Package 'msaenet'

March 4, 2024

Type Package

Title Multi-Step Adaptive Estimation Methods for Sparse Regressions

Version 3.1.1

Maintainer Nan Xiao <me@nanx.me>

Description Multi-step adaptive elastic-net (MSAENet) algorithm for feature selection in high-dimensional regressions proposed in Xiao and Xu (2015) <DOI:10.1080/00949655.2015.1016944>, with support for multi-step adaptive MCP-net (MSAMNet) and multi-step adaptive SCAD-net (MSASNet) methods.

License GPL (>= 3)

URL https://nanx.me/msaenet/, https://github.com/nanxstats/msaenet

Encoding UTF-8

VignetteBuilder knitr

BugReports https://github.com/nanxstats/msaenet/issues

Depends R (>= 3.0.2)

Imports glmnet, ncvreg (>= 3.8-0), foreach, mvtnorm, survival, Matrix

Suggests knitr, rmarkdown, doParallel

RoxygenNote 7.3.1

NeedsCompilation no

Author Nan Xiao [aut, cre] (<https://orcid.org/0000-0002-0250-5673>), Qing-Song Xu [aut]

Repository CRAN

Date/Publication 2024-03-04 04:40:02 UTC

R topics documented:

`aenet` *Adaptive Elastic-Net*

Description

Adaptive Elastic-Net

Usage

```
aenet(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "cox"),
  init = c("enet", "ridge"),
  alphas = seq(0.05, 0.95, 0.05),
  tune = c("cv", "ebic", "bic", "aic"),
  nfolds = 5L,
  rule = c("lambda.min", "lambda.1se"),
  ebic.gamma = 1,
  scale = 1,
  lower.limits = -Inf,
  upper.limits = Inf,
  penalty.factor.init = rep(1, ncol(x)),
  seed = 1001,
  parallel = FALSE,
```

```
  verbose = FALSE
)
```

Arguments

x	Data matrix.
y	Response vector if family is "gaussian", "binomial", or "poisson". If family is "cox", a response matrix created by Surv.
family	Model family, can be "gaussian", "binomial", "poisson", or "cox".
init	Type of the penalty used in the initial estimation step. Can be "enet" or "ridge".
alphas	Vector of candidate alphas to use in cv.glmnet.
tune	Parameter tuning method for each estimation step. Possible options are "cv", "ebic", "bic", and "aic". Default is "cv".
nfolds	Fold numbers of cross-validation when tune = "cv".
rule	Lambda selection criterion when tune = "cv", can be "lambda.min" or "lambda.1se". See cv.glmnet for details.
ebic.gamma	Parameter for Extended BIC penalizing size of the model space when tune = "ebic", default is 1. For details, see Chen and Chen (2008).
scale	Scaling factor for adaptive weights: weights = coefficients^(-scale).
lower.limits	Lower limits for coefficients. Default is -Inf. For details, see glmnet.
upper.limits	Upper limits for coefficients. Default is Inf. For details, see glmnet.
penalty.factor.init	The multiplicative factor for the penalty applied to each coefficient in the initial estimation step. This is useful for incorporating prior information about variable weights, for example, emphasizing specific clinical variables. To make certain variables more likely to be selected, assign a smaller value. Default is rep(1, ncol(x)).
seed	Random seed for cross-validation fold division.
parallel	Logical. Enable parallel parameter tuning or not, default is FALSE. To enable parallel tuning, load the doParallel package and run registerDoParallel() with the number of CPU cores before calling this function.
verbose	Should we print out the estimation progress?

Value

List of model coefficients, glmnet model object, and the optimal parameter set.

Author(s)

Nan Xiao <https://nanx.me>

References

Zou, Hui, and Hao Helen Zhang. (2009). On the adaptive elastic-net with a diverging number of parameters. *The Annals of Statistics* 37(4), 1733–1751.

