Package ‘mdpeer’

May 30, 2017

Title Graph-Constrained Regression with Enhanced Regularization
Parameters Selection

Version 1.0.1

Author Marta Karas [aut, cre], Damian Brzyski [ctb], Jaroslaw Harezlak [ctb]

Maintainer Marta Karas <marta.karass@gmail.com>

Description Provides graph-constrained regression methods in which
regularization parameters are selected automatically via estimation of
equivalent Linear Mixed Model formulation. 'riPEER' (ridgified Partially
Empirical Eigenvectors for Regression) method employs a penalty term being
a linear combination of graph-originated and ridge-originated penalty terms,
whose two regularization parameters are ML estimators from corresponding
Linear Mixed Model solution; a graph-originated penalty term allows imposing
similarity between coefficients based on graph information given whereas
additional ridge-originated penalty term facilitates parameters estimation:
it reduces computational issues arising from singularity in a graph-originated
penalty matrix and yields plausible results in situations when graph information
is not informative. 'riPEERc' (ridgified Partially Empirical Eigenvectors
for Regression with constant) method utilizes addition of a diagonal matrix
multiplied by a predefined (small) scalar to handle the non-invertibility of
a graph Laplacian matrix. 'vrPEER' (variable reduced PEER) method performs
variable-reduction procedure to handle the non-invertibility of a graph
Laplacian matrix.

Depends R (>= 3.3.3)
Imports reshape2, ggplot2, nlme, boot, nloptr, rootSolve, psych,
magic, glmnet

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no
**R topics documented:**

Adj2Lap ......................................................... 2
L2L.normalized .............................................. 3
mdpeer .......................................................... 3
riPEER .......................................................... 4
riPEERc ......................................................... 7
vizu.mat ....................................................... 9
vizu.mat.factor .............................................. 12
vrPEER ........................................................ 14

**Index**

| Adj2Lap | Compute graph Laplacian matrix from graph adjacency matrix |

**Description**

Compute graph Laplacian matrix from graph adjacency matrix

**Usage**

Adj2Lap(adj)

**Arguments**

**adj**

graph adjacency matrix (squared symmetric matrix)

**Value**

graph Laplacian matrix

**Examples**

# Define exemplary adjacency matrix
p1 <- 10
p2 <- 40
p <- p1 + p2
A <- matrix(rep(0, p * p), p, p)
A[1:p1, 1:p1] <- 1
vizu.mat(A, "adjacency matrix")

# Compute corresponding Laplacian matrix
L <- Adj2Lap(A)
vizu.mat(L, "Laplacian matrix")
L2L.normalized

Description

Compute normalized version of graph Laplacian matrix

Usage

L2L.normalized(L)

Arguments

L  
graph Laplacian matrix

Value

normalized graph Laplacian matrix

Examples

# Define exemplary adjacency matrix
p1 <- 10
p2 <- 40
p <- p1 + p2
A <- matrix(rep(0, p * p), p, p)
A[1:p1, 1:p1] <- 1
vizu.mat(A, "adjacency matrix")

# Compute corresponding Laplacian matrix
L <- Adj2Lap(A)
vizu.mat(L, "Laplacian matrix")

# Compute corresponding Laplacian matrix - normalized
L.norm <- L2L.normalized(L)
vizu.mat(L.norm, "L Laplacian matrix (normalized)")
Description

Provides graph-constrained regression methods in which regularization parameters are selected automatically via estimation of equivalent Linear Mixed Model formulation. 'riPEER' (ridgified Partially Empirical Eigenvectors for Regression) method employs a penalty term being a linear combination of graph-originated and ridge-originated penalty terms, whose two regularization parameters are ML estimators from corresponding Linear Mixed Model solution; a graph-originated penalty term allows imposing similarity between coefficients based on graph information given whereas additional ridge-originated penalty term facilitates parameters estimation: it reduces computational issues arising from singularity in a graph-originated penalty matrix and yields plausible results in situations when graph information is not informative. 'riPEERc' (ridgified Partially Empirical Eigenvectors for Regression with constant) method utilizes addition of a diagonal matrix multiplied by a predefined (small) scalar to handle the non-invertibility of a graph Laplacian matrix. 'vrPEER' (variable reduced PEER) method performs variable-reduction procedure to handle the non-invertibility of a graph Laplacian matrix.

