Package: pboost (via r-universe)

May 22, 2026

Title Profile Boosting Framework for Parametric Models

Version 0.4.11

Maintainer Zengchao Xu <zengc.xu@aliyun.com>

Description A profile boosting framework for feature selection in parametric models. It offers a unified interface pboost() and several wrapped models, including linear model, generalized linear models, quantile regression, Cox proportional hazards model, beta regression, spatial auto-regressive models.

Imports stats, Matrix, MASS, betareg, quantreg, survival, Formula (>= 1.2.5), glmnet, ncvreg, rqPen, spatialreg

License GPL (>= 3)

Encoding UTF-8

URL https://github.com/paradoxical-rhapsody/pboost

BugReports https://github.com/paradoxical-rhapsody/pboost/issues

Roxygen list(load=` `source", markdown = TRUE)

Depends R (>= 4.1.0)

RoxygenNote 8.0.0

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

Repository https://paradoxical-rhapsody.r-universe.dev

Date/Publication 2026-05-22 11:29:18 UTC

RemoteUrl https://github.com/paradoxical-rhapsody/pboost

RemoteRef HEAD

RemoteSha af80813c05c71e07c9ca9ff48f4e004af89ff603

Contents

`coef.penaltyglmnet` *Extract Coefficients from Lasso Penalized Models*

Description

Extract Coefficients from Lasso Penalized Models

Usage

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

Arguments

`object`	Object.
`...`	Additional arguments (not used).

Value

Named vector of non-zero coefficients of under cross-validation error.

coef.penaltyncvreg *Extract Coefficients from Non-Convex Penalized Models*

Description

Extract Coefficients from Non-Convex Penalized Models

Usage

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

Arguments

object	Object.
...	Additional arguments (not used).

Value

Named vector of non-zero coefficients of under cross-validation error.

coef.penaltyrq *Extract Coefficients from Non-Convex Penalized Models*

Description

Extract Coefficients from Non-Convex Penalized Models

Usage

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

Arguments

object	Object.
...	Additional arguments (not used).

Value

A named vector of coefficients.

EBIC *Extended Bayesian Information Criterion*

Description

The Extended BIC possesses the selection consistency in high-dimensional model.

It can be called by the fitted model that has standard logLik method to access the attributes `nobs` and `df`, such as lm, glm.

Usage

```
EBIC(object, p, p.keep, ...)
```

Arguments

`object`	Fitted model object.
`p`	Total number of candidate features, which is available in pboost.
`p.keep`	Number of features that are pre-specified to be kept in model.
`...`	Additional parameters, which is available in pboost.

Details

The extended BIC (EBIC) is defined as

```
EBIC(obj) = BIC(obj) + 2 * r * log(choose(p - |p.keep|, df - |p.keep|)).
```

Value

A function to obtain the EBIC value of a fitted object.

References

- Jiahua Chen and Zehua Chen (2008). Extended Bayesian information criteria for model selection with large model spaces. Biometrika, 95(3):759–771. doi:10.1093/biomet/asn034
- Jiahua Chen and Zehua Chen (2012). Extended BIC for small-n-large-p sparse GLM. Statistical Sinica, 22(2):555–574. doi:10.5705/ss.2010.216

fbetareg *Forward Regression Selection for Beta Regression Models*

Description

`fbetareg()` inherits the usage of the function betareg::betareg, and performs forward regression selection for beta regression models.

Usage

```
fbetareg(
  formula,
  data,
  subset,
  na.action,
  weights,
  offset,
  link = c("logit", "probit", "cloglog", "cauchit", "log", "loglog"),
  link.phi = NULL,
  type = c("ML", "BC", "BR"),
  dist = NULL,
  nu = NULL,
  control = betareg.control(...),
  model = TRUE,
  y = TRUE,
  x = FALSE,
  ...,
  selectFun = logLik,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

`formula`	Parameter passed to betareg::betareg.
`data`	Parameter passed to betareg::betareg.
`subset`	Parameter passed to betareg::betareg.
`na.action`	Parameter passed to betareg::betareg.
`weights`	Parameter passed to betareg::betareg.
`offset`	Parameter passed to betareg::betareg.
`link`	Parameter passed to betareg::betareg.
`link.phi`	Parameter passed to betareg::betareg.
`type`	Parameter passed to betareg::betareg.

dist	Parameter passed to betareg::betareg.
nu	Parameter passed to betareg::betareg.
control	Parameter passed to betareg::betareg.
model	Parameter passed to betareg::betareg.
y	Parameter passed to betareg::betareg.
x	Parameter passed to betareg::betareg.
...	Parameters passed to betareg::betareg.
selectFun	Parameter passed to frs.
stopFun	Parameter passed to frs.
keep	Parameter passed to frs.
maxK	Parameter passed to frs.
verbose	Parameter passed to frs.

Value

A betareg model object fitted on the selected features.

---

fcoxph *Forward Regression Selection for Cox proportional hazards Model*

Description

Forward regression selection for Cox model.

Usage

