Package 'ncpen'

February 14, 2018

Type Package

Title Nonconvex penalty estimation

Version 0.1.10

Date 2017-12-13

Description Estimates nonconvex penalty.
This project is funded by Julian Virtue Professorship from
Center for Applied Research at Graziadio School of Business and Management at
Pepperdine University.

License GPL (>= 3) + file LICENSE

LazyData TRUE

Imports Rcpp (>= 0.11.2)

LinkingTo Rcpp, RcppArmadillo

Depends R(>= 3.1)

RoxygenNote 6.0.1

R topics documented:

`ncpen-package` *ncpen: A package for non-convex penalized estimation in generalized linear models*

Description

This package fits the generalized linear models with various non-convex penalties. A unified algorithm is implmented in **ncpen** based on the convex concave procedure or difference convex algorithm that can be applied to most of existing non-convex penalties. The available penalties in the pacakge are the least absolute shrinkage and selection operator(LASSO), smoothly clipped absolute deviation (SCAD), minimax concave penalty (MCP), truncated ℓ_1-penalty (TLP), clipped LASSO (CLASSO), sparse bridge (SRIDGE), modified bridge (MBRIDGE), and modified log (MLOG) penalites.

Details

Accepts a design matrix X and vector of responses y, and produces the regularization path ovaer a grid of values for the tuning parameter `lambda`. Also provides user-friendly processes for plotting, selecting tuning parameters using cross-validation or generalized information criterion (GIC), ℓ_2-regularization, penalty weights, standardization and intercept.

Note

This project is funded by Julian Virtue Professorship from Center for Applied Research at Graziadio School of Business and Management at Pepperdine University.

Author(s)

Dongshin Kim, Sunghoon Kwon and Sangin Lee

References

Kim, D., Kwon, S. and Lee, S. (2017). A unified algorithm for various penalized regression models: **R** Package **ncpen**.

`coef.cv.ncpen` *Extracts the optimal vector of coefficients from a* `cv.ncpen` *object.*

Description

This function returns the optimal vector of coefficients.

Usage

```
## S3 method for class 'cv.ncpen'
coef(object, type = c("error", "deviance"), ...)
```

Arguments

object	fitted cv.ncpen object.
type	(character) a cross-validated error type which is either "error" or "deviance". Each error type is defined in cv.ncpen.
...	Other arguments to coef. Not supported.

Value

the optimal coefficients vector selected by cross-validation method.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

Maintainer: Dongshin Kim<dongshin.kim@outlook.com>, Sunghoon Kwon<shkwon0522@gmail.com>

References

Kim, D., Kwon, S. and Lee, S. (2017). A unified algorithm for various penalized regression models: **R** Package **ncpen**.

See Also

cv.ncpen, plot.cv.ncpen

Examples

```
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5,family="binomial", seed = 1234)
x.mat = s0$x.mat
y.vec = s0$y.vec

cvfit = cv.ncpen(y.vec=y.vec, x.mat=x.mat, family="binomial")
coef.cv.ncpen(cvfit, type="deviance")
```

---

coef.ncpen *Extract the coefficients from an* ncpen *object*

Description

This function returns the coefficients matrix for all lambda values.

Usage

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

Arguments

object	Fitted ncpen object.
...	Other parameters to coef. Not supported.

Value

The coefficients matrix.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

Maintainer: Dongshin Kim<dongshin.kim@outlook.com>, Sunghoon Kwon<shkwon0522@gmail.com>

References

Kim, D., Kwon, S. and Lee, S. (2017). A unified algorithm for various penalized regression models: **R** Package **ncpen**.

See Also

ncpen, plot.ncpen, predict.ncpen

Examples

```
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5, seed = 1234)
x.mat = s0$x.mat
y.vec = s0$y.vec

fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="gaussian")
coef(fit)
```

---

cv.ncpen *Cross-validation for ncpen*

Description

Performs k-fold cross-validation for noncovex penalized regression models over a sequence of the regularization parameter lambda.

