Package ‘pboost’

January 8, 2026

Title Profile Boosting Framework for Parametric Models

Version 0.2.1

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.
An S3 interface EBIC() is provided as the stopping rule for the profile boosting by default.

Imports stats, Matrix, MASS, betareg, quantreg, survival, Formula

License GPL (>= 3)

Encoding UTF-8

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

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

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

NeedsCompilation no

Author Zengchao Xu [aut, cre]

Repository CRAN

Date/Publication 2026-01-08 18:50:02 UTC

Contents

EBIC 2
pbetareg 3
pboost 5
pcoxph 6
pggm 8
pglm 9
plm 11
prq 13

Index **15**

`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, ...)

## S3 method for class 'betareg'
EBIC(object, p, p.keep, ...)

## S3 method for class 'coxph'
EBIC(object, p, p.keep, ...)

## S3 method for class 'glm'
EBIC(object, p, p.keep, ...)

## S3 method for class 'lm'
EBIC(object, p, p.keep, ...)

## S3 method for class 'rq'
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

See Also

plm, pglm, pcoxph, prq, pbetareg.

---

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`	See pboost.
`data`	See pboost.

`subset`	Parameters passed to betareg::betareg.
`na.action`	Parameters passed to betareg::betareg.
`weights`	Parameters passed to betareg::betareg.
`offset`	Parameters passed to betareg::betareg.
`link`	Parameters passed to betareg::betareg.
`link.phi`	Parameters passed to betareg::betareg.
`type`	Parameters passed to betareg::betareg.
`dist`	Parameters passed to betareg::betareg.
`nu`	Parameters passed to betareg::betareg.
`control`	Parameters passed to betareg::betareg.
`model`	Parameters passed to betareg::betareg.
`y`	Parameters passed to betareg::betareg.
`x`	Parameters passed to betareg::betareg.
`...`	Parameters passed to betareg::betareg.
`stopFun`	Parameters passed to pboost.
`keep`	Parameters passed to pboost.
`maxK`	Parameters passed to pboost.
`verbose`	Parameters passed to pboost.

Value

An `betareg` model object fitted on the selected features.

Examples

```
library(betareg)
set.seed(2025)
n <- 300
p <- 100
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)
```

`pboost` *Profile Boosting Framework*

Description

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

Usage

```
pboost(
  formula,
  data,
  fitFun,
  scoreFun,
  stopFun,
  ...,
  keep = NULL,
  maxK = NULL,
  verbose = FALSE
)
```

Arguments

`formula`	An object of class formula of the form `LHS ~ RHS`, where the right-hand side (RHS) specifies the candidate features for the linear predictor $\eta = \sum_j \beta_j x_j$. The following restrictions and recommendations apply: • All variables appearing on the RHS must be numeric in the supplied `data` • For computational efficiency, each term on the RHS must correspond to a single column in the resulting model matrix. Supported expressions include main effects (`x1`), interactions (`x1:x2`), and simple transformations (`log(x1)`, `I(x1^2)`, etc.). Complex terms that expand into multiple columns—such as `poly(x, degree)`, `bs(x)`, or `ns(x)`—are **not supported**. • Offset terms should not be included in the formula. Instead, provide them via the dedicated `offset` argument of `fitFun`.
`data`	An data frame containing the variables in the model.
`fitFun`	Function to fit the empirical risk function in the form `fitFun(formula, data, ...)`.
`scoreFun`	Function to compute the derivative 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`.

keep	Initial set of features that are included in model fitting. **If keep is specified, it should also be fully included in the RHS of formula.**
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.