```
fcoxph(
  formula,
  data,
  weights,
  subset,
  na.action,
  init,
  control,
  ties = c("efron", "breslow", "exact"),
  singular.ok = TRUE,
  robust,
  model = FALSE,
  x = FALSE,
  y = TRUE,
  tt,
  method = ties,
  id,
  cluster,
```

```
  istate,
  statedata,
  nocenter = c(-1, 0, 1),
  ...,
  selectFun = logLik,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

formula	Parameter passed to survival::coxph.
data	Parameter passed to survival::coxph.
weights	Parameter passed to survival::coxph.
subset	Parameter passed to survival::coxph.
na.action	Parameter passed to survival::coxph.
init	Parameter passed to survival::coxph.
control	Parameter passed to survival::coxph.
ties	Parameter passed to survival::coxph.
singular.ok	Parameter passed to survival::coxph.
robust	Parameter passed to survival::coxph.
model	Parameter passed to survival::coxph.
x	Parameter passed to survival::coxph.
y	Parameter passed to survival::coxph.
tt	Parameter passed to survival::coxph.
method	Parameter passed to survival::coxph.
id	Parameter passed to survival::coxph.
cluster	Parameter passed to survival::coxph.
istate	Parameter passed to survival::coxph.
statedata	Parameter passed to survival::coxph.
nocenter	Parameter passed to survival::coxph.
...	Parameters passed to survival::coxph.
selectFun	Parameter passed to frs.
stopFun	Parameter passed to frs.
keep	Parameter passed to frs.
maxK	Parameter passed to frs.
verbose	Parameter passed to frs.

Value

A coxph model object fitted on the selected features.

fglm *Forward Regression Selection for Generalized Linear Models*

Description

`fglm()` inherits the usage of glm, and performs forward regression selection for generalized linear models.

Usage

```
fglm(
  formula,
  family = gaussian,
  data,
  weights,
  subset,
  na.action,
  start = NULL,
  etastart,
  mustart,
  offset,
  control = list(...),
  model = TRUE,
  method = "glm.fit",
  x = FALSE,
  y = TRUE,
  singular.ok = TRUE,
  contrasts = NULL,
  ...,
  selectFun = logLik,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)

fglm.fit(
  x,
  y,
  weights = rep.int(1, NROW(y)),
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  offset = rep.int(0, NROW(y)),
  family = gaussian(),
  control = list(),
  intercept = TRUE,
```

```
  singular.ok = TRUE,
  selectFun = "logLik",
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

formula	Parameter passed to glm.
family	Parameter passed to glm.
data	Parameter passed to glm.
weights	Parameter passed to glm.
subset	Parameter passed to glm.
na.action	Parameter passed to glm.
start	Parameter passed to glm.
etastart	Parameter passed to glm.
mustart	Parameter passed to glm.
offset	Parameter passed to glm.
control	Parameter passed to glm.
model	Parameter passed to glm.
method	Parameter passed to glm.
x	Parameter passed to glm.
y	Parameter passed to glm.
singular.ok	Parameter passed to glm.
contrasts	Parameter passed to glm.
...	Parameters passed to glm.
selectFun	Parameter passed to frs.
stopFun	Parameter passed to frs.
keep	Parameter passed to frs.
maxK	Parameter passed to frs.
verbose	Parameter passed to frs.
intercept	Parameter passed to glm.fit.

Value

A glm model object fitted on the selected features.

`flagsarlm` *Forward Stepwise Spatial Auto-Regressive Model*

Description

Forward stepwise strategy for spatial auto-regressive model.

Usage

```
flagsarlm(
  formula,
  data = list(),
  listw,
  na.action,
  Durbin,
  type,
  method = "eigen",
  quiet = NULL,
  zero.policy = NULL,
  interval = NULL,
  tol.solve = .Machine$double.eps,
  trs = NULL,
  control = list(),
  selectFun = logLik,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)

fsar.fit(
  x,
  y,
  w,
  selectFun = logLik,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

`formula`	Parameters passed to spatialreg::lagsarlm.
`data`	Parameters passed to spatialreg::lagsarlm.
`listw`	Parameters passed to spatialreg::lagsarlm.

