Package 'regRSM'

May 9, 2026

Type Package

Title Random Subspace Method (RSM) for Linear Regression

Version 0.5

Date 2015-09-11

Author Pawel Teisseyre, Robert A. Klopotek

Maintainer Pawel Teisseyre <teisseyrep@ipipan.waw.pl>

Description Performs Random Subspace Method (RSM) for high-dimensional linear regression to obtain variable importance measures. The final model is chosen based on validation set or Generalized Information Criterion.

License LGPL-2 | LGPL-3 | GPL-2 | GPL-3

Depends R (>= 3.0.0)

Imports Rmpi, doParallel, parallel, foreach

URL http://www.ipipan.eu/~teisseyrep/SOFTWARE/

Archs x64

NeedsCompilation no

Repository CRAN

Date/Publication 2015-09-11 16:38:15

Contents

`ImpPlot.regRSM` *Variable importance plot from 'regRSM' object.*

Description

This function produces a dot plot showing final scores from RSM procedure.

Usage

```
## S3 method for class 'regRSM'
ImpPlot(object)
```

Arguments

`object`	Fitted 'regRSM' model object.

Details

This function produces a dot plot showing final scores from RSM procedure. Final scores describe importances of explanatory variables.

Author(s)

Pawel Teisseyre, Robert A. Klopotek.

Examples

```
p=100
n=100
beta1 = numeric(p)
beta1[c(1,5,10)]=c(1,1,1)
x = matrix(0,ncol=p,nrow=n)
for(j in 1:p){
    x[,j]=rnorm(n,0,1)
}
y = x %*% beta1 + rnorm(n)

p1=regRSM(x,y)
ImpPlot(p1)
```

plot.regRSM *Plot from 'regRSM' object.*

Description

This function produces a plot showing prediction errors on validation set (or the value of Generalized Information Criterion) with respect to the number of variables included in the model.

Usage

```
## S3 method for class 'regRSM'
plot(x,...)
```

Arguments

x	Fitted 'regRSM' model object.
...	Other arguments to plot.

Details

If Generalized Information Criterion (GIC) was used in the second step of RSM procedure (useGIC=TRUE) then the function produces a plot showing the value of GIC with respect to the number of variables included in the model. Model corresponding to the minimal value of GIC is chosen as a final one. If GIC was not used (useGIC=FALSE) and the validation set is supplied then the function produces a plot showing prediction errors on validation set with respect to the number of variables included in the model. Model corresponding to the minimal value of the prediction error is chosen as a final one.

Author(s)

Pawel Teisseyre, Robert A. Klopotek.

Examples

```
p=500
n=50
beta1 = numeric(p)
beta1[c(1,5,10)]=c(1,1,1)
x = matrix(0,ncol=p,nrow=n)
xval = matrix(0,ncol=p,nrow=n)
xtest = matrix(0,ncol=p,nrow=n)
for(j in 1:p){
    x[,j]=rnorm(n,0,1)
    xval[,j]=rnorm(n,0,1)
}
y = x %*% beta1 + rnorm(n)
yval = xval %*% beta1 + rnorm(n)
```

```
p1=regRSM(x,y)
plot(p1)

p2 = regRSM(x,y,yval,xval,useGIC=FALSE)
plot(p2)
```

`predict.regRSM` *Predictions from a 'regRSM' object.*

Description

This function makes predictions from a 'regRSM' object.

Usage

```
## S3 method for class 'regRSM'
predict(object, xnew,...)
```

Arguments

`object`	Fitted 'regRSM' model object
`xnew`	Matrix of new values for x at which predictions are to be made.
`...`	Additional arguments not used.

Details

Prediction is made based on a final model which is chosen using validation set or Generalized Information Criterion (GIC).

Value

predict.regRSM produces a vector of predictions.

Author(s)

Pawel Teisseyre, Robert A. Klopotek.

Examples

```
p = 100
n = 100
beta1 = numeric(p)
beta1[c(1,5,10)] = c(1,1,1)
x = matrix(0,ncol=p,nrow=n)
xtest = matrix(0,ncol=p,nrow=n)
for(j in 1:p){
    x[,j] = rnorm(n,0,1)
    xtest[,j] = rnorm(n,0,1)
}
```

```
y = x %*% beta1 + rnorm(n)
p1 = regRSM(x,y)
predict(p1,xtest)
```

`print.regRSM` *Print 'regRSM' object.*

Description

This function print the summary of the RSM procedure.

Usage

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

Arguments

`x`	Fitted 'regRSM' model object.
`...`	Additional arguments not used.

Details

The function prints out information about the selection method, screening, initial weights, version (sequential or parallel), size of the random subpace, number of simulations.

Author(s)

Pawel Teisseyre, Robert A. Klopotek.