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

aenet.fit <- aenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2), seed = 1002
)

print(aenet.fit)
msaenet.nzv(aenet.fit)
msaenet.fp(aenet.fit, 1:5)
msaenet.tp(aenet.fit, 1:5)
aenet.pred <- predict(aenet.fit, dat$x.te)
msaenet.rmse(dat$y.te, aenet.pred)
plot(aenet.fit)
```

---

amnet *Adaptive MCP-Net*

Description

Adaptive MCP-Net

Usage

```
amnet(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "cox"),
  init = c("mnet", "ridge"),
  gammas = 3,
  alphas = seq(0.05, 0.95, 0.05),
  tune = c("cv", "ebic", "bic", "aic"),
  nfolds = 5L,
  ebic.gamma = 1,
  scale = 1,
  eps = 1e-04,
  max.iter = 10000L,
  penalty.factor.init = rep(1, ncol(x)),
  seed = 1001,
  parallel = FALSE,
  verbose = FALSE
)
```

Arguments

x	Data matrix.
y	Response vector if family is "gaussian", "binomial", or "poisson". If family is "cox", a response matrix created by Surv.
family	Model family, can be "gaussian", "binomial", "poisson", or "cox".
init	Type of the penalty used in the initial estimation step. Can be "mnet" or "ridge".
gammas	Vector of candidate gammas (the concavity parameter) to use in MCP-Net. Default is 3.
alphas	Vector of candidate alphas to use in MCP-Net.
tune	Parameter tuning method for each estimation step. Possible options are "cv", "ebic", "bic", and "aic". Default is "cv".
nfolds	Fold numbers of cross-validation when tune = "cv".
ebic.gamma	Parameter for Extended BIC penalizing size of the model space when tune = "ebic", default is 1. For details, see Chen and Chen (2008).
scale	Scaling factor for adaptive weights: weights = coefficients^(-scale).
eps	Convergence threshhold to use in MCP-net.
max.iter	Maximum number of iterations to use in MCP-net.
penalty.factor.init	The multiplicative factor for the penalty applied to each coefficient in the initial estimation step. This is useful for incorporating prior information about variable weights, for example, emphasizing specific clinical variables. To make certain variables more likely to be selected, assign a smaller value. Default is rep(1, ncol(x)).
seed	Random seed for cross-validation fold division.
parallel	Logical. Enable parallel parameter tuning or not, default is FALSE. To enable parallel tuning, load the doParallel package and run registerDoParallel() with the number of CPU cores before calling this function.
verbose	Should we print out the estimation progress?

Value

List of model coefficients, ncvreg model object, and the optimal parameter set.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)
```

```
amnet.fit <- amnet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2), seed = 1002
)

print(amnet.fit)
msaenet.nzv(amnet.fit)
msaenet.fp(amnet.fit, 1:5)
msaenet.tp(amnet.fit, 1:5)
amnet.pred <- predict(amnet.fit, dat$x.te)
msaenet.rmse(dat$y.te, amnet.pred)
plot(amnet.fit)
```

`asnet` *Adaptive SCAD-Net*

Description

Adaptive SCAD-Net

Usage

```
asnet(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "cox"),
  init = c("snet", "ridge"),
  gammas = 3.7,
  alphas = seq(0.05, 0.95, 0.05),
  tune = c("cv", "ebic", "bic", "aic"),
  nfolds = 5L,
  ebic.gamma = 1,
  scale = 1,
  eps = 1e-04,
  max.iter = 10000L,
  penalty.factor.init = rep(1, ncol(x)),
  seed = 1001,
  parallel = FALSE,
  verbose = FALSE
)
```

Arguments

`x`	Data matrix.
`y`	Response vector if `family` is `"gaussian"`, `"binomial"`, or `"poisson"`. If `family` is `"cox"`, a response matrix created by `Surv`.
`family`	Model family, can be `"gaussian"`, `"binomial"`, `"poisson"`, or `"cox"`.

`init`	Type of the penalty used in the initial estimation step. Can be `"snet"` or `"ridge"`.
`gammas`	Vector of candidate `gammas` (the concavity parameter) to use in SCAD-Net. Default is `3.7`.
`alphas`	Vector of candidate `alphas` to use in SCAD-Net.
`tune`	Parameter tuning method for each estimation step. Possible options are `"cv"`, `"ebic"`, `"bic"`, and `"aic"`. Default is `"cv"`.
`nfolds`	Fold numbers of cross-validation when `tune = "cv"`.
`ebic.gamma`	Parameter for Extended BIC penalizing size of the model space when `tune = "ebic"`, default is `1`. For details, see Chen and Chen (2008).
`scale`	Scaling factor for adaptive weights: `weights = coefficients^(-scale)`.
`eps`	Convergence threshhold to use in SCAD-net.
`max.iter`	Maximum number of iterations to use in SCAD-net.
`penalty.factor.init`	The multiplicative factor for the penalty applied to each coefficient in the initial estimation step. This is useful for incorporating prior information about variable weights, for example, emphasizing specific clinical variables. To make certain variables more likely to be selected, assign a smaller value. Default is `rep(1, ncol(x))`.
`seed`	Random seed for cross-validation fold division.
`parallel`	Logical. Enable parallel parameter tuning or not, default is `FALSE`. To enable parallel tuning, load the `doParallel` package and run `registerDoParallel()` with the number of CPU cores before calling this function.
`verbose`	Should we print out the estimation progress?