Usage

```
cv.ncpen(y.vec, x.mat, family = c("gaussian", "binomial", "poisson"),
  penalty = c("scad", "mcp", "tlp", "lasso", "classo", "sridge", "mbridge",
  "mlog"), n.fold = 10, lambda = NULL, n.lambda = 100, r.lambda = 0.001,
  pen.weight = NULL, tau = switch(penalty, scad = 3.7, mcp = 3, tlp = 0.1,
  lasso = 1, classo = 2, sridge = 2, mbridge = 0.1, mlog = 0.1),
  gamma = 1e-06, ridge = 1e-06, df.max = 50, proj.min = 50,
  iter.max = 1000, b.eps = 1e-07, k.eps = 1e-06, x.standardize = TRUE,
  intercept = TRUE)
```

Arguments

y.vec	(numeric vector) response vector.
x.mat	(numeric matrix) design matrix. Each row is an observation vector.
family	(character) regression model. Default is "gaussian".
penalty	(character) penalty function. Default is "scad".
n.fold	(numeric) the number of folds. Default value is 10. It should be 3 or greater.
lambda	(numeric vector): user-specified sequence of lambda values.
n.lambda	(numeric) the number of lambda values. Default is 100.
r.lambda	(numeric) ratio of the smallest value for lambda to lambda.max (which derived from data) for which all coefficients are zero. Default is 1e-3.
pen.weight	(numeric vector) penalty weights for each coefficient. If a penalty weigth is set to zero, the corresponding coefficient is always non-zero without shrinkage. Note: the penalty weights are internally rescaled to sum to the number of variables, and the lambda sequence reflects this change.
tau	(numeric) concavity parameter of the concave penalties (see reference). Default is 3.7 for scad, 3 for mcp, 2 for classo and sridge, 0.1 for tlp, mbridge and mlog.
gamma	(numeric) addtional tunning parameter for the classo and sbridge. Default value is 1e-6.
ridge	(numeric) ridge effect (amount of ridge penalty). Default value is 1e-6.
df.max	(numeric) the maximum number of nonzero coefficients. Default is 50.
proj.min	(numeric) the minimum number of iterations which will be applied to projections (see details). Default value is 50.
iter.max	(numeric) maximum number of iterations. Default valu eis 1e+3.
b.eps	(numeric) convergence threshold for $L2$ norms of coefficnets vector. Default value is 1e-7.
k.eps	(numeric) convergence threshold for KKT conditions. Default value is 1e-6.
x.standardize	(logical) whether to standardize the x.mat prior to fitting the model. The estimated coefficients are always restored to the original scale. Default value is TRUE.
intercept	(logical) whether to include an intercept in the model. Default value is TRUE.
...	other paramemters are same as in ncpen.

Details

The function runs the ncpen function for n.fold+1 times. The first run is to get the sequence of lambda and then the rest runs are to compute the fit with each of the folds omitted. It provides the cross validated-error based on the squared-error loss and the deviance loss.

Value

An object with S3 class cv.ncpen.

ncpen.fit	the fitted ncpen object.
opt.ebeta	the optimal coefficients vector selected by using the squared-error loss in the cross-valdation.

opt.dbeta	the optimal coefficients vector selected by using the deviance loss in the cross-valdation.
cv.error	the averaged cross-validated error for each value of lambdas.
cv.deviance	the averaged cross-validated deviance for each value of lambdas.
elambda	the lambda sequence used for computing cv error.
dlambda	the lambda sequence used for computing cv deviance.
opt.elambda	the optimal value of lambda based on cv error.
opt.dlambda	the optimal value of lambda based on cv deviance.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

Maintainer: Dongshin Kim<dongshin.kim@outlook.com>, Sunghoon Kwon<shkwon0522@gmail.com>

References

Kim, D., Kwon, S. and Lee, S. (2017). A unified algorithm for various penalized regression models: **R** Package **ncpen**.

