Package ‘miselect’

July 22, 2025

Title Variable Selection for Multiply Imputed Data

Version 0.9.2

Description Penalized regression methods, such as lasso and elastic net, are used in many biomedical applications when simultaneous regression coefficient estimation and variable selection is desired. However, missing data complicates the implementation of these methods, particularly when missingness is handled using multiple imputation. Applying a variable selection algorithm on each imputed dataset will likely lead to different sets of selected predictors, making it difficult to ascertain a final active set without resorting to ad hoc combination rules. 'miselect' presents Stacked Adaptive Elastic Net (saenet) and Grouped Adaptive LASSO (galasso) for continuous and binary outcomes, developed by Du et al (2022) <doi:10.1080/10618600.2022.2035739>. They, by construction, force selection of the same variables across multiply imputed data. 'miselect' also provides cross validated variants of these methods.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 3.5.0)

Suggests mice, knitr, rmarkdown, testthat

VignetteBuilder knitr

NeedsCompilation no

Author Michael Kleinsasser [cre],
Alexander Rix [aut],
Jiacong Du [aut]

Maintainer Michael Kleinsasser <biostat-cran-manager@umich.edu>

Repository CRAN

Date/Publication 2024-03-05 17:00:08 UTC

Contents

`coef.cv.galasso` *Extract Coefficients From a "cv.galasso" Object*

Description

Extract Coefficients From a "cv.galasso" Object

Usage

```
## S3 method for class 'cv.galasso'
coef(object, lambda = object$lambda.min, ...)
```

Arguments

`object`	A "cv.galasso" fit
`lambda`	Chosen value of lambda. Must be between "min(lambda)" and "max(lambda)". Default is "lambda.min"
`...`	Additional unused arguments

Value

A list of numeric vectors containing the coefficients from running `galasso` on `lambda` for each imputation.

coef.cv.saenet *Extract Coefficients From a "cv.saenet" Object*

Description

Extract Coefficients From a "cv.saenet" Object

Usage

```
## S3 method for class 'cv.saenet'
coef(object, lambda = object$lambda.min, alpha = object$alpha.min, ...)
```

Arguments

`object`	A "cv.saenet" fit
`lambda`	Chosen value of lambda. Must be between "min(lambda)" and "max(lambda)". Default is "lambda.min"
`alpha`	Chosen value of alpha. Must be between "min(alpha)" and "max(alpha)". Default is "alpha.min"
`...`	Additional unused arguments

Value

A numeric vector containing the coefficients from running `saenet` on `lambda` and `alpha`.

coef.galasso *Extract Coefficients From a "galasso" Object*

Description

Extract Coefficients From a "galasso" Object

Usage

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

Arguments

`object`	A "galasso" fit
`lambda`	Chosen value of lambda. Must be between "min(lambda)" and "max(lambda)". Default is "lambda.min"
`...`	Additional unused arguments

Value

A list of length D containing the coefficient estimates from running `galasso` on `lambda`.

coef.saenet *Extract Coefficients From a "saenet" Object*

Description

`coef.galasso` averages the estimates across imputations to return a single vector instead of a matrix.

Usage

```
## S3 method for class 'saenet'
coef(object, lambda, alpha, ...)
```

Arguments

`object`	A "cv.saenet" fit
`lambda`	Chosen value of lambda. Must be between "min(lambda)" and "max(lambda)". Default is "lambda.min"
`alpha`	Chosen value of alpha. Must be between "min(alpha)" and "max(alpha)". Default is "alpha.min"
`...`	Additional unused arguments

Value

A numeric vector containing the coefficients from running `saenet` on `lambda` and `alpha`.

cv.galasso *Cross Validated Multiple Imputation Grouped Adaptive LASSO*

Description

Does k-fold cross-validation for `galasso`, and returns an optimal value for lambda.