Value

List of model coefficients, `ncvreg` model object, and the optimal parameter set.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

asnet.fit <- asnet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2), seed = 1002
)

print(asnet.fit)
msaenet.nzv(asnet.fit)
```

```
msaenet.fp(asnet.fit, 1:5)
msaenet.tp(asnet.fit, 1:5)
asnet.pred <- predict(asnet.fit, dat$x.te)
msaenet.rmse(dat$y.te, asnet.pred)
plot(asnet.fit)
```

`coef.msaenet` *Extract Model Coefficients*

Description

Extract model coefficients from the final model in msaenet model objects.

Usage

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

Arguments

`object`	An object of class `msaenet` produced by `aenet`, `amnet`, `asnet`, `msaenet`, `msamnet`, or `msasnet`.
`...`	Additional parameters for `coef` (not used).

Value

A numerical vector of model coefficients.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

msaenet.fit <- msaenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2),
  nsteps = 3L, seed = 1003
)

coef(msaenet.fit)
```

`msaenet` *Multi-Step Adaptive Elastic-Net*

Description

Multi-Step Adaptive Elastic-Net

Usage

```
msaenet(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "cox"),
  init = c("enet", "ridge"),
  alphas = seq(0.05, 0.95, 0.05),
  tune = c("cv", "ebic", "bic", "aic"),
  nfolds = 5L,
  rule = c("lambda.min", "lambda.1se"),
  ebic.gamma = 1,
  nsteps = 2L,
  tune.nsteps = c("max", "ebic", "bic", "aic"),
  ebic.gamma.nsteps = 1,
  scale = 1,
  lower.limits = -Inf,
  upper.limits = Inf,
  penalty.factor.init = rep(1, ncol(x)),
  seed = 1001,
  parallel = FALSE,
  verbose = FALSE
)
```

Arguments

`x`	Data matrix.
`y`	Response vector if `family` is `"gaussian"`, `"binomial"`, or `"poisson"`. If `family` is `"cox"`, a response matrix created by `Surv`.
`family`	Model family, can be `"gaussian"`, `"binomial"`, `"poisson"`, or `"cox"`.
`init`	Type of the penalty used in the initial estimation step. Can be `"enet"` or `"ridge"`. See `glmnet` for details.
`alphas`	Vector of candidate `alphas` to use in `cv.glmnet`.
`tune`	Parameter tuning method for each estimation step. Possible options are `"cv"`, `"ebic"`, `"bic"`, and `"aic"`. Default is `"cv"`.
`nfolds`	Fold numbers of cross-validation when `tune = "cv"`.
`rule`	Lambda selection criterion when `tune = "cv"`, can be `"lambda.min"` or `"lambda.1se"`. See `cv.glmnet` for details.

`ebic.gamma` Parameter for Extended BIC penalizing size of the model space when `tune = "ebic"`, default is 1. For details, see Chen and Chen (2008).

`nsteps` Maximum number of adaptive estimation steps. At least 2, assuming adaptive elastic-net has only one adaptive estimation step.

`tune.nsteps` Optimal step number selection method (aggregate the optimal model from the each step and compare). Options include `"max"` (select the final-step model directly), or compare these models using `"ebic"`, `"bic"`, or `"aic"`. Default is `"max"`.

`ebic.gamma.nsteps` Parameter for Extended BIC penalizing size of the model space when `tune.nsteps = "ebic"`, default is 1.

`scale` Scaling factor for adaptive weights: `weights = coefficients^(-scale)`.

`lower.limits` Lower limits for coefficients. Default is `-Inf`. For details, see `glmnet`.

`upper.limits` Upper limits for coefficients. Default is `Inf`. For details, see `glmnet`.

