Package ‘regnet’

March 29, 2022

Type Package

Title Network-Based Regularization for Generalized Linear Models

Version 0.5.0

Author Jie Ren, Luann C. Jung, Yinhao Du, Cen Wu, Yu Jiang, Junhao Liu

Maintainer Jie Ren <jieren@ksu.edu>

Description Network-based regularization has achieved success in variable selection for high-dimensional biological data due to its ability to incorporate correlations among genomic features. This package provides procedures of network-based variable selection for generalized linear models (Ren et al. (2017) <doi:10.1186/s12863-017-0495-5> and Ren et al. (2019) <doi:10.1002/gepi.22194>). Two recent additions are the robust network regularization for the survival response and the network regularization for continuous response. Functions for other regularization methods will be included in the forthcoming upgraded versions.

Depends R (>= 3.5.0)

License GPL-2

Encoding UTF-8

LazyData true

Imports glmnet, stats, Rcpp, igraph, utils

URL https://github.com/jrhub/regnet

BugReports https://github.com/jrhub/regnet/issues

RoxygenNote 7.1.2

LinkingTo Rcpp, RcppArmadillo, RcppThread

Suggests testthat, covr

Archs x64

R topics documented:

`regnet-package` *Network-Based Regularization for Generalized Linear Models*

Description

This package provides implementation of the network-based variable selection method proposed in Ren et al (2017) and the robust network-based method for survival response in Ren et al (2019). In addition to network penalty, regnet also allows users to use classical MCP or LASSO penalty.

Details

The easy-to-use, integrated interfaces cv.regnet() and regnet() allow users to flexibly choose the fitting methods they prefer. There are three arguments control the fitting method

response:	three types of response are supported: "binary", "continuous" and "survival".
penalty:	three choices of the penalty functions are available: "network", "mcp" and "lasso".
robust:	whether to use robust methods for modeling (Robust methods are only available for survival response for now).

In penalized regression, the tuning parameter λ_1 controls the sparsity of the coefficient profile. For network-based methods, an additional tuning parameter λ_2 is needed for controlling the smoothness among coefficient profiles. Typical usage of the package is to have the cv.regnet() compute the optimal values of lambdas, then provide them to the regnet() function for estimating the coefficients

If the users want to include clinical variables that are not subject to penalty in the model, the argument 'clv' can be used to indicate the positions of clinical variables in the X matrix. e.g. 'clv=(1:5)' meaning that the first five variables in X will not be penalized. It is recommended to put the clinical variables at the beginning of the X matrix in a contiguous way (see the 'Value' section of regnet() function). However, non-contiguous indices, e.g. 'clv=(2,4,6)', are also allowed.

References

Ren, J., Du, Y., Li, S., Ma, S., Jiang,Y. and Wu, C. (2019). Robust network-based regularization and variable selection for high dimensional genomics data in cancer prognosis. *Genet. Epidemiol.*, 43:276-291 doi: 10.1002/gepi.22194

Ren, J., He, T., Li, Y., Liu, S., Du, Y., Jiang, Y., and Wu, C. (2017). Network-based regularization for high dimensional SNP data in the case-control study of Type 2 diabetes. *BMC Genetics*, 18(1):44 doi: 10.1186/s1286301704955

Wu, C, Jiang, Y, Ren, J, Cui, Y, Ma, S. (2018). Dissecting gene-environment interactions: A penalized robust approach accounting for hierarchical structures. *Statistics in Medicine*, 37:437–456 doi: 10.1002/sim.7518

Wu, C., and Ma, S. (2015). A selective review of robust variable selection with applications in bioinformatics. *Briefings in Bioinformatics*, 16(5), 873–883 doi: 10.1093/bib/bbu046

Wu, C., Shi, X., Cui, Y. and Ma, S. (2015). A penalized robust semiparametric approach for gene-environment interactions. *Statistics in Medicine*, 34 (30): 4016–4030 doi: 10.1002/sim.6609

See Also

cv.regnet regnet

Examples

```
## Survival response using robust network method
data(SurvExample)
X = rgn.surv$X
Y = rgn.surv$Y
clv = c(1:5) # variables 1 to 5 are treated as clinical variables, we choose not to penalize them.
out = cv.regnet(X, Y, response="survival", penalty="network", clv=clv, robust=TRUE, verbo = TRUE)
out$lambda

fit = regnet(X, Y, "survival", "network", out$lambda[1,1], out$lambda[1,2], clv=clv, robust=TRUE)
index = which(rgn.surv$beta[-(1:6)] != 0)  # [-(1:6)] removes the intercept and clinical variables
pos = which(fit$coeff[-(1:6)] != 0)
tp = length(intersect(index, pos))
fp = length(pos) - tp
list(tp=tp, fp=fp)
```

---

cv.regnet *k-folds cross-validation for regnet*

Description

This function does k-fold cross-validation for regnet and returns the optimal value(s) of lambda.