See Also

ncpen, plot.cv.ncpen, coef.cv.ncpen

Examples

```
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5,family="gaussian", seed = 1234)
x.mat = s0$x.mat
y.vec = s0$y.vec

cvfit = cv.ncpen(y.vec=y.vec,x.mat=x.mat,family="gaussian",n.fold=10)  # not run !!!
coef.cv.ncpen(cvfit)
plot.cv.ncpen(cvfit)
fit = cvfit$ncpen.fit
opt = which(cvfit$opt.elambda==fit$lambda)
coef(fit)[,opt]
```

---

excluded *Check whether a pair should be excluded from interactions.*

Description

This is internal use only function.

Usage

```
excluded(excluded.pair, a, b)
```

Arguments

excluded.pair	a pair.
a	first column to be compared.
b	second coumn to be compared.

Value

TRUE if exculuded, FALSE otherwise.

---

gic.ncpen *Compute the GIC values for the selection of the regularizatin parameter lambda.*

Description

This function provides the selection of the regularization parameter lambda based on the generlized information criterion (GIC) including AIC and BIC. It computes the GIC values at a grid of values for the regularization parameter lambda.

Usage

```
gic.ncpen(ncpen.fit, y.vec, x.mat, df.weight = log(length(y.vec)),
  verbose = TRUE)
```

Arguments

ncpen.fit	Fitted ncpen model object.
y.vec	the response vector.
x.mat	the design matrix.
df.weight	the weight factor for various information critera. For example, AIC if df.weight=2, BIC if df.weight=log(n). Default is BIC.
verbose	(logical) whether to plot the GIC curve. Default is verbose=TRUE.

Value

The coefficients matrix.

opt.beta	the optimal coefficients vector selected by GIC.
lambda	the sequence of lambda values in the ncpen object.
gic	the GIC values for all lambda values.
opt.lambda	the optimal lambda value.
plot	the GIC curve.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

Maintainer: Dongshin Kim<dongshin.kim@outlook.com>, Sunghoon Kwon<shkwon0522@gmail.com>

References

Kim, D., Kwon, S. and Lee, S. (2017). A unified algorithm for various penalized regression models: **R** Package **ncpen**.

Kim, Y., Kwon, S. and Choi, H. (2012). Consistent Model Selection Criteria on High Dimensions. *Journal of Machine Learning Research*, **13**, 1037-1057.

See Also

ncpen

Examples

```
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5, seed = 1234)
x.mat = s0$x.mat
y.vec = s0$y.vec

fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="gaussian")
gic.ncpen(fit,y.vec,x.mat,verbose=TRUE)
```

---

interact.data *Construct Interaction Matrix*

Description

interact.data interacts all the data in a data.frame or matrix.

Usage

```
interact.data(data, base.cols = NULL, exclude.pair = NULL)
```

Arguments

data	a data.frame or matrix to interact.
base.cols	indicates coulumns from one category. Interactions among variables from a same base.col will be avoided. For example, if three indicator columns, "ChannelR", "ChannelC" and "ChannelB", are created from a categorical column "Channel", then the interaction among them can be excluded by assining base.cols=c("Channel"). Multiple base.cols are possible.
exclude.pair	the pairs will be excluded from interactions. This should be a list object of pairs. For example, list(c("a1", "a2"), c("d1", "d2")).

Value

This returns an object of matrix which contains interactions.

Examples

```
df = data.frame(1:3, 4:6, 7:9, 10:12, 13:15);
colnames(df) = c("aa", "bb", "cc", "dd", "aa2");
df

interact.data(df);
interact.data(df, base.cols = "aa");
interact.data(df, base.cols = "aa", exclude.pair = list(c("bb", "cc")));
```

---

native_cpp_ncpen_fun_ *Native point ncpen function.*

Description

This is internal use only function.

Usage

```
native_cpp_ncpen_fun_(y_vec, x_mat0, x_std, intc, w_vec0, lam_vec0, r_lam, gam,
  tau, p_max, iter_max, b_eps, k_eps, p_eff, r_eff, family, penalty)
```

Arguments

y_vec	.
x_mat0	.
x_std	.
intc	.
w_vec0	.
lam_vec0	.
r_lam	.
gam	.
tau	.
p_max	.
iter_max	.
b_eps	.
k_eps	.
p_eff	.
r_eff	.
family	.
penalty	.