`penalty.factor.init` The multiplicative factor for the penalty applied to each coefficient in the initial estimation step. This is useful for incorporating prior information about variable weights, for example, emphasizing specific clinical variables. To make certain variables more likely to be selected, assign a smaller value. Default is `rep(1, ncol(x))`.

`seed` Random seed for cross-validation fold division.

`parallel` Logical. Enable parallel parameter tuning or not, default is `FALSE`. To enable parallel tuning, load the `doParallel` package and run `registerDoParallel()` with the number of CPU cores before calling this function.

`verbose` Should we print out the estimation progress?

Value

List of model coefficients, `glmnet` model object, and the optimal parameter set.

Author(s)

Nan Xiao <https://nanx.me>

References

Nan Xiao and Qing-Song Xu. (2015). Multi-step adaptive elastic-net: reducing false positives in high-dimensional variable selection. *Journal of Statistical Computation and Simulation* 85(18), 3755–3765.

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)
```

```
msaenet.fit <- msaenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2),
  nsteps = 3L, seed = 1003
)

print(msaenet.fit)
msaenet.nzv(msaenet.fit)
msaenet.fp(msaenet.fit, 1:5)
msaenet.tp(msaenet.fit, 1:5)
msaenet.pred <- predict(msaenet.fit, dat$x.te)
msaenet.rmse(dat$y.te, msaenet.pred)
plot(msaenet.fit)
```

`msaenet.fn` *Get the Number of False Negative Selections*

Description

Get the number of false negative selections from msaenet model objects, given the indices of true variables (if known).

Usage

```
msaenet.fn(object, true.idx)
```

Arguments

`object`	An object of class `msaenet` produced by `aenet`, `amnet`, `asnet`, `msaenet`, `msamnet`, or `msasnet`.
`true.idx`	Vector. Indices of true variables.

Value

Number of false negative variables in the model.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)
```

```
msaenet.fit <- msaenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2),
  nsteps = 3L, seed = 1003
)

msaenet.fn(msaenet.fit, 1:5)
```

`msaenet.fp` *Get the Number of False Positive Selections*

Description

Get the number of false positive selections from msaenet model objects, given the indices of true variables (if known).

Usage

```
msaenet.fp(object, true.idx)
```

Arguments

`object`	An object of class `msaenet` produced by `aenet`, `amnet`, `asnet`, `msaenet`, `msamnet`, or `msasnet`.
`true.idx`	Vector. Indices of true variables.

Value

Number of false positive variables in the model.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

msaenet.fit <- msaenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2),
  nsteps = 3L, seed = 1003
)

msaenet.fp(msaenet.fit, 1:5)
```

msaenet.mae *Mean Absolute Error (MAE)*

Description

Compute mean absolute error (MAE).

Usage

```
msaenet.mae(yreal, ypred)
```

Arguments

yreal	Vector. True response.
ypred	Vector. Predicted response.

Value

MAE

Author(s)

Nan Xiao <https://nanx.me>

msaenet.mse *Mean Squared Error (MSE)*

Description

Compute mean squared error (MSE).

Usage

```
msaenet.mse(yreal, ypred)
```

Arguments

yreal	Vector. True response.
ypred	Vector. Predicted response.

Value

MSE

Author(s)

Nan Xiao <https://nanx.me>

`msaenet.nzv` *Get Indices of Non-Zero Variables*

Description

Get the indices of non-zero variables from msaenet model objects.

Usage

```
msaenet.nzv(object)
```

Arguments

`object`	An object of class `msaenet` produced by `aenet`, `amnet`, `asnet`, `msaenet`, `msamnet`, or `msasnet`.

Value

Indices vector of non-zero variables in the model.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

msaenet.fit <- msaenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2),
  nsteps = 3L, seed = 1003
)

msaenet.nzv(msaenet.fit)

# coefficients of non-zero variables
coef(msaenet.fit)[msaenet.nzv(msaenet.fit)]
```

`msaenet.nzv.all` *Get Indices of Non-Zero Variables in All Steps*

Description

Get the indices of non-zero variables in all steps from msaenet model objects.

Usage

```
msaenet.nzv.all(object)
```

Arguments

`object`	An object of class `msaenet` produced by `aenet`, `amnet`, `asnet`, `msaenet`, `msamnet`, or `msasnet`.