Usage

```
cv.galasso(
  x,
  y,
  pf,
  adWeight,
  family = c("gaussian", "binomial"),
  nlambda = 100,
 lambda.min.ratio = ifelse(isTRUE(all.equal(adWeight, rep(1, p))), 0.001, 1e-06),
  lambda = NULL,
  nfolds = 5,
```

```
  foldid = NULL,
  maxit = 1000,
  eps = 1e-05
)
```

Arguments

- `x` A length `m` list of `n * p` numeric matrices. No matrix should contain an intercept, or any missing values
- `y` A length `m` list of length `n` numeric response vectors. No vector should contain missing values
- `pf` Penalty factor. Can be used to differentially penalize certain variables
- `adWeight` Numeric vector of length p representing the adaptive weights for the L1 penalty
- `family` The type of response. "gaussian" implies a continuous response and "binomial" implies a binary response. Default is "gaussian".
- `nlambda` Length of automatically generated "lambda" sequence. If "lambda" is non NULL, "nlambda" is ignored. Default is 100
- `lambda.min.ratio` Ratio that determines the minimum value of "lambda" when automatically generating a "lambda" sequence. If "lambda" is not NULL, "lambda.min.ratio" is ignored. Default is 1e-4
- `lambda` Optional numeric vector of lambdas to fit. If NULL, `galasso` will automatically generate a lambda sequence based off of `nlambda` and `lambda.min.ratio`. Default is NULL
- `nfolds` Number of foldid to use for cross validation. Default is 5, minimum is 3
- `foldid` an optional length `n` vector of values between 1 and `cv.galasso` will automatically generate folds
- `maxit` Maximum number of iterations to run. Default is 10000
- `eps` Tolerance for convergence. Default is 1e-5

Details

`cv.galasso` works by adding a group penalty to the aggregated objective function to ensure selection consistency across imputations. Simulations suggest that the "stacked" objective function approaches (i.e., `saenet`) tend to be more computationally efficient and have better estimation and selection properties.

Value

An object of type "cv.galasso" with 7 elements:

call The call that generated the output.

lambda The sequence of lambdas fit.

cvm Average cross validation error for each "lambda". For family = "gaussian", "cvm" corresponds to mean squared error, and for binomial "cvm" corresponds to deviance.

cvse Standard error of "cvm".

galasso.fit A "galasso" object fit to the full data.

lambda.min The lambda value for the model with the minimum cross validation error.

lambda.1se The lambda value for the sparsest model within one standard error of the minimum cross validation error.

df The number of nonzero coefficients for each value of lambda.

References

Du, J., Boss, J., Han, P., Beesley, L. J., Kleinsasser, M., Goutman, S. A., ... & Mukherjee, B. (2022). Variable selection with multiply-imputed datasets: choosing between stacked and grouped methods. Journal of Computational and Graphical Statistics, 31(4), 1063-1075. <doi:10.1080/10618600.2022.2035739>

Examples

```
library(miselect)
library(mice)

set.seed(48109)

# Using the mice defaults for sake of example only.
mids <- mice(miselect.df, m = 5, printFlag = FALSE)
dfs <- lapply(1:5, function(i) complete(mids, action = i))

# Generate list of imputed design matrices and imputed responses
x <- list()
y <- list()
for (i in 1:5) {
    x[[i]] <- as.matrix(dfs[[i]][, paste0("X", 1:20)])
    y[[i]] <- dfs[[i]]$Y
}

pf       <- rep(1, 20)
adWeight <- rep(1, 20)

fit <- cv.galasso(x, y, pf, adWeight)

# By default 'coef' returns the betas for lambda.min.
coef(fit)
```

`cv.saenet` *Cross Validated Multiple Imputation Stacked Adaptive Elastic Net*

Description

Does k-fold cross-validation for `saenet`, and returns optimal values for lambda and alpha.

Usage

```
cv.saenet(
  x,
  y,
  pf,
  adWeight,
  weights,
  family = c("gaussian", "binomial"),
  alpha = 1,
  nlambda = 100,
  lambda.min.ratio = ifelse(isTRUE(all.equal(adWeight, rep(1, p))), 0.001, 1e-06),
  lambda = NULL,
  nfolds = 5,
  foldid = NULL,
  maxit = 1000,
  eps = 1e-05
)
```