Usage

```
cv.regnet(
  X,
  Y,
  response = c("binary", "continuous", "survival"),
  penalty = c("network", "mcp", "lasso"),
  lamb.1 = NULL,
  lamb.2 = NULL,
  folds = 5,
  r = NULL,
  clv = NULL,
  initiation = NULL,
  alpha.i = 1,
  robust = FALSE,
  ncores = 1,
  verbo = FALSE,
  debugging = FALSE
)
```

Arguments

- `X` X matrix as in regnet.
- `Y` response Y as in regnet.
- `response` response type. regnet now supports three types of response: "binary", "continuous" and "survival".
- `penalty` penalty type. regnet provides three choices for the penalty function: "network", "mcp" and "lasso".
- `lamb.1` a user-supplied sequence of λ_1 values, which serves as a tuning parameter to impose sparsity. If it is left as NULL, regnet will compute its own sequence.
- `lamb.2` a user-supplied sequence of λ_2 values for network method. λ_2 controls the smoothness among coefficient profiles. If it is left as NULL, a default sequence will be used.
- `folds` the number of folds for cross-validation; default is 5.
- `r` the regularization parameter in MCP; default is 5. For binary response, r should be larger than 4.
- `clv` a value or a vector, indexing variables that are not subject to penalty. clv only works for continuous and survival responses for now, and will be ignored for other types of responses.
- `initiation` the method for initiating the coefficient vector. The default method is elastic-net.
- `alpha.i` the elastic-net mixing parameter. The program can use the elastic-net for choosing initial values of the coefficient vector. alpha.i is the elastic-net mixing parameter, with $0 \leq$ alpha.i ≤ 1. alpha.i=1 is the lasso penalty, and alpha.i=0 is the ridge penalty. If the user chooses a method other than elastic-net for initializing coefficients, alpha.i will be ignored.
- `robust` logical flag. Whether or not to use robust methods. Robust methods are only available for survival response in the current version of regnet.
- `ncores` the number of cores to be used for parallelization.
- `verbo` output progress to the console.
- `debugging` logical flag. If TRUE, extra information will be returned.

Details

When lamb.1 is left as NULL, regnet computes its own sequence. You can find the lamb.1 sequence used by the program in the returned CVM matrix (see the 'Value' section). If you find the default sequence does not work well, you can try (1) standardize the response vector Y; or (2) provide a customized lamb.1 sequence for your data.

Sometimes multiple optimal values(pairs) of lambda(s) can be found (see 'Value'). This is usually normal when the response is binary. However, if the response is survival or continuous, you may want to check (1) if the sequence of lambda is too large (i.e. all coefficients are shrunken to zero under all values of lambda) ; or (2) if the sequence is too small (i.e. all coefficients are non-zero under all values of lambda). If neither, simply choose the value(pair) of lambda based on your preference.

Value

an object of class "cv.regnet" is returned, which is a list with components:

lambda — the optimal value(s) of λ. More than one values will be returned, if multiple lambdas have the cross-validated error = min(cross-validated errors). If the network penalty is used, lambda contains optimal pair(s) of λ_1 and λ_2.

mcvm — the cross-validated error of the optimal λ. For binary response, the error is misclassification rate. For continuous response, mean squared error (MSE) is used. For survival response, the MSE is used for non-robust methods, and the criterion for robust methods is least absolute deviation (LAD).

CVM — a matrix of the mean cross-validated errors of all lambdas used in the fits. The row names of CVM are the values of λ_1. If the network penalty was used, the column names are the values of λ_2.