Value

List containing indices vectors of non-zero variables in all steps.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

msaenet.fit <- msaenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2),
  nsteps = 3L, seed = 1003
)

msaenet.nzv.all(msaenet.fit)
```

msaenet.rmse *Root Mean Squared Error (RMSE)*

Description

Compute root mean squared error (RMSE).

Usage

```
msaenet.rmse(yreal, ypred)
```

Arguments

yreal	Vector. True response.
ypred	Vector. Predicted response.

Value

RMSE

Author(s)

Nan Xiao <https://nanx.me>

msaenet.rmsle *Root Mean Squared Logarithmic Error (RMSLE)*

Description

Compute root mean squared logarithmic error (RMSLE).

Usage

```
msaenet.rmsle(yreal, ypred)
```

Arguments

yreal	Vector. True response.
ypred	Vector. Predicted response.

Value

RMSLE

Author(s)

Nan Xiao <https://nanx.me>

`msaenet.sim.binomial` *Generate Simulation Data for Benchmarking Sparse Regressions (Binomial Response)*

Description

Generate simulation data for benchmarking sparse logistic regression models.

Usage

```
msaenet.sim.binomial(
  n = 300,
  p = 500,
  rho = 0.5,
  coef = rep(0.2, 50),
  snr = 1,
  p.train = 0.7,
  seed = 1001
)
```

Arguments

`n`	Number of observations.
`p`	Number of variables.
`rho`	Correlation base for generating correlated variables.
`coef`	Vector of non-zero coefficients.
`snr`	Signal-to-noise ratio (SNR).
`p.train`	Percentage of training set.
`seed`	Random seed for reproducibility.

Value

List of `x.tr`, `x.te`, `y.tr`, and `y.te`.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.binomial(
  n = 300, p = 500, rho = 0.6,
  coef = rep(1, 10), snr = 3, p.train = 0.7,
  seed = 1001
)
```

```
dim(dat$x.tr)
dim(dat$x.te)
table(dat$y.tr)
table(dat$y.te)
```

`msaenet.sim.cox` *Generate Simulation Data for Benchmarking Sparse Regressions (Cox Model)*

Description

Generate simulation data for benchmarking sparse Cox regression models.

Usage

```
msaenet.sim.cox(
  n = 300,
  p = 500,
  rho = 0.5,
  coef = rep(0.2, 50),
  snr = 1,
  p.train = 0.7,
  seed = 1001
)
```

Arguments

`n`	Number of observations.
`p`	Number of variables.
`rho`	Correlation base for generating correlated variables.
`coef`	Vector of non-zero coefficients.
`snr`	Signal-to-noise ratio (SNR).
`p.train`	Percentage of training set.
`seed`	Random seed for reproducibility.

Value

List of `x.tr`, `x.te`, `y.tr`, and `y.te`.

Author(s)

Nan Xiao <https://nanx.me>

References

Simon, N., Friedman, J., Hastie, T., & Tibshirani, R. (2011). Regularization Paths for Cox's Proportional Hazards Model via Coordinate Descent. *Journal of Statistical Software*, 39(5), 1–13.

Examples

```
dat <- msaenet.sim.cox(
  n = 300, p = 500, rho = 0.6,
  coef = rep(1, 10), snr = 3, p.train = 0.7,
  seed = 1001
)

dim(dat$x.tr)
dim(dat$x.te)
dim(dat$y.tr)
dim(dat$y.te)
```

---

msaenet.sim.gaussian *Generate Simulation Data for Benchmarking Sparse Regressions (Gaussian Response)*

Description

Generate simulation data (Gaussian case) following the settings in Xiao and Xu (2015).

Usage

```
msaenet.sim.gaussian(
  n = 300,
  p = 500,
  rho = 0.5,
  coef = rep(0.2, 50),
  snr = 1,
  p.train = 0.7,
  seed = 1001
)
```

Arguments

n	Number of observations.
p	Number of variables.
rho	Correlation base for generating correlated variables.
coef	Vector of non-zero coefficients.
snr	Signal-to-noise ratio (SNR). SNR is defined as $$\frac{Var(E(y\|X))}{Var(Y-E(y\|X))}=\frac{Var(f(X))}{Var(\varepsilon)}=\frac{Var(X^T\beta)}{Var(\varepsilon)}=\frac{Var(\beta^T\Sigma\beta)}{\sigma^2}.$$
p.train	Percentage of training set.
seed	Random seed for reproducibility.