Examples

```
p=100
n=100
beta1 = numeric(p)
beta1[c(1,5,10)]=c(1,1,1)
x = matrix(0,ncol=p,nrow=n)
xval = matrix(0,ncol=p,nrow=n)
xtest = matrix(0,ncol=p,nrow=n)
for(j in 1:p){
    x[,j]=rnorm(n,0,1)
}
y = x %*% beta1 + rnorm(n)

p1=regRSM(x,y)
print(p1)
```

regRSM *Random Subspace Method (RSM) for linear regression.*

Description

Performs Random Subspace Method (RSM) for high-dimensional linear regression to obtain variable importance measures. The final model is chosen based on validation set or Generalized Information Criterion.

Usage

```
## S3 method for class 'formula'
regRSM(formula, data=NULL, ...)
## Default S3 method:
regRSM(x, y, yval, xval, m, B, parallel, nslaves, store_data, screening,
init_weights, useGIC, thrs, penalty,...)
```

Arguments

formula	Formula describing the model to be fitted.
data	Data frame containing the variables in the model
y	Quatitative response vector of length n, where n is a number of observations.
x	Input matrix with n rows and p columns, where p is a number of variables. Each row is an observation vector.
yval	Optional quatitative response vector from validation set. Default is NULL.
xval	Optional input matrix from validation set. Default is NULL.
m	The size of the random subspace. Default is $min(n-1, p)/2$. Parameter m cannot be larger than the number of observations minus two.
B	Number of repetitions in RSM procedure. Default is 1000.
parallel	This argument indicates which version should be used. Default is NO, which indicates that sequential version isused. See also in details.
nslaves	Number of slaves. Default is 4.
store_data	Logical argument indicating whether matrix x and vector y should be stored. Default is FALSE. This argument must be TRUE when function validate.regRSM is used.
screening	If the screeing argument is in $(0, 1)$, the initial screening of variables is performed and 'screening %' of variables least correlated with the response are discarded. Then the RSM procedure is performaed on the remaining variables. Default is NULL, which indicates that no screening is used.
init_weights	This argument indicates whether weighted version of the procedure should be used. If the init_weights argument is TRUE, weighted version of the RSM procedure is used (called WRSM). In WRSM, variables are drawn to random subspaces with probabilities proportional to their correlations with the response y. Default is FALSE, which indicates that variables are drawn to subspaces with equal probabilities.

useGIC	Logical argument indicating whether Generalized Information Criterion (GIC) should be used in the second step of the procedure. Default is TRUE.
thrs	Cut off threshold. The hierarchical list of models given by the ordering of variables is cut off at level thrs in order to avoid fitting linear models which are close to saturated model. Default is thrs=p if $p < n/2$ and thrs=n/2 if $p > n/2$.
penalty	Penalty in Generalized Information Criterion (GIC). Default is $log(n)$ which corresponds to Bayesian Information Criterion (BIC).
...	other arguments not available now.

Details

The Random Subspace Method (RSM) is used to compute importance measures of explanatory variables (RSM final scores). In the second step the variables are ordered with respect to the final scores. From the nested list of models, given by the ordering, the final model is selected (the list is truncated at the level thrs to avoid fitting models which are close to saturated model). By default the final model that minimizes Generalized Information Criterion (GIC) is chosen. If the validation set is supplied and useGIC=FALSE then the final model that minimizes prediction error on validation set is selected.

When screening and weighted version are used together, in the first step screening is performed and then the weighted version (WRSM) is used on the reamining variables.

When parallel=NO sequential code is used for computaion. Else in our implementation the most costly operation (number of repetitions in RSM procedure) is parallelized. It is very inefficient to use parallelisation on machine with only single processor with singe core.

regRSM function in parallel mode does not close created slaves, because creation of slaves is usually very time consuming. Next parallel call will reuse existing slave processes. If you want change the number of slaves please execute mpi.close.Rslaves() (if parallel=MPI) and then call function regRSM with new parameter nslaves.

When parallel=POSIX then OpenMP like parallel implementation is used. This parallel execution is handled by doParallel library. It uses parallelisation of loops. The optimal value of nslaves is the number of prosessor cores in a machine.

When parallel=MPI then MPI parallel implementation is used. This parallel execution is handled by Rmpi library. MPI (Message Passing Interface) uses messages to send job tasks from main process (master) to other processes (slaves). These processes can be running not nessesery on one machine. In our implementation the most costly operation (number of repetitions in RSM procedure) is parallelized. The optimal value of nslaves is the number of computing cores of all machines configured in MPI framework. If only one machine is used, the best value is number of prosessor cores. If you don't want to use this kind of parallel computations any more remember to close MPI framework by calling mpi.close.Rslaves().

regRSM function in parallel mode does not close created slaves, because creation and destruction of slaves is usually very time consuming. Next parallel call will reuse existing slave processes. If you want change number of slaves please execute mpi.close.Rslaves() and then call function regRSM with new parameter nslaves. If you don't want to use parallel computations any more remember to close MPI framework by calling mpi.close.Rslaves().