na.action	Parameters passed to spatialreg::lagsarlm.
Durbin	Parameters passed to spatialreg::lagsarlm.
type	Parameters passed to spatialreg::lagsarlm.
method	Parameters passed to spatialreg::lagsarlm.
quiet	Parameters passed to spatialreg::lagsarlm.
zero.policy	Parameters passed to spatialreg::lagsarlm.
interval	Parameters passed to spatialreg::lagsarlm.
tol.solve	Parameters passed to spatialreg::lagsarlm.
trs	Parameters passed to spatialreg::lagsarlm.
control	Parameters passed to spatialreg::lagsarlm.
selectFun	Parameter passed to frs.
stopFun	Parameter passed to frs.
keep	Parameter passed to frs.
maxK	Parameter passed to frs.
verbose	Parameter passed to frs.
x	Matrix of covariates.
y	Vector of response.
w	Weight matrix (row-sum scaled being one).

Value

Model object fitted on the selected features.

`flm` *Forward Regression Selection for Linear Models.*

Description

`flm()` inherits the usage of the function lm, and performs forward regression selection for linear models.

Usage

```
flm(
  formula,
  data,
  subset,
  weights,
  na.action,
  method = "qr",
  model = TRUE,
  x = FALSE,
```

```
  y = FALSE,
  qr = TRUE,
  singular.ok = TRUE,
  contrasts = NULL,
  offset,
  ...,
  selectFun = logLik,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)

flm.fit(
  x,
  y,
  offset = NULL,
  method = "qr",
  tol = 1e-07,
  singular.ok = TRUE,
  ...,
  selectFun = "logLik",
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

formula	Parameter passed to lm.
data	Parameter passed to lm.
subset	Parameter passed to lm.
weights	Parameter passed to lm.
na.action	Parameter passed to lm.
method	Parameter passed to lm.
model	Parameter passed to lm.
x	Parameter passed to lm.
y	Parameter passed to lm.
qr	Parameter passed to lm.
singular.ok	Parameter passed to lm.
contrasts	Parameter passed to lm.
offset	Parameter passed to lm or lm.fit.
...	Parameters passed to lm or lm.fit.
selectFun	Parameter passed to frs.

stopFun	Parameter passed to frs.
keep	Parameter passed to frs.
maxK	Parameter passed to frs.
verbose	Parameter passed to frs.
tol	Parameter passed to lm.fit.

Value

A lm model object fitted on the selected features.

---

frq *Forward Regression Selection for Quantile Regression Models*

Description

frq() inherits the usage of the function quantreg::rq, and performs forward regression selection for quantile regression models.

Usage

```
frq(
  formula,
  tau = 0.5,
  data,
  subset,
  weights,
  na.action,
  method = "br",
  model = TRUE,
  contrasts = NULL,
  ...,
  selectFun = logLik,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)

frq.fit(
  x,
  y,
  tau = 0.5,
  method = "br",
  ...,
  selectFun = "logLik",
  stopFun = "EBIC",
```

```
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

formula	Parameter passed to quantreg::rq.
tau	Parameter passed to quantreg::rq.
data	Parameter passed to quantreg::rq.
subset	Parameter passed to quantreg::rq.
weights	Parameter passed to quantreg::rq.
na.action	Parameter passed to quantreg::rq.
method	Parameter passed to quantreg::rq.
model	Parameter passed to quantreg::rq.
contrasts	Parameter passed to quantreg::rq.
...	Parameters passed to quantreg::rq.
selectFun	Parameter passed to frs.
stopFun	Parameter passed to frs.
keep	Parameter passed to frs.
maxK	Parameter passed to frs.
verbose	Parameter passed to frs.
x	Parameter passed to quantreg::rq.fit.
y	Parameter passed to quantreg::rq.fit.

Value

A rq model object fitted on the selected features.

NULL

---

frs *Forward Regression Selection Framework*

Description

frs() is a generic workhorse function of forward regression selection for parametric regression.