Value

List of `x.tr`, `x.te`, `y.tr`, and `y.te`.

Author(s)

Nan Xiao <https://nanx.me>

References

Nan Xiao and Qing-Song Xu. (2015). Multi-step adaptive elastic-net: reducing false positives in high-dimensional variable selection. *Journal of Statistical Computation and Simulation* 85(18), 3755–3765.

Examples

```
dat <- msaenet.sim.gaussian(
  n = 300, p = 500, rho = 0.6,
  coef = rep(1, 10), snr = 3, p.train = 0.7,
  seed = 1001
)

dim(dat$x.tr)
dim(dat$x.te)
```

---

msaenet.sim.poisson *Generate Simulation Data for Benchmarking Sparse Regressions (Poisson Response)*

Description

Generate simulation data for benchmarking sparse Poisson regression models.

Usage

```
msaenet.sim.poisson(
  n = 300,
  p = 500,
  rho = 0.5,
  coef = rep(0.2, 50),
  snr = 1,
  p.train = 0.7,
  seed = 1001
)
```

Arguments

n	Number of observations.
p	Number of variables.
rho	Correlation base for generating correlated variables.
coef	Vector of non-zero coefficients.
snr	Signal-to-noise ratio (SNR).
p.train	Percentage of training set.
seed	Random seed for reproducibility.

Value

List of `x.tr`, `x.te`, `y.tr`, and `y.te`.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.poisson(
  n = 300, p = 500, rho = 0.6,
  coef = rep(1, 10), snr = 3, p.train = 0.7,
  seed = 1001
)

dim(dat$x.tr)
dim(dat$x.te)
```

---

msaenet.tp *Get the Number of True Positive Selections*

Description

Get the number of true positive selections from msaenet model objects, given the indices of true variables (if known).

Usage

```
msaenet.tp(object, true.idx)
```

Arguments

object	An object of class `msaenet` produced by `aenet`, `amnet`, `asnet`, `msaenet`, `msamnet`, or `msasnet`.
true.idx	Vector. Indices of true variables.

Value

Number of true positive variables in the model.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

msaenet.fit <- msaenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2),
  nsteps = 3L, seed = 1003
)

msaenet.tp(msaenet.fit, 1:5)
```

---

msamnet *Multi-Step Adaptive MCP-Net*

Description

Multi-Step Adaptive MCP-Net

Usage

```
msamnet(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "cox"),
  init = c("mnet", "ridge"),
  gammas = 3,
  alphas = seq(0.05, 0.95, 0.05),
  tune = c("cv", "ebic", "bic", "aic"),
  nfolds = 5L,
  ebic.gamma = 1,
  nsteps = 2L,
  tune.nsteps = c("max", "ebic", "bic", "aic"),
  ebic.gamma.nsteps = 1,
  scale = 1,
  eps = 1e-04,
  max.iter = 10000L,
```

```
  penalty.factor.init = rep(1, ncol(x)),
  seed = 1001,
  parallel = FALSE,
  verbose = FALSE
)
```

Arguments

x	Data matrix.
y	Response vector if family is "gaussian", "binomial", or "poisson". If family is "cox", a response matrix created by Surv.
family	Model family, can be "gaussian", "binomial", "poisson", or "cox".
init	Type of the penalty used in the initial estimation step. Can be "mnet" or "ridge".
gammas	Vector of candidate gammas (the concavity parameter) to use in MCP-Net. Default is 3.
alphas	Vector of candidate alphas to use in MCP-Net.
tune	Parameter tuning method for each estimation step. Possible options are "cv", "ebic", "bic", and "aic". Default is "cv".
nfolds	Fold numbers of cross-validation when tune = "cv".
ebic.gamma	Parameter for Extended BIC penalizing size of the model space when tune = "ebic", default is 1. For details, see Chen and Chen (2008).
nsteps	Maximum number of adaptive estimation steps. At least 2, assuming adaptive MCP-net has only one adaptive estimation step.
tune.nsteps	Optimal step number selection method (aggregate the optimal model from the each step and compare). Options include "max" (select the final-step model directly), or compare these models using "ebic", "bic", or "aic". Default is "max".
ebic.gamma.nsteps	Parameter for Extended BIC penalizing size of the model space when tune.nsteps = "ebic", default is 1.
scale	Scaling factor for adaptive weights: weights = coefficients^(-scale).
eps	Convergence threshhold to use in MCP-net.
max.iter	Maximum number of iterations to use in MCP-net.
penalty.factor.init	The multiplicative factor for the penalty applied to each coefficient in the initial estimation step. This is useful for incorporating prior information about variable weights, for example, emphasizing specific clinical variables. To make certain variables more likely to be selected, assign a smaller value. Default is rep(1, ncol(x)).
seed	Random seed for cross-validation fold division.
parallel	Logical. Enable parallel parameter tuning or not, default is FALSE. To enable parallel tuning, load the doParallel package and run registerDoParallel() with the number of CPU cores before calling this function.
verbose	Should we print out the estimation progress?