Arguments

`x`	A length `m` list of `n * p` numeric matrices. No matrix should contain an intercept, or any missing values
`y`	A length `m` list of length `n` numeric response vectors. No vector should contain missing values
`pf`	Penalty factor of length `p`. Can be used to differentially penalize certain variables. 0 indicates to not penalize the covariate
`adWeight`	Numeric vector of length p representing the adaptive weights for the L1 penalty
`weights`	Numeric vector of length `n` containing the proportion observed (non-missing) for each row in the un-imputed data.
`family`	The type of response. "gaussian" implies a continuous response and "binomial" implies a binary response. Default is "gaussian".
`alpha`	Elastic net parameter. Can be a vector to cross validate over. Default is 1
`nlambda`	Length of automatically generated "lambda" sequence. If "lambda" is non NULL, "nlambda" is ignored. Default is 100
`lambda.min.ratio`	Ratio that determines the minimum value of "lambda" when automatically generating a "lambda" sequence. If "lambda" is not NULL, "lambda.min.ratio" is ignored. Default is 1e-3
`lambda`	Optional numeric vector of lambdas to fit. If NULL, `galasso` will automatically generate a lambda sequence based off of `nlambda` and `lambda.min.ratio`. Default is NULL
`nfolds`	Number of foldid to use for cross validation. Default is 5, minimum is 3
`foldid`	an optional length `n` vector of values between 1 and `cv.galasso` will automatically generate folds
`maxit`	Maximum number of iterations to run. Default is 1000
`eps`	Tolerance for convergence. Default is 1e-5

Details

`cv.saenet` works by stacking the multiply imputed data into a single matrix and running a weighted adaptive elastic net on it. Simulations suggest that the "stacked" objective function approaches tend to be more computationally efficient and have better estimation and selection properties.

Due to stacking, the automatically generated `lambda` sequence `cv.saenet` generates may end up underestimating `lambda.max`, and thus the degrees of freedom may be nonzero at the first lambda value.

Value

An object of type "cv.saenet" with 9 elements:

call The call that generated the output.

lambda Sequence of lambdas fit.

cvm Average cross validation error for each lambda and alpha. For family = "gaussian", "cvm" corresponds to mean squared error, and for binomial "cvm" corresponds to deviance.

cvse Standard error of "cvm".

saenet.fit A "saenet" object fit to the full data.

lambda.min The lambda value for the model with the minimum cross validation error.

lambda.1se The lambda value for the sparsest model within one standard error of the minimum cross validation error.

alpha.min The alpha value for the model with the minimum cross validation error.

alpha.1se The alpha value for the sparsest model within one standard error of the minimum cross validation error.

df The number of nonzero coefficients for each value of lambda and alpha.

References

Du, J., Boss, J., Han, P., Beesley, L. J., Kleinsasser, M., Goutman, S. A., ... & Mukherjee, B. (2022). Variable selection with multiply-imputed datasets: choosing between stacked and grouped methods. Journal of Computational and Graphical Statistics, 31(4), 1063-1075. <doi:10.1080/10618600.2022.2035739>

Examples

```
library(miselect)
library(mice)

set.seed(48109)

# Using the mice defaults for sake of example only.
mids <- mice(miselect.df, m = 5, printFlag = FALSE)
dfs <- lapply(1:5, function(i) complete(mids, action = i))

# Generate list of imputed design matrices and imputed responses
x <- list()
y <- list()
for (i in 1:5) {
```

```
    x[[i]] <- as.matrix(dfs[[i]][, paste0("X", 1:20)])
    y[[i]] <- dfs[[i]]$Y
}

# Calculate observational weights
weights  <- 1 - rowMeans(is.na(miselect.df))
pf       <- rep(1, 20)
adWeight <- rep(1, 20)

# Since 'Y' is a binary variable, we use 'family = "binomial"'
fit <- cv.saenet(x, y, pf, adWeight, weights, family = "binomial")

# By default 'coef' returns the betas for (lambda.min , alpha.min)
coef(fit)


# You can also cross validate over alpha

fit <- cv.saenet(x, y, pf, adWeight, weights, family = "binomial",
                 alpha = c(.5, 1))
# Get selected variables from the 1 standard error rule
coef(fit, lambda = fit$lambda.1se, alpha = fit$alpha.1se)
```

---

galasso *Multiple Imputation Grouped Adaptive LASSO*

Description

galasso fits an adaptive LASSO for multiply imputed data. "galasso" supports both continuous and binary responses.