Installing MPI for multiple machines:

In the following we give some guidelines how to install and configure MPI framework on multiple machines. MPI configuration on multiple machines is straightforward. Each machine must be connected to the main machine (master). While using **Rmpi** package we must remember that it works a little bit different than typical C MPI application. Usually master process transfers through MPI the whole application and replicates it on available slots (slaves). **Rmpi** uses existing R installation, so on each machine all required packages must be installed. Only R source code and data are transferred. We present the required steps under Ubuntu operating system (we use Ubuntu 12.04 LTS version). To install Open MPI on Ubuntu type:

```
sudo apt-get install libopenmpi-dev openmpi-bin
```

On Ubuntu with installed Open MPI and R one may just run R and type:

```
install.packages("Rmpi")
```

Consider a case when we have several (2 or more) machines with Ubuntu 12.04 LTS operating system, R 3.0, **Rmpi** and **regRSM** installed and all machines are connected to the same network. Moreover let's assume we have one network card which is mapped to `eth0`. With command `ifconfig` we can check what ip addresses our machines have. For simplicity, to avoid changing the configuration, we assign a static address to each machine. In our network we have 4 PCs with 4 core processor each. We give them the following names and ip addresses:

```
node09: 10.200.1.159
node08: 10.200.1.158
node07: 10.200.1.157
node06: 10.200.1.156
```

We create text file with ip and number of slots in each line. Slot is an instance of our application working in a slave mode. For example if we have a line `127.0.0.1 slots=4` then on our machine (localhost) MPI should run up to 4 slave processes. If we request more slaves than slots then there will be oversubscription of the node and the performance can drop. We can limit the number of slots to 4 by changing the line to `127.0.0.1 slots=4 max_slots=4`. In this case request on more than 4 processes on this node will result in an error. While setting hard limits one should remember that the total number of processes created by **Rmpi** package is equal to the number of slaves plus one (master process). For example if we want each computer to run 4 parallel tasks then we assign 4 slots to each machine. Example of our hostfile `myhosts`:

```
10.200.1.159 slots=4
10.200.1.157 slots=4
10.200.1.158 slots=4
10.200.1.156 slots=4
```

We run MPI application by executing:

```
mpiexec -n <no_of_program_copies> -hosts <file_with_hosts> <program_name>
```

Parameter `-n` can be misleading when working with `Rmpi` package. We want to start one R instance on which we run our experiment. Thus this value should be set to 1. To give **Rmpi** our hostfile just run command:

```
mpiexec -n 1 -hostfile myhosts R --no-save
```

which means we run one `Rscript` process with given hostfile for MPI configuration. In R terminal we type:

```
> library(Rmpi)
library(Rmpi)
> mpi.spawn.Rslaves()
mpi.spawn.Rslaves()
16 slaves are spawned successfully. 0 failed.
master (rank 0 , comm 1) of size 17 is running on: node09
slave1 (rank 1 , comm 1) of size 17 is running on: node09
slave2 (rank 2 , comm 1) of size 17 is running on: node09
slave3 (rank 3 , comm 1) of size 17 is running on: node09
... ... ...
slave15 (rank 15, comm 1) of size 17 is running on: node06
slave16 (rank 16, comm 1) of size 17 is running on: node09
```

The above lines indicate that all MPI processes are launched successfully.

Value

`scores`	RSM final scores.
`model`	The final model chosen from the list given by the ordering of variables according to the RSM scores.
`time`	Computational time.
`data_transfer`	Data transfer time.
`coefficients`	Coefficients in the selected linear model.
`input_data`	Input data `x` and `y`. These objects are stored only if `store_data=TRUE`.
`control`	List constining information about input parameters.
`informationCriterion`	Values of Generalized Information Criterion calculated for all models from the nested list given by the ordering.
`predError`	Prediction errors on validation set calculated for all models from the nested list given by the ordering.

Author(s)

Pawel Teisseyre, Robert A. Klopotek.

References

Mielniczuk, J., Teisseyre, P., *Using random subspace method for prediction and variable importance assessment in linear regression, Computational Statistics and Data Analysis, Vol. 71, 725-742, 2014.*

See Also

`predict`, `plot`, `ImpPlot`, `validate`, `roc` methods.