Usage

```
frs(
  xmat,
  yvec,
  fitFun,
  ...,
  use.formula = TRUE,
  use.intercept = TRUE,
  selectFun = logLik,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

xmat	See pboost.
yvec	See pboost.
fitFun	See pboost.
...	See pboost.
use.formula	See pboost.
use.intercept	See pboost.
selectFun	A function to evaluate the importance of an unselected feature when it is added to current model. The default is logLik, meaning that the feature with the largest post-added log-likelihood is identified as the next one to be added to the model. Note that selectFun is only used for selecting features, and it does not affect the stopping rule of forward regression selection, which is determined by stopFun.
stopFun	See pboost.
keep	See pboost.
maxK	See pboost.
verbose	See pboost.

Value

Model object fitted on the selected features.

See Also

fbetareg, fcoxph, fglm, flm, frq, fsar.

pbetareg *Profile Boosting for Beta Regression*

Description

pbetareg inherits the usage of betareg::betareg.

Usage

```
pbetareg(
  formula,
  data,
  subset,
  na.action,
  weights,
  offset,
  link = c("logit", "probit", "cloglog", "cauchit", "log", "loglog"),
  link.phi = NULL,
  type = c("ML", "BC", "BR"),
  dist = NULL,
  nu = NULL,
  control = betareg.control(...),
  model = TRUE,
  y = TRUE,
  x = FALSE,
  ...,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

formula	Parameter passed to betareg::betareg.
data	Parameter passed to betareg::betareg.
subset	Parameter passed to betareg::betareg.
na.action	Parameter passed to betareg::betareg.
weights	Parameter passed to betareg::betareg.
offset	Parameter passed to betareg::betareg.
link	Parameter passed to betareg::betareg.
link.phi	Parameter passed to betareg::betareg.
type	Parameter passed to betareg::betareg.
dist	Parameter passed to betareg::betareg.

nu	Parameter passed to betareg::betareg.
control	Parameter passed to betareg::betareg.
model	Parameter passed to betareg::betareg.
y	Parameter passed to betareg::betareg.
x	Parameter passed to betareg::betareg.
...	Parameters passed to betareg::betareg.
stopFun	Parameter passed to pboost.
keep	Parameter passed to pboost.
maxK	Parameter passed to pboost.
verbose	Parameter passed to pboost.

Value

A betareg model object fitted on the selected features.

Examples

```
## Not run:
set.seed(2026)
n <- 300
p <- 20
x <- matrix(runif(n*p), n)
mu <- runif(n)
phi <- 1.0

shape1 <- mu * phi
shape2 <- (1-mu) * phi
y <- rbeta(n, shape1, shape2)
DF <- data.frame(y, x)

pbetareg(y ~ ., DF, verbose=TRUE)
fbetareg(y ~ ., DF, verbose=TRUE)

## End(Not run)
```

---

pboost *Profile Boosting Framework*

Description

pboost() is the generic workhorse function of profile boosting framework for parametric regression.

Usage

```
pboost(
  xmat,
  yvec,
  fitFun,
  scoreFun,
  stopFun = "EBIC",
  ...,
  use.formula = TRUE,
  use.intercept = TRUE,
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

xmat	Numeric feature matrix.
yvec	Response vector.
fitFun	Function to fit the empirical risk function in the form `fitFun(formula, data, ...)`.
scoreFun	Function to compute the derivative, denoted by $\frac{\partial \ell(y,\eta)}{\partial \eta}$, of empirical risk function in the form `scoreFun(object)`, where `object` is returned by `fitFun`. `scoreFun()` should return a vector with the same length of `y` in `data`.
stopFun	Stopping rule for profile boosting, which has the form `stopFun(object)` to evaluate the performance of model `object` returned by `fitFun`, such as EBIC or BIC.
...	Additional arguments to be passed to `fitFun`.
use.formula	Whether to use formula interface for model fitting. Default to `TRUE`. When `use.formula=TRUE`, the the model fitting function has the form `fitFun(formula, data, ...)`; otherwise, `fitFun(x, y, ...)`.
use.intercept	Include intercept in the model fitting? Valid only when `use.formula=TRUE`.
keep	Vector of indices or feature names initial features to include.
maxK	Maximal number of identified features. If `maxK` is specified, it will suppress `stopFun`, saying that the profile boosting continues until the procedure identifies `maxK` features. The pre-specified features in `keep` are counted toward `maxK`.
verbose	Print the procedure path?

Value

Model object fitted on the selected features.

See Also

pbetareg, pcoxph, pglm, plm, prq, psar.

pcoxph *Profile Boosting for Cox proportional hazards Model*

Description

Profile boosting for Cox model.

Usage

```
pcoxph(
  formula,
  data,
  weights,
  subset,
  na.action,
  init,
  control,
  ties = c("efron", "breslow", "exact"),
  singular.ok = TRUE,
  robust,
  model = FALSE,
  x = FALSE,
  y = TRUE,
  tt,
  method = ties,
  id,
  cluster,
  istate,
  statedata,
  nocenter = c(-1, 0, 1),
  ...,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

formula	Parameter passed to survival::coxph.
data	Parameter passed to survival::coxph.
weights	Parameter passed to survival::coxph.
subset	Parameter passed to survival::coxph.
na.action	Parameter passed to survival::coxph.
init	Parameter passed to survival::coxph.