Value

.

native_cpp_obj_fun_ *Native object function.*

Description

This is internal use only function.

Usage

```
native_cpp_obj_fun_(name, y_vec, x_mat, b_vec, r_eff)
```

Arguments

name	.
y_vec	.
x_mat	.
b_vec	.
r_eff	.

Value

.

native_cpp_obj_grad_fun_ *Native object gradient function.*

Description

This is internal use only function.

Usage

```
native_cpp_obj_grad_fun_(name, y_vec, x_mat, b_vec, r_eff)
```

Arguments

name	.
y_vec	.
x_mat	.
b_vec	.
r_eff	.

Value

.

native_cpp_p_ncpen_fun_ *Native point ncpen function.*

Description

This is internal use only function.

Usage

```
native_cpp_p_ncpen_fun_(y_vec, x_mat, b_vec, w_vec, lam, gam, tau, iter_max,
  b_eps, k_eps, p_eff, r_eff, family, penalty)
```

Arguments

y_vec	.
x_mat	.
b_vec	.
w_vec	.
lam	.
gam	.
tau	.
iter_max	.
b_eps	.
k_eps	.
p_eff	.
r_eff	.
family	.
penalty	.

Value

.

native_cpp_qlasso_fun_ *Native QLASSO function.*

Description

This is internal use only function.

Usage

```
native_cpp_qlasso_fun_(q_mat, l_vec, b_vec0, w_vec, lam, iter_max, b_eps, k_eps,
  p_eff, q_rank)
```

Arguments

q_mat	.
l_vec	.
b_vec0	.
w_vec	.
lam	.
iter_max	.
b_eps	.
k_eps	.
p_eff	.
q_rank	.

Value

.

---

ncpen *Fits a generalized linear model (GLM) with various nonconvex penalties*

Description

Fits a generalized linear model by penalized maximum likelihood estimation. The coefficients path is computed for the penalized regression model over a grid of values for the regularization parameter λ. Fits gaussian (linear), binomial (logistic) and poisson regression models with various non-convex penalties such as SCAD, MCP and clipped Lasso.

Usage

```
ncpen(y.vec, x.mat, family = c("gaussian", "binomial", "poisson"),
  penalty = c("scad", "mcp", "tlp", "lasso", "classo", "sridge", "mbridge",
  "mlog"), lambda = NULL, n.lambda = 100, r.lambda = 0.001,
  pen.weight = NULL, tau = switch(penalty, scad = 3.7, mcp = 3, tlp = 0.1,
  lasso = 1, classo = 2, sridge = 2, mbridge = 0.1, mlog = 0.1),
  gamma = 1e-06, ridge = 1e-06, df.max = 50, proj.min = 50,
  iter.max = 1000, b.eps = 1e-07, k.eps = 1e-06, x.standardize = TRUE,
  intercept = TRUE)
```

Arguments

y.vec	(numeric vector) response vector.
x.mat	(numeric matrix) design matrix. Each row is an observation vector.
family	(character) regression model. Default is "gaussian".
penalty	(character) penalty function. Default is "scad".
lambda	(numeric vector): user-specified sequence of lambda values.
n.lambda	(numeric) the number of lambda values. Default is 100.