Usage

riPEER(Q, y, Z, X = NULL, optim.method = "rootSolve",
        rootSolve.x0 = c(1e-05, 1e-05), rootSolve.Q0.x0 = 1e-05, sbplx.x0 = c(1,
                      1), sbplx.lambda.lo = c(10^(-5), 10^(-5)), sbplx.lambda.up = c(1e+06,
                      1e+06), compute.boot.CI = FALSE, boot.R = 1000, boot.conf = 0.95,
        boot.set.seed = TRUE, boot.parallel = "multicore", boot.ncpus = 4,
        verbose = TRUE)

Arguments

Q          graph-originated penalty matrix \((p \times p)\); typically: a graph Laplacian matrix
y          response values matrix \((n \times 1)\)
Z          design matrix \((n \times p)\) modeled as random effects variables (to be penalized in regression modeling); assumed to be already standarized
X          design matrix \((n \times k)\) modeled as fixed effects variables (not to be penalized in regression modeling); if does not contain columns of 1s, such column will be added to be treated as intercept in a model
optim.method optimization method used to optimize \(\lambda = (\lambda_Q, \lambda_R)\)
• "rootSolve" (default) - optimizes by finding roots of non-linear equations by the Newton-Raphson method; from rootSolve package
• "sbplx" - optimizes with the use of Subplex Algorithm: 'Subplex is a variant of Nelder-Mead that uses Nelder-Mead on a sequence of subspaces'; from nloptr package

rootSolve.x0 vector containing initial guesses for $\lambda = (\lambda_Q, \lambda_R)$ used in "rootSolve" algorithm
rootSolve.Q0.x0 vector containing initial guess for $\lambda_R$ used in "rootSolve" algorithm
sbplx.x0 vector containing initial guesses for $\lambda = (\lambda_Q, \lambda_R)$ used in "sbplx" algorithm
sbplx.lambda.lo vector containing minimum values of $\lambda = (\lambda_Q, \lambda_R)$ grid search in "sbplx" algorithm
sbplx.lambda.up vector containing maximum values of $\lambda = (\lambda_Q, \lambda_R)$ grid search in "sbplx" algorithm
compute.boot.CI logical whether or not compute bootstrap confidence intervals for $b$ regression coefficient estimates
boot.R number of bootstrap replications used in bootstrap confidence intervals computation
boot.conf confidence level assumed in bootstrap confidence intervals computation
boot.set.seed logical whether or not set seed in bootstrap confidence intervals computation
boot.parallel value of parallel argument in boot function in bootstrap confidence intervals computation
boot.ncpus value of ncpus argument in boot function in bootstrap confidence intervals computation
verbose logical whether or not set verbose mode (print out function execution messages)

Details

Estimates coefficients of linear model of the formula:

$$y = X\beta + Zb + \epsilon$$

where:

• $y$ - response,
• $X$ - data matrix,
• $Z$ - data matrix,
• $\beta$ - regression coefficients, not penalized in estimation process,
• $b$ - regression coefficients, penalized in estimation process and for whom there is, possibly a prior graph of similarity / graph of connections available.
The method uses a penalty being a linear combination of a graph-based and ridge penalty terms:

$$\beta_{est}, b_{est} = \arg \min_{\beta, b} \{(y - X\beta - Zb)^T (y - X\beta - Zb) + \lambda_Q b^T Qb + \lambda_R b^T b\}$$

where:

- $Q$ - a graph-originated penalty matrix; typically: a graph Laplacian matrix,
- $\lambda_Q$ - regularization parameter for a graph-based penalty term
- $\lambda_R$ - regularization parameter for ridge penalty term

The two regularization parameters, $\lambda_Q$ and $\lambda_R$, are estimated as ML estimators from equivalent Linear Mixed Model optimization problem formulation (see: References).