control	Parameter passed to survival::coxph.
ties	Parameter passed to survival::coxph.
singular.ok	Parameter passed to survival::coxph.
robust	Parameter passed to survival::coxph.
model	Parameter passed to survival::coxph.
x	Parameter passed to survival::coxph.
y	Parameter passed to survival::coxph.
tt	Parameter passed to survival::coxph.
method	Parameter passed to survival::coxph.
id	Parameter passed to survival::coxph.
cluster	Parameter passed to survival::coxph.
istate	Parameter passed to survival::coxph.
statedata	Parameter passed to survival::coxph.
nocenter	Parameter passed to survival::coxph.
...	Parameters passed to survival::coxph.
stopFun	Parameter passed to pboost.
keep	Parameter passed to pboost.
maxK	Parameter passed to pboost.
verbose	Parameter passed to pboost.

Value

A coxph model object fitted on the selected features.

Examples

```
library(survival)
set.seed(2026)
n <- 300
p <- 200

DF <- data.frame(
    time = rpois(n, 5),
    status = rbinom(n, 1, 0.3),
    matrix(rnorm(n*p), n)
)

pcoxph(Surv(time, status) ~ ., DF, verbose=TRUE)
fcoxph(Surv(time, status) ~ ., DF, verbose=TRUE)
```

`penalization` *Penalized Methods for Feature Selection*

Description

Penalized methods for feature selection. These functions are only designed for comparison of numerical simulations.

- `pen_glmnet(x, y, family)`: LASSO penalized models.
- `pen_ncvreg(x, y, family, penalty)`: Non-Convex penalized models.
- `pen_rq(x, y, tau, penalty)`: penalized quantile regression models.
- `pen_sar(x, y, rho, w, penalty)`: penalized spatial auto-regressive models.

Usage

```
pen_glmnet(x, y, family)

pen_ncvreg(x, y, family, penalty)

pen_rq(x, y, tau, penalty)

pen_sar(x, y, rho, w, penalty)
```

Arguments

`x`	Feature matrix
`y`	Response vector.
`family`	"gaussian", "binomial", or "cox".
`penalty`	"lasso", "SCAD" or "MCP".
`tau`	Quantiles to be modeled.
`rho`	Spatial autoregressive parameter. If missing or NULL, it will be estimated.
`w`	Weight matrix (row-sum scaled being one).

Value

Set of selected features.

Examples

```
library(survival)
set.seed(2026)
n <- 10
p <- 20

x <- replicate(p, rnorm(n))
b0 <- runif(3, 1.5, 2.0)
eta <- drop(x[, 1:3] %*% b0)

## ---------- linear ----------
y <- rnorm(n, eta)

pen_glmnet(x=x, y=y, family="gaussian") |> coef()
pen_ncvreg(x=x, y=y, family="gaussian", penalty="MCP") |> coef()
pen_ncvreg(x=x, y=y, family="gaussian", penalty="SCAD") |> coef()


## ---------- logistic ----------
y <- rbinom(n, 1, 1.0 / (1.0 + exp(-eta)))

pen_glmnet(x=x, y=y, family="binomial") |> coef()
pen_ncvreg(x=x, y=y, family="binomial", penalty="MCP") |> coef()
pen_ncvreg(x=x, y=y, family="binomial", penalty="SCAD") |> coef()


## ---------- cox ----------
h0 <- 0.01
censoringRate <- 0.3
survivalTime <- -log(runif(n)) / (h0 * exp(eta))
censoringTime <- rexp(n, rate = -log(1 - censoringRate)/median(survivalTime))
y <- cbind(
    time = pmin(survivalTime, censoringTime),
    status = as.numeric(survivalTime <= censoringTime)
)

pen_glmnet(x=x, y=y, family="cox") |> coef()
pen_ncvreg(x=x, y=y, family="cox", penalty="MCP") |> coef()
pen_ncvreg(x=x, y=y, family="cox", penalty="SCAD") |> coef()


## ---------- quantile ----------
y <- eta + rt(n, 2)

pen_rq(x=x, y=y, tau=0.5, penalty="lasso") |> coef()
pen_rq(x=x, y=y, tau=0.5, penalty="MCP") |> coef()
pen_rq(x=x, y=y, tau=0.5, penalty="SCAD") |> coef()


## ---------- sar ----------
w0 <- set_rook_matrix(5, n/5)
rho0 <- 0.5
```

```
y <- solve(diag(n) - rho0 * w0, rnorm(n, eta))

pen_sar(x=x, y=y, rho=rho0, w=w0, penalty="lasso") |> coef()
pen_sar(x=x, y=y, rho=rho0, w=w0, penalty="MCP") |> coef()
pen_sar(x=x, y=y, rho=rho0, w=w0, penalty="SCAD") |> coef()
```

---

pggm *Profile Boosting for Gaussian Graphical Model*

Description

Profile boosting for Gaussian graphical model.