Usage

```
galasso(
  x,
  y,
  pf,
  adWeight,
  family = c("gaussian", "binomial"),
  nlambda = 100,
  lambda.min.ratio = ifelse(isTRUE(all.equal(adWeight, rep(1, p))), 0.001, 1e-06),
  lambda = NULL,
  maxit = 10000,
  eps = 1e-05
)
```

Arguments

`x`	A length `m` list of `n * p` numeric matrices. No matrix should contain an intercept, or any missing values
`y`	A length `m` list of length `n` numeric response vectors. No vector should contain missing values
`pf`	Penalty factor. Can be used to differentially penalize certain variables
`adWeight`	Numeric vector of length p representing the adaptive weights for the L1 penalty
`family`	The type of response. "gaussian" implies a continuous response and "binomial" implies a binary response. Default is "gaussian".
`nlambda`	Length of automatically generated "lambda" sequence. If "lambda" is non NULL, "nlambda" is ignored. Default is 100
`lambda.min.ratio`	Ratio that determines the minimum value of "lambda" when automatically generating a "lambda" sequence. If "lambda" is not NULL, "lambda.min.ratio" is ignored. Default is 1e-4
`lambda`	Optional numeric vector of lambdas to fit. If NULL, `galasso` will automatically generate a lambda sequence based off of `nlambda` and `lambda.min.ratio`. Default is NULL
`maxit`	Maximum number of iterations to run. Default is 10000
`eps`	Tolerance for convergence. Default is 1e-5

Details

`galasso` works by adding a group penalty to the aggregated objective function to ensure selection consistency across imputations. The objective function is:

$$argmin_{\beta_{jk}} - L(\beta_{jk}|X_{ijk}, Y_{ik})$$

$$+\lambda * \Sigma_{j=1}^{p} \hat{a}_j * pf_j * \sqrt{\Sigma_{k=1}^{m} \beta_{jk}^2}$$

Where L is the log likelihood,`a` is the adaptive weights, and `pf` is the penalty factor. Simulations suggest that the "stacked" objective function approach (i.e., `saenet`) tends to be more computationally efficient and have better estimation and selection properties. However, the advantage of `galasso` is that it allows one to look at the differences between coefficient estimates across imputations.

Value

An object with type galasso and subtype galasso.gaussian or galasso.binomial, depending on which family was used. Both subtypes have 4 elements:

lambda Sequence of lambda fit.

coef a list of length D containing the coefficient estimates from running galasso at each value of lambda. Each element in the list is a nlambda x (p+1) matrix.

df Number of nonzero betas at each value of lambda.

References

Du, J., Boss, J., Han, P., Beesley, L. J., Kleinsasser, M., Goutman, S. A., ... & Mukherjee, B. (2022). Variable selection with multiply-imputed datasets: choosing between stacked and grouped methods. Journal of Computational and Graphical Statistics, 31(4), 1063-1075. <doi:10.1080/10618600.2022.2035739>

Examples

```
library(miselect)
library(mice)

mids <- mice(miselect.df, m = 5, printFlag = FALSE)
dfs <- lapply(1:5, function(i) complete(mids, action = i))

# Generate list of imputed design matrices and imputed responses
x <- list()
y <- list()
for (i in 1:5) {
    x[[i]] <- as.matrix(dfs[[i]][, paste0("X", 1:20)])
    y[[i]] <- dfs[[i]]$Y
}

pf       <- rep(1, 20)
adWeight <- rep(1, 20)

fit <- galasso(x, y, pf, adWeight)
```

`miselect.df` *Synthetic Example Data For "miselect"*

Description

This synthetic data is taken from the first simulation case from the miselect paper

Usage

```
miselect.df
```

Format

A data.frame with 500 observations on 21 variables:

Y Binary response.

X1-X20 Covariates with missing data.

print.cv.galasso *Print cv.galasso Objects*

Description

`print.cv.galasso` print the fit and returns it invisibly.

Usage

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

Arguments

`x`	An object of type "cv.galasso" to print
`...`	Further arguments passed to or from other methods

print.cv.saenet *Print cv.saenet Objects*

Description

`print.cv.saenet` print the fit and returns it invisibly.

Usage

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

Arguments

`x`	An object of type "cv.saenet" to print
`...`	Further arguments passed to or from other methods

saenet *Multiple Imputation Stacked Adaptive Elastic Net*

Description

Fits an adaptive elastic net for multiply imputed data. The data is stacked and is penalized that each imputation selects the same betas at each value of lambda. "saenet" supports both continuous and binary responses.