- Graph-originated penalty term allows imposing similarity between coefficients based on graph information given.
- Ridge-originated penalty term facilitates parameters estimation: it reduces computational issues arising from singularity in a graph-originated penalty matrix and yields plausible results in situations when graph information is not informative.

Bootstrap confidence intervals computation is available (not set as a default option).

**Value**

- `b.est` vector of \( b \) coefficient estimates
- `beta.est` vector of \( \beta \) coefficient estimates
- `lambda.Q` \( \lambda_Q \) regularization parameter value
- `lambda.R` \( \lambda_R \) regularization parameter value
- `lambda.2` \( \lambda_R/\lambda_Q \) value
- `boot.CI` data frame with two columns, lower and upper, containing, respectively, values of lower and upper bootstrap confidence intervals for \( b \) regression coefficient estimates
- `obj.fn.val` optimization problem objective function value

**References**


**Examples**

```r
set.seed(1234)
n <- 200
p1 <- 10
p2 <- 90
p <- p1 + p2
# Define graph adjacency matrix
A <- matrix(rep(0, p*p), nrow = p, ncol = p)
A[1:p1, 1:p1] <- 1
```
riPEERc

Graph-constrained regression with addition of a small ridge term to handle the non-invertibility of a graph Laplacian matrix

Description

Graph-constrained regression with addition of a diagonal matrix multiplied by a predefined (small) scalar to handle the non-invertibility of a graph Laplacian matrix (see: References).

Bootstrap confidence intervals computation is available (not set as a default option).

Usage

riPEERc(Q, y, Z, X = NULL, lambda = 0.001, compute.boot.CI = FALSE, boot.R = 10000, boot.conf = 0.95, boot.set.seed = TRUE, boot.parallel = "multicore", boot.ncpus = 4, verbose = TRUE)
Arguments

- **Q**: graph-originated penalty matrix \((p \times p)\); typically: a graph Laplacian matrix
- **y**: response values matrix \((n \times 1)\)
- **Z**: design matrix \((n \times p)\) modeled as random effects variables (to be penalized in regression modeling); **assumed to be already standarized**
- **X**: design matrix \((n \times k)\) modeled as fixed effects variables (not to be penalized in regression modeling); **should contain column of 1s if intercept is to be considered in a model**
- **lambda.NR**: (small) scalar value of regularization parameter for diagonal matrix by adding which the **Q** matrix is corrected (note: correction is done before \(\lambda_Q\) regularization parameter value estimation; in other words: \(\lambda_Q\) estimation is done for the corrected Q matrix)
- **compute.boot.CI**: logical whether or not compute bootstrap confidence intervals for **b** regression coefficient estimates
- **boot.R**: number of bootstrap replications used in bootstrap confidence intervals computation
- **boot.conf**: confidence level assumed in bootstrap confidence intervals computation
- **boot.set.seed**: logical whether or not set seed in bootstrap confidence intervals computation
- **boot.parallel**: value of **parallel** argument in **boot** function in bootstrap confidence intervals computation
- **boot.ncpus**: value of **ncpus** argument in **boot** function in bootstrap confidence intervals computation
- **verbose**: logical whether or not set verbose mode (print out function execution messages)

Value

- **b.est**: vector of **b** coefficient estimates
- **beta.est**: vector of \(\beta\) coefficient estimates
- **lambda.Q**: \(\lambda_Q\) regularization parameter value
- **lambda.R**: \(\lambda_Q \times \text{lambda.2} \) value
- **lambda.2**: \(\text{lambda.2} \) supplied argument value
- **boot.CI**: data frame with two columns, **lower** and **upper**, containing, respectively, values of lower and upper bootstrap confidence intervals for **b** regression coefficient estimates