References

Ren, J., Du, Y., Li, S., Ma, S., Jiang,Y. and Wu, C. (2019). Robust network-based regularization and variable selection for high dimensional genomics data in cancer prognosis. *Genet. Epidemiol.*, 43:276-291 doi: 10.1002/gepi.22194

Ren, J., He, T., Li, Y., Liu, S., Du, Y., Jiang, Y., and Wu, C. (2017). Network-based regularization for high dimensional SNP data in the case-control study of Type 2 diabetes. *BMC Genetics*, 18(1):44 doi: 10.1186/s1286301704955

See Also

regnet

Examples

```
## Binary response using network method
data(LogisticExample)
X = rgn.logi$X
Y = rgn.logi$Y
out = cv.regnet(X, Y, response="binary", penalty="network", folds=5, r = 4.5)
out$lambda
fit = regnet(X, Y, "binary", "network", out$lambda[1,1], out$lambda[1,2], r = 4.5)
index = which(rgn.logi$beta != 0)
pos = which(fit$coeff != 0)
tp = length(intersect(index, pos))
fp = length(pos) - tp
list(tp=tp, fp=fp)

## Binary response using MCP method
out = cv.regnet(X, Y, response="binary", penalty="mcp", folds=5, r = 4.5)
out$lambda
fit = regnet(X, Y, "binary", "mcp", out$lambda[1], r = 4.5)
index = which(rgn.logi$beta != 0)
pos = which(fit$coeff != 0)
tp = length(intersect(index, pos))
fp = length(pos) - tp
list(tp=tp, fp=fp)
```

`plot.regnet` *plot a regnet object*

Description

plot the network structures of the identified genetic variants.

Usage

```
## S3 method for class 'regnet'
plot(x, subnetworks=FALSE, vsize=10, labelDist=2, minVertices=2, theta=1, ...)
```

Arguments

`x`	regnet object.
`subnetworks`	whether to plot sub-networks
`vsize`	the size of the vertex
`labelDist`	the distance of the label from the center of the vertex.
`minVertices`	the minimum number of vertices a sub-network should contain.
`theta`	the multiplier fro the width of the edge. Specifically, $edge.width = \theta \times adjacency$. The defualt is 1.
`...`	other plot arguments

Details

This function depends on the "igraph" package in generating the network graphs. It returns a (list of) igraph object(s), on which users can do further modification on the network graphs.

Value

an object of class "igraph" is returned in default. When *subnetworks=TRUE*, a list of "igraph" objects (sub-networks) is returned.

See Also

`regnet`

Examples

```
data(ContExample)
X = rgn.tcga$X
Y = rgn.tcga$Y
clv = (1:2)
fit = regnet(X, Y, "continuous", "network", rgn.tcga$lamb1, rgn.tcga$lamb2, clv =clv, alpha.i=0.5)

plot(fit)
plot(fit, subnetworks = TRUE, vsize=20, labelDist = 3, theta = 5)
```

`print.cv.regnet` *print a cv.regnet object*

Description

Print a summary of a cv.regnet object

Usage

```
## S3 method for class 'cv.regnet'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	cv.regnet object.
`digits`	significant digits in printout.
`...`	other print arguments

See Also

`cv.regnet`

`print.regnet` *print a regnet object*

Description

Print a summary of a regnet object

Usage

```
## S3 method for class 'regnet'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

`x`	regnet object.
`digits`	significant digits in printout.
`...`	other print arguments

See Also

`regnet`

regnet *fit a regression for given lambda with network-based regularization*

Description

Network-based penalization regression for given values of λ_1 and λ_2. Typical usage is to have the cv.regnet function compute the optimal lambdas, then provide them to the regnet function. Users could also use MCP or Lasso.

Usage

```
regnet(
  X,
  Y,
  response = c("binary", "continuous", "survival"),
  penalty = c("network", "mcp", "lasso"),
  lamb.1 = NULL,
  lamb.2 = NULL,
  r = NULL,
  clv = NULL,
  initiation = NULL,
  alpha.i = 1,
  robust = FALSE,
  debugging = FALSE
)
```