Usage

```
saenet(
  x,
  y,
  pf,
  adWeight,
  weights,
  family = c("gaussian", "binomial"),
  alpha = 1,
  nlambda = 100,
  lambda.min.ratio = ifelse(isTRUE(all.equal(adWeight, rep(1, p))), 0.001, 1e-06),
  lambda = NULL,
  maxit = 1000,
  eps = 1e-05
)
```

Arguments

- `x` A length `m` list of `n * p` numeric matrices. No matrix should contain an intercept, or any missing values
- `y` A length `m` list of length `n` numeric response vectors. No vector should contain missing values
- `pf` Penalty factor. Can be used to differentially penalize certain variables
- `adWeight` Numeric vector of length p representing the adaptive weights for the L1 penalty
- `weights` Numeric vector of length n containing the proportion observed (non-missing) for each row in the un-imputed data.
- `family` The type of response. "gaussian" implies a continuous response and "binomial" implies a binary response. Default is "gaussian".
- `alpha` Elastic net parameter. Can be a vector to cross validate over. Default is 1
- `nlambda` Length of automatically generated "lambda" sequence. If "lambda" is non NULL, "nlambda" is ignored. Default is 100
- `lambda.min.ratio` Ratio that determines the minimum value of "lambda" when automatically generating a "lambda" sequence. If "lambda" is not NULL, "lambda.min.ratio" is ignored. Default is 1e-3

`lambda` Optional numeric vector of lambdas to fit. If NULL, `galasso` will automatically generate a lambda sequence based off of `nlambda` and `lambda.min.ratio`. Default is NULL

`maxit` Maximum number of iterations to run. Default is 1000

`eps` Tolerance for convergence. Default is 1e-5

Details

`saenet` works by stacking the multiply imputed data into a single matrix and running a weighted adaptive elastic net on it. The objective function is:

$$argmin_{\beta_j} - \frac{1}{n}\sum_{k=1}^{m}\sum_{i=1}^{n} o_i * L(\beta_j|Y_{ik}, X_{ijk})$$

$$+\lambda(\alpha\sum_{j=1}^{p}\hat{a}_j * pf_j|\beta_j|$$

$$+(1-\alpha)\sum_{j=1}^{p} pf_j * \beta_j^2)$$

Where L is the log likelihood, `o = w / m`, `a` is the adaptive weights, and `pf` is the penalty factor. Simulations suggest that the "stacked" objective function approach (i.e., `saenet`) tends to be more computationally efficient and have better estimation and selection properties. However, the advantage of `galasso` is that it allows one to look at the differences between coefficient estimates across imputations.

Value

An object with type saenet and subtype saenet.gaussian or saenet.binomial, depending on which family was used. Both subtypes have 4 elements:

lambda Sequence of lambda fit.

coef nlambda x nalpha x p + 1 tensor representing the estimated betas at each value of lambda and alpha.

df Number of nonzero betas at each value of lambda and alpha.

References

Du, J., Boss, J., Han, P., Beesley, L. J., Kleinsasser, M., Goutman, S. A., ... & Mukherjee, B. (2022). Variable selection with multiply-imputed datasets: choosing between stacked and grouped methods. Journal of Computational and Graphical Statistics, 31(4), 1063-1075. <doi:10.1080/10618600.2022.2035739>

Examples

```
library(miselect)
library(mice)

mids <- mice(miselect.df, m = 5, printFlag = FALSE)
dfs <- lapply(1:5, function(i) complete(mids, action = i))
```

```
# Generate list of imputed design matrices and imputed responses
x <- list()
y <- list()
for (i in 1:5) {
    x[[i]] <- as.matrix(dfs[[i]][, paste0("X", 1:20)])
    y[[i]] <- dfs[[i]]$Y
}

# Calculate observational weights
weights  <- 1 - rowMeans(is.na(miselect.df))
pf       <- rep(1, 20)
adWeight <- rep(1, 20)

# Since 'Y' is a binary variable, we use 'family = "binomial"'
fit <- saenet(x, y, pf, adWeight, weights, family = "binomial")
```

Index