Usage

```
pggm(
  S,
  nObs,
  maxK = floor(min(nObs - 1, NROW(S) - 1, 50)),
  digits = 8,
  verbose = FALSE
)
```

Arguments

S	Covariance matrix.
nObs	Number of observations.
maxK	Maximum number of identified edges.
digits	Integer indicating the number of decimal places or significant digits to be used.
verbose	Print the procedure path?

Value

Index set of identified features.

Examples

```
library(MASS)
library(Matrix)

set.seed(2025)
n <- 1000
p <- 10

Omega <- Diagonal(p)
diag(Omega[1:4, 2:5]) <- diag(Omega[2:5, 1:4]) <- 0.5
Sigma <- chol2inv(chol(Omega))
```

```
X <- mvrnorm(n, rep(0, p), Sigma, empirical=TRUE)
S <- cov(X)
system.time( egg <- pggm(S, n) )
```

---

pglm *Profile Boosting for Generalized Linear Models.*

Description

pglm inherits the usage of the built-in function glm.

Usage

```
pglm(
  formula,
  family = gaussian,
  data,
  weights,
  subset,
  na.action,
  start = NULL,
  etastart,
  mustart,
  offset,
  control = list(...),
  model = TRUE,
  method = "glm.fit",
  x = FALSE,
  y = TRUE,
  singular.ok = TRUE,
  contrasts = NULL,
  ...,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)

pglm.fit(
  x,
  y,
  weights = rep.int(1, NROW(y)),
  start = NULL,
  etastart = NULL,
  mustart = NULL,
  offset = rep.int(0, NROW(y)),
```

```
  family = gaussian(),
  control = list(),
  intercept = TRUE,
  singular.ok = TRUE,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

formula	Parameter passed to glm.
family	Parameter passed to glm.
data	Parameter passed to glm.
weights	Parameter passed to glm.
subset	Parameter passed to glm.
na.action	Parameter passed to glm.
start	Parameter passed to glm.
etastart	Parameter passed to glm.
mustart	Parameter passed to glm.
offset	Parameter passed to glm.
control	Parameter passed to glm.
model	Parameter passed to glm.
method	Parameter passed to glm.
x	Parameter passed to glm.
y	Parameter passed to glm.
singular.ok	Parameter passed to glm.
contrasts	Parameter passed to glm.
...	Parameters passed to glm.
stopFun	Parameter passed to pboost.
keep	Parameter passed to pboost.
maxK	Parameter passed to pboost.
verbose	Parameter passed to pboost.
intercept	Parameter passed to glm.fit.

Value

A glm model object fitted on the selected features.

References

Zengchao Xu, Shan Luo and Zehua Chen (2022). Partial profile score feature selection in high-dimensional generalized linear interaction models. Statistics and Its Interface. doi:10.4310/21-SII706

Examples

```
set.seed(2026)
n <- 200
p <- 100
x <- matrix(rnorm(n*p), n)

eta <- drop( x[, 1:3] %*% runif(3, 1.0, 1.5) )
y <- rbinom(n, 1, 1/(1+exp(-eta)))
DF <- data.frame(y, x)

## ---------- pboost ----------
pglm(y ~ ., "binomial", DF, verbose=TRUE)
pglm(y ~ ., "binomial", DF, stopFun=BIC, verbose=TRUE)

scoreLogistic <- function(object) {
   eta.hat <- object[["linear.predictors"]]
   return(object[["y"]] - 1/(1+exp(-eta.hat)))
}
(result <- pboost(x, y, glm, scoreLogistic, family="binomial", verbose=TRUE))
all.vars(formula(result)[[3]])

## ---------- frs ----------
fglm(y ~ ., "binomial", DF, verbose=TRUE)
fglm(y ~ ., "binomial", DF, stopFun=BIC, verbose=TRUE)

frs(x, y, glm, family="binomial", verbose=TRUE)
```

`plagsarlm` *Profile Boosting for Spatial Auto-regressive Model*

Description

Profile boosting variable selection for spatial auto-regressive (SAR) model

$$\boldsymbol{y} = \rho W \boldsymbol{y} + X\beta + \boldsymbol{\varepsilon},$$

where $\boldsymbol{y}, \boldsymbol{\varepsilon} \in \mathbb{R}^n$, $X \in \mathbb{R}^{n\times p}$, $\rho \in (-1, 1)$.

Usage