Value

List of model coefficients, ncvreg model object, and the optimal parameter set.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

msamnet.fit <- msamnet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.3, 0.9, 0.3),
  nsteps = 3L, seed = 1003
)

print(msamnet.fit)
msaenet.nzv(msamnet.fit)
msaenet.fp(msamnet.fit, 1:5)
msaenet.tp(msamnet.fit, 1:5)
msamnet.pred <- predict(msamnet.fit, dat$x.te)
msaenet.rmse(dat$y.te, msamnet.pred)
plot(msamnet.fit)
```

---

msasnet *Multi-Step Adaptive SCAD-Net*

Description

Multi-Step Adaptive SCAD-Net

Usage

```
msasnet(
  x,
  y,
  family = c("gaussian", "binomial", "poisson", "cox"),
  init = c("snet", "ridge"),
  gammas = 3.7,
  alphas = seq(0.05, 0.95, 0.05),
  tune = c("cv", "ebic", "bic", "aic"),
  nfolds = 5L,
  ebic.gamma = 1,
```

```
  nsteps = 2L,
  tune.nsteps = c("max", "ebic", "bic", "aic"),
  ebic.gamma.nsteps = 1,
  scale = 1,
  eps = 1e-04,
  max.iter = 10000L,
  penalty.factor.init = rep(1, ncol(x)),
  seed = 1001,
  parallel = FALSE,
  verbose = FALSE
)
```

Arguments

x	Data matrix.
y	Response vector if family is "gaussian", "binomial", or "poisson". If family is "cox", a response matrix created by Surv.
family	Model family, can be "gaussian", "binomial", "poisson", or "cox".
init	Type of the penalty used in the initial estimation step. Can be "snet" or "ridge".
gammas	Vector of candidate gammas (the concavity parameter) to use in SCAD-Net. Default is 3.7.
alphas	Vector of candidate alphas to use in SCAD-Net.
tune	Parameter tuning method for each estimation step. Possible options are "cv", "ebic", "bic", and "aic". Default is "cv".
nfolds	Fold numbers of cross-validation when tune = "cv".
ebic.gamma	Parameter for Extended BIC penalizing size of the model space when tune = "ebic", default is 1. For details, see Chen and Chen (2008).
nsteps	Maximum number of adaptive estimation steps. At least 2, assuming adaptive SCAD-net has only one adaptive estimation step.
tune.nsteps	Optimal step number selection method (aggregate the optimal model from the each step and compare). Options include "max" (select the final-step model directly), or compare these models using "ebic", "bic", or "aic". Default is "max".
ebic.gamma.nsteps	Parameter for Extended BIC penalizing size of the model space when tune.nsteps = "ebic", default is 1.
scale	Scaling factor for adaptive weights: weights = coefficients^(-scale).
eps	Convergence threshhold to use in SCAD-net.
max.iter	Maximum number of iterations to use in SCAD-net.
penalty.factor.init	The multiplicative factor for the penalty applied to each coefficient in the initial estimation step. This is useful for incorporating prior information about variable weights, for example, emphasizing specific clinical variables. To make certain variables more likely to be selected, assign a smaller value. Default is rep(1, ncol(x)).

seed	Random seed for cross-validation fold division.
parallel	Logical. Enable parallel parameter tuning or not, default is FALSE. To enable parallel tuning, load the doParallel package and run registerDoParallel() with the number of CPU cores before calling this function.
verbose	Should we print out the estimation progress?

Value

List of model coefficients, ncvreg model object, and the optimal parameter set.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

msasnet.fit <- msasnet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.3, 0.9, 0.3),
  nsteps = 3L, seed = 1003
)

print(msasnet.fit)
msaenet.nzv(msasnet.fit)
msaenet.fp(msasnet.fit, 1:5)
msaenet.tp(msasnet.fit, 1:5)
msasnet.pred <- predict(msasnet.fit, dat$x.te)
msaenet.rmse(dat$y.te, msasnet.pred)
plot(msasnet.fit)
```

---

plot.msaenet *Plot msaenet Model Objects*

Description

Plot msaenet model objects.

