("x" = seq_len(g), "y" = seq_len(g))
# Sample points from grid without replacement
GridSample <- Grid[sample(nrow(Grid), n, replace = FALSE),]
#Generate outcome and regressors
b <- matrix(rnorm(p*n), n , p)
a <- rnorm(n, mean = b %*% rbinom(p, size = 1, p = 0.2)) #20% signal
#Compile to a matrix
df <- data.frame("a" = a, "b" = b, GridSample)
# Define the correlation structure (see ?nlme::gls), with initial nugget 0.5 and range 5
corStruct <- corGaus(form = ~ x + y, nugget = TRUE, value = c("range" = 5, "nugget" = 0.5))
#Fit the pengls model, for simplicity for a simple lambda
penglsFit <- pengls(data = df, outVar = "a", xNames = grep(names(df), pattern = "b", value =TRUE),
glsSt = corStruct, nfolds = 5)

### Example 2: timecourse data
dfTime <- data.frame("a" = a, "b" = b, "t" = seq_len(n))
dfTime$a[-1] = dfTime$a[-n]*0.25 #Some temporal signal
corStructTime <- corAR1(form = ~ t, value = 0.5)
penglsFitTime <- pengls(data = dfTime, outVar = "a",
xNames = grep(names(dfTime), pattern = "b", value =TRUE),
glsSt = corStructTime, nfolds = 5)
```

`predict.cv.pengls` *Make predictions from a cv.pengls model*

Description

Make predictions from a cv.pengls model

Usage

```
## S3 method for class 'cv.pengls'
predict(object, ...)
```

Arguments

`object`	A cv.pengls object
`...`	further arguments, currently ignored

Value

A vector with predicted values

`predict.pengls` *Make predictions from a pengls model*

Description

Make predictions from a pengls model

Usage

```
## S3 method for class 'pengls'
predict(object, newx, ...)
```

Arguments

`object`	A pengls object
`newx`	The test data
`...`	further arguments, currently ignored

Value

A vector with predicted values

`print.cv.pengls` *Print a summary of a cv.pengls model*

Description

Print a summary of a cv.pengls model

Usage

```
## S3 method for class 'cv.pengls'
print(x, ...)
```

Arguments

`x`	A cv.pengls object
`...`	further arguments, currently ignored

Value

Prints output to console

`print.pengls` *Print a summary of a pengls model*

Description

Print a summary of a pengls model

Usage

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

Arguments

`x`	A pengls object
`...`	further arguments, currently ignored

Value

Prints output to console

Index