Examples

```
p = 500
n = 50
beta1 = numeric(p)
beta1[c(1,5,10)] = c(1,1,1)
x = matrix(0,ncol=p,nrow=n)
xtest = matrix(0,ncol=p,nrow=n)
for(j in 1:p){
    x[,j] = rnorm(n,0,1)
    xtest[,j] = rnorm(n,0,1)
}
y = x %*% beta1 + rnorm(n)
p1 = regRSM(x,y)

data1 = data.frame(y,x)
p2 = regRSM(y~.,data=data1)
```

`roc.regRSM` *ROC curve and AUC parameter.*

Description

This function produces ROC curve and computes AUC parameter.

Usage

```
## S3 method for class 'regRSM'
roc(object, truemodel, plotit, ...)
```

Arguments

`object`	Fitted 'regRSM' model object.
`truemodel`	User specified vector containing indexes of all significant variables.
`plotit`	Logical argument indicating whether a plot should be produced. If the value is `FALSE`, then the value of parameter AUC is returned. Default is `TRUE`.
`...`	Other arguments to plot.

Details

Let $i_1, \ldots, i_p$ be the ordering of variables (e.g. given by the RSM final scores), p is the number of all variables. ROC curve for ordering is defined as

$$\text{ROC}(s) := (FPR(s), TPR(s)), \quad s \in \{1, \ldots, p\},$$

where

$$FPR(s) := \frac{|SelectedModel(s) \setminus truemodel|}{|truemodel^C|},$$

$$TPR(s) := \frac{|SelectedModel(s) \cap truemodel|}{|truemodel|},$$

$$SelectedModel(s) := \{i_1, \ldots, i_s\},$$

$|A|$ denotes cardinality of A and A^C denotes a complement of A.

This function is useful for the evaluation of the ranking produced by the RSM procedure, when the set of significant variables is known (e.g. in the simulation experiments on artificial datasets). When AUC is equal one it means that all significant variables, suplied by the user in argment `truemodel`, are placed on the top of the ranking list.

Value

ROC curve is produced and the value of parameter AUC is returned.

Author(s)

Pawel Teisseyre, Robert A. Klopotek.

Examples

```
p=100
n=100
beta1 = numeric(p)
beta1[c(1,5,10)]=c(1,1,1)
x = matrix(0,ncol=p,nrow=n)
for(j in 1:p){
    x[,j]=rnorm(n,0,1)
}
y = x %*% beta1 + rnorm(n)
p1 = regRSM(x,y,store_data=TRUE)
true = c(1,5,10)
roc(p1,true,plotit=TRUE)
```

`summary.regRSM` *Print 'regRSM' object.*

Description

This function print the summary of the RSM procedure.

Usage

```
## S3 method for class 'regRSM'
summary(object,...)
```

Arguments

`object`	Fitted 'regRSM' model object.
`...`	Additional arguments not used.

Details

The function prints out information about the selection method, screening, initial weights, version (sequential or parallel), size of the random subpace, number of simulations.

Author(s)

Pawel Teisseyre, Robert A. Klopotek.

Examples

```
p=100
n=100
beta1 = numeric(p)
beta1[c(1,5,10)]=c(1,1,1)
x = matrix(0,ncol=p,nrow=n)
xval = matrix(0,ncol=p,nrow=n)
xtest = matrix(0,ncol=p,nrow=n)
for(j in 1:p){
    x[,j]=rnorm(n,0,1)
}
y = x %*% beta1 + rnorm(n)

p1=regRSM(x,y)
summary(p1)
```

---

validate.regRSM *Selects the new final model from existing 'regRSM' object.*

Description

This function selects the new final model based on the previously computed final scores.

Usage

```
## S3 method for class 'regRSM'
validate(object, yval, xval)
```

Arguments

object	Fitted 'regRSM' model object.
yval	Quantitative response vector from validation set.
xval	Input matrix from validation set.

Details

To use the function, the argument store_data in the 'regRSM' object must be TRUE. The function uses final scores from 'regRSM' object to create a ranking of variables. Then the final model which minimizes the prediction error on specified validation set is chosen. Object of class 'regRSM' is returned. The final scores in the original 'regRSM' object and in the new one coincide. However the final models can be different.

Value

Object of class 'regRSM' is returned.

Author(s)

Pawel Teisseyre, Robert A. Klopotek.

Examples

```
p=100
n=100
beta1 = numeric(p)
beta1[c(1,5,10)]=c(1,1,1)
x = matrix(0,ncol=p,nrow=n)
xval = matrix(0,ncol=p,nrow=n)
for(j in 1:p){
    x[,j]=rnorm(n,0,1)
    xval[,j]=rnorm(n,0,1)
}
y = x %*% beta1 + rnorm(n)
yval = xval %*% beta1 + rnorm(n)

p1 = regRSM(x,y,store_data=TRUE)
p2 = validate(p1,yval,xval)
```

Index