r.lambda	(numeric) ratio of the smallest value for lambda to lambda.max (which derived from data) for which all coefficients are zero. Default is 1e-3.
pen.weight	(numeric vector) penalty weights for each coefficient. If a penalty weigth is set to zero, the corresponding coefficient is always non-zero without shrinkage. Note: the penalty weights are internally rescaled to sum to the number of variables, and the lambda sequence reflects this change.
tau	(numeric) concavity parameter of the concave penalties (see reference). Default is 3.7 for scad, 3 for mcp, 2 for classo and sridge, 0.1 for tlp, mbridge and mlog.
gamma	(numeric) addtional tunning parameter for the classo and sbridge. Default value is 1e-6.
ridge	(numeric) ridge effect (amount of ridge penalty). Default value is 1e-6.
df.max	(numeric) the maximum number of nonzero coefficients. Default is 50.
proj.min	(numeric) the minimum number of iterations which will be applied to projections (see details). Default value is 50.
iter.max	(numeric) maximum number of iterations. Default valu eis 1e+3.
b.eps	(numeric) convergence threshold for $L2$ norms of coefficnets vector. Default value is 1e-7.
k.eps	(numeric) convergence threshold for KKT conditions. Default value is 1e-6.
x.standardize	(logical) whether to standardize the x.mat prior to fitting the model. The estimated coefficients are always restored to the original scale. Default value is TRUE.
intercept	(logical) whether to include an intercept in the model. Default value is TRUE.

Details

The sequence of models indexed by the regularization parameter lambda is fit by the unified algorithm using concave convex procedure and coordiante descent algorithm. Note that the objective function is

$$RSS/2n + penalty$$

for family="gaussian", and

$$(negativelog - likelihood)/n + penalty$$

for family="binomial" or family="poisson", where log-likelihood is computed with assuming the canonical link (logit for binomial; log for poisson).

The algorithm fits the coefficients in the active set using the projection method after proj.min iteration instead of cycling coordinates, which makes the algorithm fast and stable.

Value

An object with S3 class ncpen.

family	regression model.
x.standardize	flag for standardization of x.mat.
intercept	flag for an intercept in the model.
coefficients	a matrix of fitted coefficients for a lambda sequence. The number of rows is same as the number of coefficients (ncol(x.mat)+1 if intercept=TRUE and ncol(x.mat) if intercept=FALSE). The number of columns is equal to nlambda.

pen.weight	penalty weights for each coefficient.
lambda	sequence of lambda values used.
df	the number of non-zero coefficients for each lambda value.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

Maintainer: Dongshin Kim<dongshin.kim@outlook.com>, Sunghoon Kwon<shkwon0522@gmail.com>

References

Kim, D., Kwon, S. and Lee, S. (2017). A unified algorithm for various penalized regression models: **R** Package **ncpen**.

See Also

plot.ncpen, coef.ncpen, cv.ncpen

Examples

```
### Linear regression
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5,family="gaussian", seed = 1234)
x.mat = s0$x.mat
y.vec = s0$y.vec

# 1. SCAD
fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="gaussian")
coef(fit)
plot(fit)
predict(fit, new.x.mat=x.mat[1:20,],type="regression")
gic.ncpen(fit,y.vec,x.mat)

# 2. CLASSO
fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="gaussian", penalty="classo")
plot(fit)
predict(fit, new.x.mat=x.mat[1:20,],type="regression")

# 3. TLP
fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="gaussian", penalty="tlp")
plot(fit)
predict(fit, new.x.mat=x.mat[1:20,],type="regression")

### Logistic regression
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5,family="binomial", seed = 1234)
x.mat = s0$x.mat
y.vec = s0$y.vec

fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="binomial")
predict(fit, new.x.mat=x.mat[1:20,],type="probability")
predict(fit, new.x.mat=x.mat[1:20,],type="response")

### Poison regression
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5,family="poisson", seed = 1234)
x.mat = s0$x.mat
```

```
y.vec = s0$y.vec

fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="poisson")
predict(fit, new.x.mat=x.mat[1:20,],type="response")
gic.ncpen(fit,y.vec,x.mat)
plot(fit)
```

`plot.cv.ncpen` *Plot cv curve from a* `cv.ncpen` *object*

Description

Produces a plot of the cross-validated error curve from a fitted `cv.ncpen` object.