Examples

```
set.seed(2025)
n <- 200
p <- 300
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)

scoreLogistic <- function(object) {
    eta.hat <- object[["linear.predictors"]]
    return(object[["y"]] - 1/(1+exp(-eta.hat)))
}

( result <- pboost(y~., DF, glm, scoreLogistic, EBIC, family="binomial") )

attr(terms(formula(result), data=DF), "term.labels")
```

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`	See pboost.
`data`	See pboost.
`weights`	Parameters passed to survival::coxph.
`subset`	Parameters passed to survival::coxph.
`na.action`	Parameters passed to survival::coxph.
`init`	Parameters passed to survival::coxph.
`control`	Parameters passed to survival::coxph.
`ties`	Parameters passed to survival::coxph.
`singular.ok`	Parameters passed to survival::coxph.
`robust`	Parameters passed to survival::coxph.
`model`	Parameters passed to survival::coxph.
`x`	Parameters passed to survival::coxph.
`y`	Parameters passed to survival::coxph.
`tt`	Parameters passed to survival::coxph.
`method`	Parameters passed to survival::coxph.
`id`	Parameters passed to survival::coxph.
`cluster`	Parameters passed to survival::coxph.
`istate`	Parameters passed to survival::coxph.
`statedata`	Parameters passed to survival::coxph.
`nocenter`	Parameters passed to survival::coxph.
`...`	Parameters passed to survival::coxph.

`stopFun`	Parameters passed to pboost.
`keep`	Parameters passed to pboost.
`maxK`	Parameters passed to pboost.
`verbose`	Parameters passed to pboost.

Value

An coxph model object fitted on the selected features.

Examples

```
library(survival)
set.seed(2025)
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)
```

`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
)
```

Arguments

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

Value

An `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(2025)
n <- 300
p <- 200
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)

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)))
}
pboost(y ~ ., DF, glm, scoreLogistic, EBIC, family="binomial", verbose=TRUE)
```

`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
)
```

Arguments

`formula`	See pboost.
`data`	See pboost.
`subset`	Parameters passed to lm.
`weights`	Parameters passed to lm.
`na.action`	Parameters passed to lm.
`method`	Parameters passed to lm.
`model`	Parameters passed to lm.
`x`	Parameters passed to lm.
`y`	Parameters passed to lm.
`qr`	Parameters passed to lm.
`singular.ok`	Parameters passed to lm.
`contrasts`	Parameters passed to lm.
`offset`	Parameters passed to lm.
`...`	Parameters passed to lm.
`stopFun`	Parameters passed to pboost.
`keep`	Parameters passed to pboost.
`maxK`	Parameters passed to pboost.
`verbose`	Parameters passed to pboost.

Details

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

Value

An `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(2025)
n <- 300
p <- 200
x <- matrix(rnorm(n*p), n)

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

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

---

prq *Profile Boosting for Quantile Regression*

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
)
```

Arguments

formula	See pboost.
tau	Parameters passed to quantreg::rq.
data	See pboost.
subset	Parameters passed to quantreg::rq.

`weights`	Parameters passed to quantreg::rq.
`na.action`	Parameters passed to quantreg::rq.
`method`	Parameters passed to quantreg::rq.
`model`	Parameters passed to quantreg::rq.
`contrasts`	Parameters passed to quantreg::rq.
`...`	Parameters passed to quantreg::rq.
`stopFun`	Parameters passed to pboost.
`keep`	Parameters passed to pboost.
`maxK`	Parameters passed to pboost.
`verbose`	Parameters passed to pboost.

Value

An `rq` model object fitted on the selected features.

Examples

```
library(quantreg)
set.seed(2025)
n <- 300
p <- 200
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)

scorerq <- function(object) {
 return(ifelse(object[["y"]] < fitted(object), tau - 1, tau))
}
pboost(y ~ ., DF, rq, scorerq, EBIC, tau=tau, verbose=TRUE)
```

Index

betareg::betareg, *3, 4*
BIC, *5*

EBIC, 2, *5*

formula, *5*

glm, *2*, *9, 10*

lm, *2*, *11*, *12*
logLik, *2*

pbetareg, *3*, 3
pboost, *2–4*, 5, *7, 8*, *10*, *12–14*
pcoxph, *3*, 6
pggm, 8
pglm, *3*, *9*, 9
plm, *3*, *11*, 11
prq, *3*, *13*, 13

quantreg::rq, *13, 14*

survival::coxph, *7*