References

Examples

```r
set.seed(1234)
n <- 200
p1 <- 10
p2 <- 90
p <- p1 + p2
# Define graph adjacency matrix
A <- matrix(rep(0, p*p), nrow = p, ncol = p)
A[1:p1, 1:p1] <- 1
L <- Adj2Lap(A)
# Define Q penalty matrix as graph Laplacian matrix normalized
Q <- L2L.normalized(L)
# Define Z,X design matrices and outcome y
Z <- matrix(rnorm(n*p), nrow = n, ncol = p)
b.true <- c(rep(1, p1), rep(0, p2))
X <- matrix(rnorm(n*3), nrow = n, ncol = 3)
beta.true <- runif(3)
intercept <- 0
eta <- intercept + Z %*% b.true + X %*% beta.true
R2 <- 0.5
sd.eps <- sqrt(var(eta) * (1 - R2) / R2)
error <- rnorm(n, sd = sd.eps)
y <- eta + error
```

## Not run:
riPEERc.out <- riPEERc(Q, y, Z, X)
plt.df <- data.frame(x = 1:p, y = riPEERc.out$b.est)
ggplot(plt.df, aes(x = x, y = y, group = 1)) + geom_line() + labs("b estimates")

## End(Not run)

## Not run:
# riPEERc with 0.95 bootstrap confidence intervals computation
riPEERc.out <- riPEERc(Q, y, Z, X, compute.boot.CI = TRUE, boot.R = 500)
plt.df <- data.frame(x = 1:p, y = riPEERc.out$b.est,
  lo = riPEERc.out$boot.CI[,1],
  up = riPEERc.out$boot.CI[,2])
ggplot(plt.df, aes(x = x, y = y, group = 1)) + geom_line() +
  geom_ribbon(aes(ymin=lo, ymax=up), alpha = 0.3)

## End(Not run)

---

**vizu.mat**

Visualize matrix data in a form of a heatmap, with continuous values

**legend**
Description

Matrix data visualization in a form of a heatmap, with the use of ggplot2 library. Minimum user input (a matrix object) is needed to produce decent visualization output. Automatic plot adjustments are implemented and used as defaults, including selecting legend color palette and legend scale limits. Further plot adjustments are available, including adding a title, font size change, axis label clearing and others.

Usage

vizu.mat(matrix.object, title = "", base_size = 12, adjust.limits = TRUE, adjust.colors = TRUE, fill.scale.limits = NULL, colors.palette = NULL, geom_tile.colour = "grey90", clear.labels = TRUE, clear.x.label = FALSE, clear.y.label = FALSE, uniform.labels = FALSE, rotate.x.labels = FALSE, x.lab = "", y.lab = "", axis.text.x.size = base_size - 2, axis.text.y.size = base_size - 2, axis.title.x.size = base_size - 2, axis.title.y.size = base_size - 2, legend.text.size = base_size - 2, legend.title.size = base_size - 2, legend.title = "value", text.font.family = "Helvetica", remove.legend = FALSE, axis.text.x.breaks.idx = NULL, axis.text.y.breaks.idx = NULL)

Arguments

matrix.object matrix
title plot title
base_size base font size
adjust.limits logical whether or not adjust legend scale limits automatically:
  • legend scale starts / ends with 0 for matrix with non-negative / non-positive values only,
  • legend scale is symmetric for matrix with both negative and positive values
adjust.colors logical whether or not adjust legend color automatically:
  • legend color palette white-red for a data matrix with non-negative values only,
  • legend color palette blue-white for a data matrix with non-positive values only,
  • legend color palette blue-white-red for a data matrix with both positive and negative values
fill.scale.limits 2-element vector defining legend scale limits
colors.palette legend color color palette
geom_tile.colour tiles color value
clear.labels logical whether or not clear both x- and y-axis labels
clear.x.label logical whether or not clear x-axis labels
clear.y.label logical whether or not clear y-axis labels
uniform.labels logical whether or not define generic short column and rows labeling:

- 'c1';'c2';....'cp' for columns,
- 'r1';'r2';....'rp' for rows; might be especially useful if the matrix some long colnames and rownames already assigned

rotate.x.labels logical whether or not rotate x-axis labels by 90 degrees

x.lab x-axis label

y.lab y-axis label

axis.text.x.size font size of x-axis text

axis.text.y.size font size of y-axis text

axis.title.x.size font size of x-axis label

axis.title.y.size font size of y-axis label

legend.text.size font size of legend text

legend.title.size font size of legend title

legend.title legend title

text.font.family font family

remove.legend logical whether or not remove legend

axis.text.x.breaks.idx indices of x-axis elements whose thicks are kept and whose numerical labels are kept

axis.text.y.breaks.idx indices of y-axis elements whose thicks are kept and whose numerical labels are kept