Usage

```
## S3 method for class 'cv.ncpen'
plot(x, type = c("error", "deviance"), log.scale = FALSE,
  ...)
```

Arguments

`x`	fitted `cv.ncpen` object.
`type`	(character) a cross-validated error type which is either "error" or "deviance". Each error type is defined in `cv.ncpen`.
`log.scale`	(logical) log scale of horizontal axis (a sequence of lambda values). Default value is FALSE.
`...`	other graphical parameters to `plot`.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

Maintainer: Dongshin Kim<dongshin.kim@outlook.com>, Sunghoon Kwon<shkwon0522@gmail.com>

References

Kim, D., Kwon, S. and Lee, S. (2017). A unified algorithm for various penalized regression models: **R** Package **ncpen**.

See Also

`cv.ncpen`, `coef.cv.ncpen`

Examples

```
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5,family="binomial")
x.mat = s0$x.mat
y.vec = s0$y.vec

cvfit = cv.ncpen(y.vec=y.vec, x.mat=x.mat, family="binomial")
plot.cv.ncpen(cvfit, type="deviance")
```

plot.ncpen *Plots coefficients from an* ncpen *object.*

Description

Produces a plot of the coefficients paths for a fitted ncpen object.

Usage

```
## S3 method for class 'ncpen'
plot(x, log.scale = FALSE, ...)
```

Arguments

x	Fitted ncpen model object.
log.scale	(logical) log scale of horizontal axis (a sequence of lambda values). Default value is FALSE.
...	other graphical parameters to plot

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

Maintainer: Dongshin Kim<dongshin.kim@outlook.com>, Sunghoon Kwon<shkwon0522@gmail.com>

References

Kim, D., Kwon, S. and Lee, S. (2017). A unified algorithm for various penalized regression models: **R** Package **ncpen**.

See Also

ncpen

Examples

```
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5, seed = 1234)
x.mat = s0$x.mat
y.vec = s0$y.vec

fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="gaussian")
plot(fit,log.scale=FALSE)
```

power.data *Power Data*

Description

power.data power data and return a data.frame with column names with tail.

Usage

```
power.data(data, power, tail = "_pow")
```

Arguments

data	a data.frame or matrix object.
power	power.
tail	tail text for column names for powered data. For example, if a column "sales" is powered by 4 (=power) and tail is "_pow", then the output column name becomes "sales_pow4".

Value

This returns an object of matrix.

Examples

```
df = data.frame(a = 1:3, b= 4:6);
power.data(df, 2, ".pow");
```

predict.ncpen *Make predictions from an* ncpen *object.*

Description

This function provides predictions from a fitted ncpen object.

Usage

```
## S3 method for class 'ncpen'
predict(object, new.x.mat = NULL, type = c("regression",
  "probability", "response"), cut = 0.5, ...)
```

Arguments

object	fitted ncpen object.
new.x.mat	(numeric matrix). A matrix of new observations at which predictions are to be made.
type	(character) type of prediction. "regression" returns the linear predictors; "probability" returns the fitted probabilities which is only available for family="binomial"; "response" returns followings depending on the models: the fitted values for "gaussian", fitted class using cut value for "binomial", and fitted means for "poisson".
cut	(numeric) threshold value of probability for logistic regression model. Default value is 0.5. This argument is only required for logistic regression (binomial).
...	Other parameters to prediction. Not supported.

Value

the matrix of the fitted values depending on type for all lambda values.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

Maintainer: Dongshin Kim<dongshin.kim@outlook.com>, Sunghoon Kwon<shkwon0522@gmail.com>

References

Kim, D., Kwon, S. and Lee, S. (2017). A unified algorithm for various penalized regression models: **R** Package **ncpen**.

See Also

ncpen

Examples

```
### Linear regression
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5,family="gaussian")
x.mat = s0$x.mat
y.vec = s0$y.vec

fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="gaussian")
predict(fit, new.x.mat=x.mat[1:20,], type="regression")

### Logistic regression
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5,family="binomial")
x.mat = s0$x.mat
y.vec = s0$y.vec

fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="binomial")
predict(fit, new.x.mat=x.mat[1:20,], type="probability")
predict(fit, new.x.mat=x.mat[1:20,], type="response")

### Poisson regression
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5,family="poisson")
```

```
x.mat = s0$x.mat
y.vec = s0$y.vec

fit = ncpen(y.vec=y.vec, x.mat=x.mat, family="poisson")
predict(fit, new.x.mat=x.mat[1:20,], type="regression")
predict(fit, new.x.mat=x.mat[1:20,], type="response")
```

`sam.gen.fun` *Generate a simulated dataset.*

Description

Generate a synthetic dataset based on the correlation structure from generalized linear models.