```
plagsarlm(
  formula,
```

```
  data = list(),
  listw,
  na.action,
  Durbin,
  type,
  method = "eigen",
  quiet = NULL,
  zero.policy = NULL,
  interval = NULL,
  tol.solve = .Machine$double.eps,
  trs = NULL,
  control = list(),
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)

psar.fit(x, y, w, stopFun = "EBIC", keep = NULL, maxK = NULL, verbose = FALSE)
```

Arguments

formula	Parameters passed to spatialreg::lagsarlm.
data	Parameters passed to spatialreg::lagsarlm.
listw	Parameters passed to spatialreg::lagsarlm.
na.action	Parameters passed to spatialreg::lagsarlm.
Durbin	Parameters passed to spatialreg::lagsarlm.
type	Parameters passed to spatialreg::lagsarlm.
method	Parameters passed to spatialreg::lagsarlm.
quiet	Parameters passed to spatialreg::lagsarlm.
zero.policy	Parameters passed to spatialreg::lagsarlm.
interval	Parameters passed to spatialreg::lagsarlm.
tol.solve	Parameters passed to spatialreg::lagsarlm.
trs	Parameters passed to spatialreg::lagsarlm.
control	Parameters passed to spatialreg::lagsarlm.
stopFun	Parameter passed to pboost.
keep	Parameter passed to pboost.
maxK	Parameter passed to pboost.
verbose	Parameter passed to pboost.
x	Numeric feature matrix.
y	Response vector.
w	Weight matrix (row-sum scaled being one).

Value

Model object fitted on the selected features.

Examples

```
set.seed(2026)

w <- set_rook_matrix(15, 15)

n <- NROW(w)
p <- 30
x <- matrix(rnorm(n*p), n) %*% chol(0.7^abs(outer(1:p, 1:p, "-")))
eta <- drop(x[, 1:3] %*% runif(3, 1.5, 2.0))

rho0 <- 0.2
sig0 <- 1.0
y <- solve(diag(n) - rho0 * w, rnorm(n, eta, sd=sig0)) |> drop()

## ---------- pboost ----------
system.time( egg <- psar.fit(x, y, w, verbose=TRUE) )
y.tilde <- (diag(NROW(x)) - egg[["rho"]] * w) %*% y

beta.hat <- egg[["beta"]]
idx <- as.integer(sub("[[:alpha:]]", "", names(beta.hat)))
sig2.hat <- mean( (y.tilde - drop(x[, idx, drop=FALSE] %*% beta.hat))^2 )
print( egg[["sig2"]] - sig2.hat )

## Not run:
system.time(
plagsarlm(y ~ ., data=data.frame(y, x), listw=spdep::mat2listw(w, style="W"), verbose=TRUE)
)

## End(Not run)

## ---------- frs ----------
fsar.fit(x, y, w, verbose=TRUE)

## Not run:
system.time(
flagsarlm(y ~ ., data=data.frame(y, x), listw=spdep::mat2listw(w, style="W"), verbose=TRUE)
)

## End(Not run)
```

---

plm *Profile Boosting for Linear Models.*

Description

plm inherits the usage of the built-in function lm.

Usage

```
plm(
  formula,
  data,
  subset,
  weights,
  na.action,
  method = "qr",
  model = TRUE,
  x = FALSE,
  y = FALSE,
  qr = TRUE,
  singular.ok = TRUE,
  contrasts = NULL,
  offset,
  ...,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)

plm.fit(
  x,
  y,
  offset = NULL,
  method = "qr",
  tol = 1e-07,
  singular.ok = TRUE,
  ...,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

formula	Parameter passed to lm.
data	Parameter passed to lm.
subset	Parameter passed to lm.
weights	Parameter passed to lm.
na.action	Parameter passed to lm.
method	Parameter passed to lm.
model	Parameter passed to lm.
x	Parameter passed to lm.fit.

y	Parameter passed to lm.fit.
qr	Parameter passed to lm.
singular.ok	Parameter passed to lm.
contrasts	Parameter passed to lm.
offset	Parameter passed to lm or lm.fit.
...	Parameters passed to lm or lm.fit.
stopFun	Parameter passed to pboost.
keep	Parameter passed to pboost.
maxK	Parameter passed to pboost.
verbose	Parameter passed to pboost.
tol	Parameter passed to lm.fit.