Usage

```
## S3 method for class 'msaenet'
plot(
  x,
  type = c("coef", "criterion", "dotplot"),
  nsteps = NULL,
  highlight = TRUE,
  col = NULL,
  label = FALSE,
  label.vars = NULL,
  label.pos = 2,
  label.offset = 0.3,
  label.cex = 0.7,
  label.srt = 90,
  xlab = NULL,
  ylab = NULL,
  abs = FALSE,
  ...
)
```

Arguments

x	An object of class msaenet produced by aenet, amnet, asnet, msaenet, msamnet, or msasnet.
type	Plot type, "coef" for a coefficient path plot across all estimation steps; "criterion" for a scree plot of the model evaluation criterion used (CV error, AIC, BIC, or EBIC); "dotplot" for a Cleveland dot plot of the coefficients estimated by the model at the optimal step.
nsteps	Maximum number of estimation steps to plot. Default is to plot all steps.
highlight	Should we highlight the "optimal" step according to the criterion? Default is TRUE.
col	Color palette to use for the coefficient paths. If it is NULL, a default color palette will be assigned.
label	Should we label all the non-zero variables of the optimal step in the coefficient plot or the dot plot? Default is FALSE. If TRUE and label.vars = NULL, the index of the non-zero variables will be used as labels.
label.vars	Labels to use for all the variables if label = "TRUE".
label.pos	Position of the labels. See argument pos in text for details.
label.offset	Offset of the labels. See argument offset in text for details.
label.cex	Character expansion factor of the labels. See argument cex in text for details.
label.srt	Label rotation in degrees for the Cleveland dot plot. Default is 90. See argument srt in par for details.
xlab	Title for x axis. If is NULL, will use the default title.
ylab	Title for y axis. If is NULL, will use the default title.

abs	Should we plot the absolute values of the coefficients instead of the raw coefficients in the Cleveland dot plot? Default is FALSE.
...	Other parameters (not used).

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

fit <- msaenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2),
  nsteps = 5L, tune.nsteps = "bic",
  seed = 1002
)

plot(fit)
plot(fit, label = TRUE)
plot(fit, label = TRUE, nsteps = 5)
plot(fit, type = "criterion")
plot(fit, type = "criterion", nsteps = 5)
plot(fit, type = "dotplot", label = TRUE)
plot(fit, type = "dotplot", label = TRUE, abs = TRUE)
```

---

predict.msaenet *Make Predictions from an msaenet Model*

Description

Make predictions on new data by a msaenet model object.

Usage

```
## S3 method for class 'msaenet'
predict(object, newx, ...)
```

Arguments

object	An object of class msaenet produced by aenet, amnet, asnet, msaenet, msamnet, or msasnet.
newx	New data to predict with.
...	Additional parameters, particularly prediction type in predict.glmnet, predict.ncvreg, or predict.ncvsurv.

Value

Numeric matrix of the predicted values.

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

msaenet.fit <- msaenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2),
  nsteps = 3L, seed = 1003
)

msaenet.pred <- predict(msaenet.fit, dat$x.te)
msaenet.rmse(dat$y.te, msaenet.pred)
```

---

print.msaenet *Print msaenet Model Information*

Description

Print msaenet model objects (currently, only printing the model information of the final step).

Usage

```
## S3 method for class 'msaenet'
print(x, ...)
```

Arguments

x	An object of class msaenet.
...	Additional parameters for print (not used).

Author(s)

Nan Xiao <https://nanx.me>

Examples

```
dat <- msaenet.sim.gaussian(
  n = 150, p = 500, rho = 0.6,
  coef = rep(1, 5), snr = 2, p.train = 0.7,
  seed = 1001
)

msaenet.fit <- msaenet(
  dat$x.tr, dat$y.tr,
  alphas = seq(0.2, 0.8, 0.2),
  nsteps = 3L, seed = 1003
)

print(msaenet.fit)
```

Index