Usage

```
sam.gen.fun(n = 100, p = 50, q = 10, bmin = 0.5, bmax = 1,
  corr = 0.5, family = "gaussian", seed = NA)
```

Arguments

`n`	(numeric) the number of samples.
`p`	(numeric) the number of variables.
`q`	(numeric) the number of nonzero coefficients.
`bmin`	(numeric) value of the minimum coefficient.
`bmax`	(numeric) value of the maximum coefficient.
`corr`	(numeric) strength of correlations in the correlation structure.
`family`	(character) model type. Default is `"gaussian"`.
`seed`	(numeric) seed number for random generation. If set to NA, no seed will be applied. Default value is NA.

Details

A design matrix for regression models is generated from the multivariate normal distribution with the correlation structure. Then the response variables are computed with a specific model based on the true coefficients. For details, see the reference.

Value

An object with list class containing

`x.mat`	`n` times `p` design matrix.
`y.vec`	vector of responses.
`b.vec`	vector of true coefficients.

Author(s)

Dongshin Kim, Sunghoon Kwon, Sangin Lee

Maintainer: Dongshin Kim<dongshin.kim@outlook.com>, Sunghoon Kwon<shkwon0522@gmail.com>

References

Kim, D., Kwon, S. and Lee, S. (2017). A unified algorithm for various penalized regression models: **R** Package **ncpen**.

See Also

ncpen

Examples

```
s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.5,bmax=1,corr=0.5,family="gaussian", seed = 1234)
head(s0$x.mat)
head(s0$y.vec)
head(s0$b.vec)

s0 = sam.gen.fun(n=100,p=20,q=10,bmin=0.2,bmax=0.5,corr=0.7,family="binomial", seed = 1234)
head(s0$y.vec)
head(s0$b.vec)

s0 = sam.gen.fun(n=100,p=20,q=5,bmin=0.5,bmax=1,corr=0.3,family="poisson", seed = 1234)
head(s0$y.vec)
head(s0$b.vec)
```

---

same.base *Check whether column names are derivation of a same base.*

Description

This is internal use only function.

Usage

```
same.base(base.cols, a, b)
```

Arguments

base.cols	vector of base column names.
a	first column to be compared.
b	second coumn to be compared.

Value

TRUE if same base, FALSE otherwise.

`to.indicators` *Construct Indicator Matrix*

Description

`to.indicators` converts a categorical variable into a `data.frame` with indicator (0 or 1) variables for each category.

Usage

```
to.indicators(vec, exclude.base = TRUE, base = NULL, prefix = NULL)
```

Arguments

`vec`	a categorical vector.
`exclude.base`	`FALSE` means to include all the categories. `TRUE` means to exlcude one category as a base case. If `base` is not specified, a random category will be removed.
`base`	a base category removed from the indicator matrix. This option works only when the `type` variable is set to `"exclude.base"`.
`prefix`	a prefix to be used for column names of the output matrix. Default is "cat_" if `prefix` is `NULL`. For example, if a category vector has values of c("a", "b", "c"), cloumn names of the output matrix will be "cat_aa", "cat_bb" and "cat_cc". If `vec` is a `data.frame` and `prefix` is `NULL`, then the `vec`'s column name followed by "_" will be used as a prefix.

Value

This returns an object of `matrix` which contains indicators.

Examples

```
a1 = 4:10;
b1 = c("aa", "bb", "cc");

to.indicators(a1, base = 10);
to.indicators(b1, base = "bb", prefix = "T_");
to.indicators(as.data.frame(b1), base = "bb");
```

Index