

# Package ‘pengls’

September 19, 2024

**Type** Package

**Title** Fit Penalised Generalised Least Squares models

**Version** 1.10.0

**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\_19

**git\_last\_commit** df9a9b6

**git\_last\_commit\_date** 2024-04-30

**Repository** Bioconductor 3.19

**Date/Publication** 2024-09-18

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

**Maintainer** Stijn Hawinkel <[stijn.hawinkel@psb.ugent.be](mailto:stijn.hawinkel@psb.ugent.be)>

Contents

coef.cv.pengls . . . . .	2
coef.pengls . . . . .	3
cv.pengls . . . . .	3
getCorMat . . . . .	5
getLoss . . . . .	6
makeFolds . . . . .	7
pengls . . . . .	7
predict.cv.pengls . . . . .	10
predict.pengls . . . . .	10
print.cv.pengls . . . . .	11
print.pengls . . . . .	11
<b>Index</b>	<b>12</b>

---

coef.cv.pengls	<i>Extract coefficients from a cv.pengls model</i>
----------------	--

---

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	<i>Extract coefficients from a pengls model</i>
-------------	---

---

**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	<i>Peform cross-validation pengls</i>
-----------	---------------------------------------

---

**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**

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

**Value**

A list with components

<code>lambda</code>	The series of <code>lambdas</code>
<code>cvm</code>	The vector of mean R2's
<code>cvsd</code>	The standard error of R2 at the maximum
<code>cvOpt</code>	The R2 according to the 1 standard error rule
<code>coefs</code>	The matrix of coefficients for every lambda value
<code>bestFit</code>	The best fitting pengls model according to the 1 standard error rule
<code>lambda.min</code>	Lambda value with maximal R2
<code>lambda.1se</code>	Smallest lambda value within 1 standard error from the maximum
<code>foldid</code>	The folds
<code>glsSt</code>	The <code>nlme</code> correlation object
<code>loss</code>	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 multithreading
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	<i>Calculate the loss given predicted and observed values</i>
---------	---

---

**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	<i>Divide observations into folds</i>
-----------	---------------------------------------

---

**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	<i>Iterative estimation of penalised generalised least squares</i>
--------	--

---

**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**

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



**Value**

A list with components

<code>glmnet</code>	The glmnet fit, which can be manipulated as such
<code>gls</code>	A list with info on the estimated correlation matrix
<code>iter</code>	The iterations needed
<code>conv</code>	A boolean, indicating whether the iteration between mean model and covariance estimation converged
<code>xNames, data, glsSt, outVar</code>	As provided
<code>lambda</code>	The lambda penalty paraneter 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	<i>Make predictions from a cv.pengls model</i>
-------------------	--

---

**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	<i>Make predictions from a pengls model</i>
----------------	---

---

**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	<i>Print a summary of a cv.pengls model</i>
-----------------	---

---

**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	<i>Print a summary of a pengls model</i>
--------------	--

---

**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

`coef.cv.pengls`, [2](#)  
`coef.pengls`, [3](#)  
`cv.pengls`, [3](#)  
  
`getCorMat`, [5](#)  
`getLoss`, [6](#)  
  
`makeFolds`, [7](#)  
  
`pengls`, [7](#)  
`predict.cv.pengls`, [10](#)  
`predict.pengls`, [10](#)  
`print.cv.pengls`, [11](#)  
`print.pengls`, [11](#)