Details

plm is an equivalent implementation to the sequential lasso method proposed by Luo and Chen(2014, doi:10.1080/01621459.2013.877275).

Value

A lm model object fitted on the selected features.

References

- Zengchao Xu, Shan Luo and Zehua Chen (2022). Partial profile score feature selection in high-dimensional generalized linear interaction models. Statistics and Its Interface. doi:10.4310/21SII706
- Shan Luo and Zehua Chen (2014). A Sequential Lasso Method for Feature Selection with Ultra-High Dimensional Feature Space. Journal of the American Statistical Association, 109(507):223–232. doi:10.1080/01621459.2013.877275

Examples

```
set.seed(2026)
n <- 300
p <- 200
x <- matrix(rnorm(n*p), n)

eta <- drop( x[, 1:3] %*% runif(3, 1.0, 1.5) )
y <- rnorm(n, eta, sd=sd(eta))
DF <- data.frame(y, x)

plm(y ~ ., DF, verbose=TRUE)
plm(y ~ ., DF, stopFun=BIC, verbose=TRUE)
pboost(x, y, lm, residuals, verbose=TRUE)

flm(y ~ ., DF, verbose=TRUE)
flm(y ~ ., DF, stopFun=BIC, verbose=TRUE)
frs(x, y, lm, verbose=TRUE)
```

prq *Profile Boosting for Quantile Regression Models*

Description

prq inherits the usage of the function quantreg::rq.

Usage

```
prq(
  formula,
  tau = 0.5,
  data,
  subset,
  weights,
  na.action,
  method = "br",
  model = TRUE,
  contrasts = NULL,
  ...,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)

prq.fit(
  x,
  y,
  tau = 0.5,
  method = "br",
  ...,
  stopFun = "EBIC",
  keep = NULL,
  maxK = NULL,
  verbose = TRUE
)
```

Arguments

formula	Parameter passed to quantreg::rq.
tau	Parameter passed to quantreg::rq.
data	Parameter passed to quantreg::rq.
subset	Parameter passed to quantreg::rq.
weights	Parameter passed to quantreg::rq.

na.action	Parameter passed to quantreg::rq.
method	Parameter passed to quantreg::rq or quantreg::rq.fit.
model	Parameter passed to quantreg::rq.
contrasts	Parameter passed to quantreg::rq.
...	Parameters passed to quantreg::rq or quantreg::rq.fit.
stopFun	Parameter passed to pboost.
keep	Parameter passed to pboost.
maxK	Parameter passed to pboost.
verbose	Parameter passed to pboost.
x	Parameter passed to quantreg::rq.fit.
y	Parameter passed to quantreg::rq.fit.

Value

A rq model object fitted on the selected features.

Examples

```
library(quantreg)
set.seed(2026)
n <- 300
p <- 20
x <- matrix(rnorm(n*p), n)

eta <- drop( x[, 1:3] %*% runif(3, 1.0, 1.5) )
y <- eta + (1.0 + x[, 3]) * rnorm(n)
DF <- data.frame(y, x)

tau <- 0.5
prq(y ~ ., tau, DF, verbose=TRUE)

BIC <- function(obj) AIC(obj, k=-1)
prq(y ~ ., tau, DF, stopFun=BIC, verbose=TRUE)
frq(y ~ ., tau, DF, stopFun=BIC, verbose=TRUE)

scoreFun <- function(object)
   return(ifelse(object[["y"]] < fitted(object), tau - 1, tau))

pboost(x, y, rq, scoreFun, BIC, tau=tau, verbose=TRUE)

prq.fit(x, y, verbose=TRUE)
frq.fit(x, y, verbose=TRUE)
```

sar.model *Functions for Spatial Auto-regressive Model*

Description

- `get_rho()`: MLE of ρ in SAR model. See psar.
- `sar.fit()`: Fit SAR model with given ρ.
- `set_rook_matrix()`: Construct Rook weight adjacency matrix.
- `logLik()`, `BIC()`, `residuals()`, `coef()`.

Usage

```
get_rho(x, y, w)

sar.fit(x, y, w)

## S3 method for class 'sarpboost'
residuals(object, ...)

## S3 method for class 'sarpboost'
logLik(object, ...)

## S3 method for class 'sarpboost'
BIC(object, ...)

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

set_rook_matrix(rNum, cNum)
```

Arguments

`x`	Numeric feature matrix.
`y`	Response vector.
`w`	Weight matrix (row-sum scaled).
`object`	Object.
`...`	Arguments.
`rNum`	Number of row units in Rook structure.
`cNum`	Number of column units in Rook structure.

Value

Value or list.

Index