Arguments

X	matrix of predictors without intercept. Each row should be an observation vector. A column of 1 will be added to the X matrix by the program as the intercept.
Y	response variable. For response="binary", Y should be a numeric vector with zeros and ones. For response="survival", Y should be a two-column matrix with columns named 'time' and 'status'. The latter is a binary variable, with '1' indicating a event, and '0' indicating censoring.
response	response type. regnet now supports three types of response: "binary", "continuous" and "survival".
penalty	penalty type. regnet provides three choices for the penalty function: "network", "mcp" and "lasso".
lamb.1	the tuning parameter λ_1 that imposes sparsity.
lamb.2	the tuning parameter λ_2 that controls the smoothness among coefficient profiles. λ_2 is needed for network penalty.
r	the regularization parameter in MCP. For binary response, r should be larger than 4.
clv	a value or a vector, indexing variables that are not subject to penalty. clv only works for continuous and survival responses for now, and will be ignored for other types of responses.
initiation	method for initiating the coefficient vector. The default method is elastic-net.

`alpha.i` the elastic-net mixing parameter. The program can use the elastic-net for choosing initial values of the coefficient vector. alpha.i is the elastic-net mixing parameter, with $0 \leq$ alpha.i ≤ 1. alpha.i=1 is the lasso penalty, and alpha.i=0 is the ridge penalty. If the user chooses a method other than elastic-net for initializing coefficients, alpha.i will be ignored.

`robust` logical flag. Whether or not to use robust methods. Robust methods are only available for survival response.

`debugging` logical flag. If TRUE, extra information will be returned.

Details

The current version of regnet supports two types of responses: “binary”, "continuous" and “survival”.

- regnet(. . . , response="binary", penalty="network") fits a network-based penalized logistic regression.
- regnet(. . . , response="continuous", penalty="network") fits a network-based least square regression.
- regnet(. . . , response="survival", penalty="network") fits a robust regularized AFT model using network penalty.

Please see the references for more details of the models. By default, regnet uses robust methods for survival response. If users would like to use non-robust methods, simply set robust=FALSE. User could also use MCP or Lasso penalty.

The coefficients are always estimated on a standardized X matrix. regnet standardizes each columns of X to have unit variance (using 1/n rather than 1/(n-1) formula). If the coefficients on the original scale are needed, the user can refit a standard model using the subset of variables that have non-zero coefficients.

Value

an object of class "regnet" is returned, which is a list with components:

`coeff` a vector of estimated coefficients. Please note that, if there are variables not subject to penalty (indicated by clv), the order of returned vector is c(Intercept, unpenalized coefficients of clv variables, penalized coefficients of other variables).

`Adj` a matrix of adjacency measures of the identified genetic variants. Identified genetic variants are those that have non-zero estimated coefficients.

References

Ren, J., He, T., Li, Y., Liu, S., Du, Y., Jiang, Y., and Wu, C. (2017). Network-based regularization for high dimensional SNP data in the case-control study of Type 2 diabetes. *BMC Genetics*, 18(1):44 doi: 10.1186/s1286301704955

Ren, J., Du, Y., Li, S., Ma, S., Jiang,Y. and Wu, C. (2019). Robust network-based regularization and variable selection for high dimensional genomics data in cancer prognosis. *Genet. Epidemiol.*, 43:276-291 doi: 10.1002/gepi.22194

See Also

`cv.regnet`

Examples

```
## Survival response
data(SurvExample)
X = rgn.surv$X
Y = rgn.surv$Y
clv = c(1:5) # variables 1 to 5 are clinical variables which we choose not to penalize.
penalty = "network"
fit = regnet(X, Y, "survival", penalty, rgn.surv$lamb1, rgn.surv$lamb2, clv=clv, robust=TRUE)
index = which(rgn.surv$beta != 0)
pos = which(fit$coeff != 0)
tp = length(intersect(index, pos))
fp = length(pos) - tp
list(tp=tp, fp=fp)
```

---

rgn *Example datasets for demonstrating the features of regnet*

Description

Example datasets for demonstrating the features of regnet.

Usage

```
data("LogisticExample")
data("SurvExample")
data("ContExample")
data("HeteroExample")
```

Format

"LogisticExample", "SurvExample" and "HeteroExample" are simulated data. Each data includes three main components: X, Y, and beta; beta is a vector of the true coefficients used to generate Y.

"ContExample" is a subset of the skin cutaneous melanoma data from the Cancer Genome Atlas (TCGA). The response variable Y is the log-transformed Breslow's depth. X is a matrix of gene expression data.

Examples

```
data("LogisticExample")
lapply(rgn.logi, class)
```

Index