Value

ggplot2 object

Examples

mat <- matrix(rnorm(30*30), nrow = 30, ncol = 30)
vizu.mat(mat)

vizu.mat(mat, fill.scale.limits = c(-3,3))

vizu.mat(mat, fill.scale.limits = c(-10,10))

vizu.mat(mat, fill.scale.limits = c(-10,10),
uniform.labels = TRUE, clear.labels = FALSE)

colnames(mat) <- paste0("col", 1:30, sample(LETTERS, 30, replace = TRUE))
rownames(mat) <- paste0("row", 1:30, sample(LETTERS, 30, replace = TRUE))

vizu.mat(mat, fill.scale.limits = c(-10,10),
clear.labels = FALSE,
rotate.x.labels = TRUE)
mat.positive <- abs(mat)
vizu.mat(mat.positive,
  title = "positive values only -> legend limits and colors automatically adjusted",
  clear.labels = FALSE,
  rotate.x.labels = TRUE)

vizu.mat.factor Visualize matrix data in a form of a heatmap, with categorical values

Description
Matrix data visualization in a form of a heatmap, with the use of ggplot2 library. Numerical values
are represented as categorical. Minimum user input (a matrix object) is needed to produce decent
visualization output. Further plot adjustments are available, including tile color change, adding a
title, font size change, axis label clearing and others.

Usage
vizu.mat.factor(matrix.object, title = "", base.size = 12,
scale.fill.manual.values = NULL, geom.tile.colour = "grey90",
clear.labels = TRUE, clear.x.label = FALSE, clear.y.label = FALSE,
uniform.labels = FALSE, rotate.x.labels = FALSE, x.lab = "",
y.lab = "", axis.text.x.size = base.size - 2,
axis.text.y.size = base.size - 2, axis.title.x.size = base.size - 2,
axis.title.y.size = base.size - 2, legend.text.size = base.size - 2,
legend.title.size = base.size - 2, legend.title = "value",
text.font.family = "Helvetica", remove.legend = FALSE,
factor.levels = NULL, axis.text.x.breaks.idx = NULL,
axis.text.y.breaks.idx = NULL)

Arguments
matrix.object matrix
title plot title
base.size base font size
scale.fill.manual.values vector of legend colors for categorical values
geom.tile.colour tiles color value
clear.labels logical whether or not clear both x- and y-axis labels
clear.x.label logical whether or not clear x-axis labels
clear.y.label logical whether or not clear y-axis labels
uniform.labels logical whether or not define generic short column and rows labeling:
   • 'c1','c2',.....,'cp' for columns,
   • 'r1','r2',.....,'rp' for rows; might be especially useful if the matrix some long
     colnames and rownames already assigned

rotate.x.labels logical whether or not rotate x-axis labels by 90 degrees

x.lab x-axis label

y.lab y-axis label

axis.text.x.size font size of x-axis text

axis.text.y.size font size of y-axis text

axis.title.x.size font size of x-axis label

axis.title.y.size font size of y-axis label

legend.text.size font size of legend text

legend.title.size font size of legend title

legend.title legend title

text.font.family font family

remove.legend logical whether or not remove legend

factor.levels vector of values defining levels of factors (might be used to redefine order of
    variables in the legend)

axis.text.x.breaks.idx indices of x-axis elements whose thicks are kept and whose numerical labels are
    kept

axis.text.y.breaks.idx indices of y-axis elements whose thicks are kept and whose numerical labels are
    kept

Value
ggplot2 object

Examples

mat <- diag(30)
vizu.mat.factor(mat)
vizu.mat.factor(mat, title = "some title",
scale_fill_manual.values = c(\"white\","red"),
axis.text.x.breaks.idx = seq(1,30,5),
axis.text.y.breaks.idx = seq(1,30,5))
vrPEER

Graph-constrained regression with variable-reduction procedure to handle the non-invertibility of a graph-originated penalty matrix

Description

Graph-constrained regression with variable-reduction procedure to handle the non-invertibility of a graph-originated penalty matrix (see: References).

Bootstrap confidence intervals computation is available (not set as a default option).

Usage

vrPEER(Q, y, Z, X = NULL, sv.thr = 1e-05, compute.boot.CI = FALSE,
boot.R = 1000, boot.conf = 0.95, boot.set.seed = TRUE,
boot.parallel = "multicore", boot.ncpus = 4, verbose = TRUE)

Arguments

Q  graph-originated penalty matrix \((p \times p)\); typically: a graph Laplacian matrix
y  response values matrix \((n \times 1)\)
Z  design matrix \((n \times p)\) modeled as random effects variables (to be penalized in regression modeling); **assumed to be already standarized**
X  design matrix \((n \times k)\) modeled as fixed effects variables (not to be penalized in regression modeling); **should contain colum of 1s if intercept is to be considered in a model**
sv.thr  threshold value above which singular values of \(Q\) are considered "zeros"
compute.boot.CI  logical whether or not compute bootstrap confidence intervals for \(b\) regression coefficient estimates
boot.R  number of bootstrap replications used in bootstrap confidence intervals computation
vrPEER

boot.conf  confidence level assumed in bootstrap confidence intervals computation
boot.set.seed  logical whether or not set seed in bootstrap confidence intervals computation
boot.parallel  value of parallel argument in boot function in bootstrap confidence intervals computation
boot.ncpus  value of ncpus argument in boot function in bootstrap confidence intervals computation
verbose  logical whether or not set verbose mode (print out function execution messages)

Value

b.est  vector of b coefficient estimates
beta.est  vector of β coefficient estimates
lambda.Q  λ_Q regularization parameter value
boot.CI  data frame with two columns, lower and upper, containing, respectively, values of lower and upper bootstrap confidence intervals for b regression coefficient estimates

References


Examples

set.seed(1234)
n <- 200
p1 <- 10
p2 <- 90
p <- p1 + p2
# Define graph adjacency matrix
A <- matrix(rep(0, p*p), nrow = p, ncol = p)
A[1:p1, 1:p1] <- 1
L <- Adj2Lap(A)
# Define Q penalty matrix as graph Laplacian matrix normalized
Q <- L2L.normalized(L)
# Define Z,X design matrices and a outcome y
Z <- matrix(rnorm(n*p), nrow = n, ncol = p)
b.true <- c(rep(1, p1), rep(0, p2))
X <- matrix(rnorm(n*3), nrow = n, ncol = 3)
beta.true <- runif(3)
intercept <- 0
eta <- intercept + Z %*% b.true + X %*% beta.true
R2 <- 0.5
sd.εs <- sqrt(var(eta) * (1 - R2) / R2)
error <- rnorm(n, sd = sd.εs)
y <- eta + error

## Not run:
# run vrPEER
vrPEER.out <- vrPEER(Q, y, Z, X)
plt.df <- data.frame(x = 1:p,
        y = vrPEER.out$b.est)
ggplot(plt.df, aes(x = x, y = y, group = 1)) + geom_line()

## End(Not run)

## Not run:
# run vrPEER with 0.95 confidence intervals
vrPEER.out <- vrPEER(Q, y, Z, X, compute.boot.CI = TRUE, boot.R = 500)
plt.df <- data.frame(x = 1:p,
        y = vrPEER.out$b.est,
        lo = vrPEER.out$boot.CI[,1],
        up = vrPEER.out$boot.CI[,2])
ggplot(plt.df, aes(x = x, y = y, group = 1)) + geom_line() +
        geom_ribbon(aes(ymin=lo, ymax=up), alpha = 0.3)

## End(Not run)
Index

Adj2Lap, 2
L2L.normalized, 3
mdpeer, 3
mdpeer-package (mdpeer), 3
riPEER, 4
riPEERc, 7
vizu.mat, 9
vizu.mat.factor, 12
vrPEER, 14